Optimal Dynamic Auctions for Cloud Markets

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

***Abstract*—Pricing cloud services 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 pay-as-you-go pricing in which the price cannot respond quickly to market dynamics (*e.g.*, Windows Azure). To overcome these problems, in this paper we propose to design *dynamic auctions* where virtual instances are periodically auctioned to accommodate user demands. With dynamic auctions, once the price is determined for a user, it remains unchanged during the remaining usage time, offering a guaranteed service. We show that our design is truthful and asymptotically optimal for large datacenters. By identifying certain optimization structures, we have significantly reduced the computational complexity of our solution. Extensive simulations show that our design closely tracks market changes, generating higher revenues than pay-as-you-go pricing.**

I. Introduction

Cloud computing revolutionizes the computer industry by commoditising computing services so that they can be traded and delivered in markets. To ensure business success, a cloud provider has to determine how its services can be optimally priced to cloud customers, so that its revenue is maximized. There exist three classes of pricing strategies that are currently adopted by cloud service providers: *flat rate*, *pay-as-you-go*, and *bid-based* pricing.

Flat rate pricing, such as Reserved Instances in Amazon EC2 [1], charges users a one-time payment to reserve one unit of resource (usually a virtual instance) for a long period of time (one to three years). Within the period, usage is either free or with a significant discount. Though flat rate pricing is preferred to cloud providers as it provides a long-term risk-free income, its upfront commitment makes it less attractive to users with flexible workloads.

With pay-as-you-go pricing, a per-unit price is set and users pay for only what they use. It has been widely adopted among cloud providers [1], [2], [3]. The per-unit price, however, is difficult to determine. On one hand, prices that are fixed or infrequently updated fail to capture market dynamics and usually bring less revenue to cloud providers. On the other hand, though frequent price updates are more agile responding to market changes, they introduce more uncertainty and are not desirable to users. Such increased risks may force users to give up on using services of this cloud provider entirely.

To compensate for such deficiency, Amazon has recently introduced Spot Instances, a bid-based pricing strategy [1]. With Spot Instances, users periodically submit bids to Amazon, who in turn posts a series of Spot Prices. Those users who bid higher than the Spot Price get their requests fulfilled. This bid-based pricing offers a 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 drawbacks in existing pricing alternatives, we are motivated to design a new pricing strategy that combines the advantages of pay-as-you-go and bid-based pricing: it can 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 virtual instances in these auctions, and every winning user is charged a price produced by the auction mechanism for each virtual instance. Requested instances are then held and later terminated by winning users.

A dynamic auction is fundamentally different from bid-based pricing. *First*, it is essentially an auction mechanism, in which all bidders have full knowledge about how winners and prices are determined, if all others' bids are available. In contrast, in bid-based pricing, how the price (*e.g.*, the Spot Price in Spot Instances) is set is unknown to bidders. *Second*, while a user's instances are forced to be terminated whenever the Spot Price goes above the submitted bid, dynamic auctions are able to offer guaranteed services: a winning user enjoys a guaranteed constant price, and its instances will never be terminated against its will.

It is important to make sure that the design of dynamic auctions is *truthful*. Being truthful eliminates the incentive of strategic behaviour, such as underbidding, that may significantly harm the auction integrity. In addition, it is desirable for the dynamic auction to generate as much revenue as possible for the cloud provider.

Towards achieving these objectives with optimal designs of dynamic auctions, the original contributions in this paper are two-fold. *First,* we present an optimal design for one auction period, which maximizes the revenue for a cloud provider. To do this, we extend the *Revenue Equivalence Theorem* [4], [5] and analytically characterize the revenue for any truthful auction carried out in one period. We show that the optimal design is at least as hard as a *0-1 knapsack problem*, and is too complex to be implemented in practice. For this reason, we turn to a simple alternative that is both truthful and asymptotically optimal, and our design approaches the optimal revenue when a datacenter has a large capacity.

Second, we extend our design for one auction period to the more general case of a *multistage* design, over a series of auction periods. We show that such a multistage design is equivalent to a *capacity control* problem: the provider can optimally reserve some capacity for future high-bid requests

and reject current low bidders. We model this problem as a Markov Decision Process, and by identifying certain optimal structures in the problem, we show that the computational complexity to solve the problem can be significantly reduced from $O(C^3)$ to $O(C^2)$, where C is the datacenter capacity.

The remainder of this paper is organized as follows. In Sec. II, we briefly survey the related work. Our model and notations are introduced in Sec. III. In Sec. IV, we characterize the revenue of a truthful auction in one period, and design a near-optimal mechanism with a simple structure. With this result, in Sec. V we model the multistage auction design as a capacity control problem and show that the computational complexity can be significantly reduced based on identified optimization structures in the problem. Extensive simulation studies are presented in Sec. VI. Sec. VII concludes the paper.

II. Related Work

Existing cloud pricing follows traditional designs in commodity markets, such as flat-rate reservation and pay-as-you-go pricing [1], [2], [3]. In contrast, designing auctions in cloud markets remains a relatively uncharted territory. Since the problem considered in this paper is closely related to mechanism design in economics, we briefly review papers that are most relevant.

The celebrated VCG auction [6] serves as a general framework for truthful mechanisms. However, it aims to maximize the *social welfare*, not the revenue, and is usually computationally prohibitive. Optimal mechanism design with maximum revenue is first considered in [4], where a single item is auctioned and the well-known *Revenue Equivalence Theorem* is proposed. This result is further generalized to cases where multiple items are auctioned [5]. Both of them rely on a standard economic assumption that every buyer bids for only one item, which is not the case in the cloud market. Other optimal designs, *e.g.*, [7], [8], though allowing bidding for multiple units, assume one-dimensional strategies — bidders can only misreport their prices, while truthfully submit quantities of requested units. In contrast, we require a two-dimensional design in the sense that cloud users truthfully report both instance prices and quantities. Although there are many works focusing on truthful designs with multidimensional strategies, they are usually too complicated to be applied in practice. Refer to [9] (Chapter 11) for an excellent survey of these auctions.

All the works above focus solely on mechanism design in one round. Other works, on the other hand, consider the design of *online auctions*, in which the seller must determine winners in the current period before observing bids in the future, *e.g.*, [10], [11], [12] and [9] (Chapter 16). These works focus more on being truthful against bidders misstating their arrival or departure time, and are suitable for scenarios where bidders know exactly their workloads and departure time at the very beginning. However, this is not true for cloud users with flexible workloads and random departure times.

[13] and [14] may be the most relevant to our work. They consider the problem of auctioning a certain amount of products to a sequence of buyers over time, and bridge mechanism design to capacity control. However, their models and techniques are not applicable to the cloud market due to two important reasons. *First*, they take standard economic assumptions and do not allow a bidder to bid for multiple units. *Second*, their models are based on the sales market, where products are sold to customers and are never available to sellers after a transaction. In contrast, cloud resources are *leased*, not sold, to users in a highly dynamic manner: auctioned instances are only temporarily unavailable to the cloud provider. After they are terminated by previous users, they can be hosted again to accommodate new requests. It is for this reason that conventional capacity control discussed in economics [15] cannot be directly applied to a cloud environment.

III. System Model

Suppose a cloud provider has allocated a fixed capacity C to host a type of virtual instances (*e.g.*, Amazon 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, such as Amazon 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 assume users are impatient: they will submit requests whenever their demands arise, and will turn to other cloud providers once their requests get rejected[1]. Each user i, $1 \le i \le N_t$, 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 at 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 p.d.f. $f_{n,v}$ on the support $[\underline{n}, \bar{n}] \times [\underline{v}, \bar{v}]$. Denote by $F_{n,v}$ the corresponding c.d.f. 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 [16], [17], 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 $b_i \neq v_i^t$ (or $r_i \neq n_i$) if it believes that this is more beneficial.

At the beginning of each period t, the provider carries out a mechanism $\mathcal{M}$ 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

[1]This is a standard economic assumption [15], [16].

instances are terminated by this user[2]. Specifically, let $n_{i,\tau}$ and $r_{i,\tau}$ be the number of needed instances and the number of rented instances at time $\tau \geq t$, respectively. Then we have $n_{i,t} = n_i$ and $r_{i,t} = r_i$. The utility of user i at time $\tau \geq t$ is defined by

$$u_{i,\tau} = \begin{cases} v_i n_{i,\tau} - p_i r_{i,\tau}\,, & \text{if } r_{i,\tau} \geq n_{i,\tau}; \\ 0\,, & \text{otherwise.} \end{cases} \tag{1}$$

Note that a rational user i will always rent exactly the same amount of instances as needed after p_i is set, *i.e.*, $r_{i,\tau} = n_{i,\tau}$ for all $\tau > t$. We define the overall utility for a winning user i by

$$u_i(r_i, b_i) = \begin{cases} \sum_{\tau=t}^{\infty} u_{i,\tau}\,, & \text{if } r_i \geq n_i; \\ 0\,, & \text{otherwise.} \end{cases} \tag{2}$$

For those rejected users, both the charged price and the overall utility are zero.

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

The cloud provider's problem is to design a sequence of auction mechanisms to maximize its expected revenue collected over time.

Furthermore, the designed mechanism $\mathcal{M}$ should possess some salient economic properties, notably *truthfulness*. Specifically, this requires that 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 provides accurate market information that could be utilized to predict future demands.

Denote by $\Gamma_{\mathcal{M}}^t$ the *overall revenue* of carrying out a mechanism $\mathcal{M}$ at time t. The value of $\Gamma_{\mathcal{M}}^t$ is calculated as the sum of the revenue collected over the entire time horizon, *i.e.*,

$$\Gamma_{\mathcal{M}}^t = \sum_{i=1}^{N_t} \sum_{j=1}^{n_i} p_i l_{i,j}, \tag{3}$$

where $l_{i,j}$ is the *life cycle* of instance j held for user i.

The cloud provider then aims to maximize $\mathbf{E}\left[\sum_{t=1}^{\infty} \Gamma_{\mathcal{M}}^t\right]$ with the constraint that at any time, the number of leased instances does not exceed the total capacity. However, optimally solving this problem requires complete information about the user demands for the entire future, which is impossible. As such, we consider two practical scenarios, with or without sufficient knowledge to predict the near future.

Unable to predict the future: In this case, the provider does not have sufficient information and cannot make any predictions. The objective is then reduced to myopically maximize the revenue collected in the current period, without considering future effects.

Able to predict the near future: In this case, the provider has accumulated sufficient information and is able to predict demands in the near future. The objective is to optimally decide winning users and charged prices at the current stage, such that the expected revenue collected in the predictable time span is maximized.

In the following two sections, we detail the formulation and analysis of these two problems, and present optimal and near-optimal truthful designs for revenue maximization.

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

IV. Single-Stage Mechanism Design

We consider single-stage truthful designs with the aim to generate as much revenue as possible in a single period. Specifically, Let C_t be the available capacity at time t. Our problem is to design an optimal mechanism $\mathcal{M}$ such that $\mathbf{E}\left[\sum_{i=1}^{N_t} p_i r_i\right]$ is maximized subject to $\sum_{i=1}^{N_t} r_i \leq C_t$. The designed mechanism $\mathcal{M}$ is also required to be truthful for two-dimensional bids (r_i, b_i).

A. Characterizing Revenue for Truthful Mechanism

First we present a revenue characterization theorem for any truthful mechanism. Proposition 1 extends Myerson's *Revenue Equivalence Theorem* [4] to the two-dimensional domain.

Proposition 1 (Revenue Characterization): Let $\mathbf{v} = (v_i)$ and $\mathbf{n} = (n_i)$. Denote by $\gamma_{\mathcal{M}}$ the revenue of a mechanism $\mathcal{M}$ with two-dimensional bids. Then for any truthful $\mathcal{M}$, we have

$$\mathbf{E}_{\mathbf{n},\mathbf{v}}[\gamma_{\mathcal{M}}] = \mathbf{E}_{\mathbf{n},\mathbf{v}}\left[\sum_{i=1}^{N_t} n_i \phi(v_i) x_i(\mathbf{n}, \mathbf{v})\right]. \tag{4}$$

Here, $\phi(v_i) = v_i - \frac{1 - F_v(v_i|n_i)}{f_v(v_i|n_i)}$ with $f_v(\cdot|n)$ (resp. $F_v(\cdot|n)$) being the conditional p.d.f. (resp. c.d.f.) of the reservation value, and $x_i(\mathbf{n}, \mathbf{v})$ takes the value 0 or 1 depending on whether user i loses or wins, respectively.

Proof:

$$\begin{aligned} \mathbf{E}_{\mathbf{n},\mathbf{v}}[\gamma_{\mathcal{M}}] &= \mathbf{E}_{\mathbf{n}}\mathbf{E}_{\mathbf{v}}\left[\sum_{i=1}^{N_t} p_i n_i\right] \\ &= \mathbf{E}_{\mathbf{n}}\left[\sum_{i=1}^{N_t} \mathbf{E}_{\mathbf{v}_{-i}}\mathbf{E}_{v_i|\mathbf{v}_{-i}}[p_i n_i]\right], \end{aligned} \tag{5}$$

where $\mathbf{v}_{-i}$ is obtained by removing the ith component v_i from $\mathbf{v}$. Now consider $\mathbf{E}_{v_i|\mathbf{v}_{-i}}[p_i n_i]$. It has been shown in [9] that, when $\mathbf{n}$ and $\mathbf{v}_{-i}$ are given, a truthful mechanism $\mathcal{M}$ always offers a *take-it-or-leave-it* payment[3], say ρ_i, for bidder i, who wins when bidding higher than ρ_i. Therefore,

$$\begin{aligned} \mathbf{E}_{v_i|\mathbf{v}_{-i}}[p_i n_i] &= n_i \rho_i (1 - F_v(\rho_i|n_i)) \\ &= n_i \int_{\rho_i}^{\bar{v}} \left(z - \frac{1 - F_v(z|n_i)}{f_v(z|n_i)}\right) f_v(z|n_i) \mathrm{d}z \\ &= \mathbf{E}_{v_i|\mathbf{v}_{-i}}\left[n_i \phi(v_i) x_i(\mathbf{n}, \mathbf{v})\right], \end{aligned} \tag{6}$$

where the second equality can be verified by performing integration by parts on the right-hand side (RHS). We conclude the proof by substituting (6) back into (5). ■

Proposition 1 greatly simplifies revenue analysis. Since $\mathcal{M}$ is truthful, for each user i, $b_i = v_i$ and $r_i = n_i$.

[3]Such payment relies on $\mathcal{M}$, $\mathbf{n}$ and $\mathbf{v}_{-i}$, but is independent of v_i.

Therefore, both n_i and $\phi(v_i)$ are available to the provider. In this case, designing an optimal auction to maximize the expected revenue is equivalent to finding a set of x_i such that the RHS of (4) is maximized, under the constraint that the accommodated requests do not exceed the currently available capacity. Formally, an optimal auction solves the following optimization problem:

$$\begin{aligned} \max_{x_i \in \{0,1\}} \quad & \sum_{i=1}^{N_t} n_i \phi(v_i) x_i \\ \text{s.t.} \quad & \sum_{i=1}^{N_t} n_i x_i \leq C_t. \end{aligned} \tag{7}$$

For mathematical convenience, we take a standard *regularity assumption* that $\phi(\cdot)$ is increasing [4], [5], [18]. This assumption generally holds for most distributions [4].

B. The Complexity of Optimal Design

To see the complexity of optimal design, recall that equation (7) essentially describes a *0-1 knapsack problem*, where C_t is the sack capacity, and n_i and $\phi(v_i)$ are item weight and item value per unit of weight, respectively. We therefore have the following proposition.

Proposition 2: Designing an optimal truthful auction that maximizes the expected revenue is at least as hard as solving a 0-1 knapsack problem described by (7).

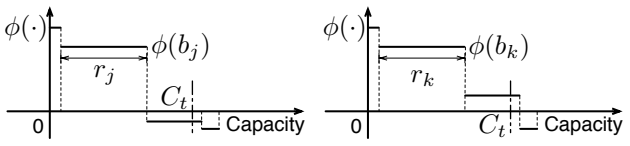
In fact, the truthful design is more complicated than just solving a knapsack problem. To see this, we consider a simplified scenario where each user i is assumed to truthfully report n_i and only has a one-dimensional strategy on b_i. Algorithm 1 gives the optimal design for this simplified scenario.

Algorithm 1 Optimal Truthful Mechanism for One-Dimensional Scenario

1. Solve (7) and denote by γ^* the optimal revenue obtained
2. Every user i with $x_i = 1$ wins the auction
3. **for all** winning user i **do**
4. Solve (7) as if user i is absent
5. Let γ^*_{-i} be the optimal revenue obtained above
6. $p_i = \phi^{-1}((\gamma^*_{-i} - \gamma^*)/n_i + \phi(v_i))$
7. **end for**

Proposition 3: Algorithm 1 maximizes the expected revenue and is truthful for one-dimensional strategies $\mathbf{b} = (b_i)$.

The proof of Proposition 3 is given in Appendix A. We see that Algorithm 1 is a complex design: winners are determined by solving a knapsack problem and are charged different prices calculated through an involved process. Such complexity significantly discourages user participation and makes the mechanism difficult to implement. Since the design is already complicated for a simplified scenario (bidders do not misreport the number of requested instances), we conclude that the optimal mechanism is not suitable to be used in practical markets.



(a) $\sum_{i=1}^{j} r_i \leq C_t$. (b) $\sum_{i=1}^{j} r_i > C_t$.

Fig. 1. Illustration of two cases in Algorithm 2. (a) Top j bidders win, each paying $\phi^{-1}(0)$. (b) Top k bidders win, each paying b_{k+1}.

C. A Near-Optimal Design

Due to the complexity of optimal auctions, we now consider a near-optimal design that has a simple structure and is easy to implement. We show that it is truthful for two-dimensional bids and achieves excellent performance guarantees.

Without loss of generality, we assume bidders are sorted in decreasing order of submitted prices, *i.e.*, $b_1 \geq b_2 \geq \cdots \geq b_{N_t}$. Due to the regularity assumption, we know that $\phi(b_1) \geq \phi(b_2) \geq \cdots \geq \phi(b_{N_t})$. Problem (7) can be approximately solved in a greedy fashion: sequentially placing items into the sack, from the top valued (highest $\phi(b_i)$) to the bottom, until there is no longer space for more. Note that those negatively valued items (*i.e.*, $\phi(b_i) \leq 0$) will always be discarded. Algorithm 2 is designed based on this process, where we assume there is at least one profitable request, *i.e.*, $\phi(b_i) > 0$ for some i.

Algorithm 2 A Near-Optimal Design

1. $j = \arg\max_i \{\phi(b_i) > 0\}$
2. **if** $\sum_{i=1}^{j} r_i \leq C_t$ **then**
3. Top j bidders win, each paying $\phi^{-1}(0)$
4. **else**
5. Let k be the index such that $\sum_{i=1}^{k} r_i \leq C_t < \sum_{i=1}^{k+1} r_i$
6. Top k bidders win, each paying b_{k+1}
7. **end if**

Two cases are considered in Algorithm 2, with or without sufficient capacity to accommodate all profitable requests (*i.e.*, the request with $\phi(b_i) > 0$), as shown in Figs. 1a and 1b, respectively. In either case, a uniform price is applied to all winning bidders[4]. Below we show the surprising result that the mechanism, though simple, is both two-dimensionally truthful and asymptotically optimal.

D. Two-Dimensional Truthfulness

To see that Algorithm 2 is truthful, we require the following lemma. The intuition can be deduced by consulting the illustrations in Figs. 1a and 1b. We give the proof sketch in Appendix B.

Lemma 1: For every bidder i, fix all others' submissions. Denote by $p(r_i, b_i)$ the uniform price calculated by Algorithm 2 when bidder i submits (r_i, b_i). Then for all b_i (resp. r_i), $p(r_i, b_i)$ is increasing w.r.t. r_i (resp. b_i).

Lemma 1 suggests Lemma 2, whose proof is given in Appendix B.

[4]Note that if $r_1 > C_t$, then $k = -1$, which means that no one wins.

TABLE I
FOUR POSSIBLE OUTCOMES WHEN BIDDING TRUTHFULLY OR UNTRUTHFULLY

	Case 1	**Case 2**	**Case 3**	**Case 4**
Truthful	Win	Win	Lose	Lose
Untruthful	Win	Lose	Win	Lose

Lemma 2: For every bidder i, there is no advantage to overbook instances, *i.e.*, given b_i, $u_i(r_i, b_i) \leq u_i(n_i, b_i)$ for all $r_i > n_i$.

By noting that no user has the incentive to request fewer instances than needed (*i.e.*, $r_i < n_i$), we see that Lemma 2 essentially indicates that users will truthfully report their n_i's. This leads to the truthfulness of Algorithm 2, as stated below.

Proposition 4: Algorithm 2 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), $i = 1, 2, \ldots, N_t$.

Proof Sketch: It suffices to consider the four cases of possible outcomes of bidding truthfully or untruthfully, as listed in Table I. Since every user i tends to truthfully report n_i, the untruthful submission is (n_i, b_i) where $b_i \neq v_i$. Let k be defined the same as that in Algorithm 2.

Case 1. Since both (n_i, b_i) and (n_i, v_i) lead to a winning result, it is easy to verify that $p(n_i, b_i) = p(n_i, v_i) = p$, where p is either $\phi^{-1}(0)$ or b_{k+1}. As a result, the statement holds with $u_i(n_i, v_i) = u_i(n_i, b_i)$.

Case 2. In this case, the statement trivially holds.

Case 3. In this case, user i loses by bidding (n_i, v_i). We can easily verify that $p(n_i, v_i) \geq v_i$. Now by changing the submission to (n_i, b_i), user i wins. This implies that $b_i > v_i$. By Lemma 1, $p(n_i, b_i) \geq p(n_i, v_i) \geq v_i$. Hence, $u_i(n_i, b_i) = \sum_{\tau=t}^{\infty} n_{i,\tau}(v_i - p(n_i, b_i)) \leq 0 = u_i(n_i, v_i)$, where the first equality is derived from (2).

Case 4. In this case, the statement trivially holds. ∎

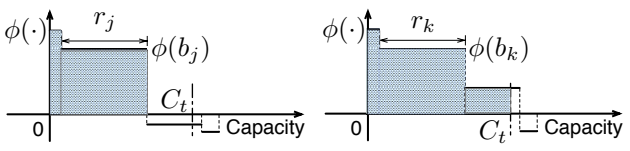
E. Asymptotic Optimality

Though simple, Algorithm 2 achieves a close approximation in general and is asymptotically optimal. To see this, let γ be the revenue obtained by Algorithm 2. Now relax problem (7) by allowing fractional x_i's, *i.e.*, $x_i \in [0, 1]$. Denote by $\bar{\gamma}$ the solution to the relaxed problem (7). Let j and k be defined the same as those in Algorithm 2. We have

$$\bar{\gamma} = \begin{cases} \sum_{i=1}^{j} r_i \phi(b_i), & \text{if } \sum_{i=1}^{j} r_i \leq C_t; \\ \sum_{i=1}^{k} r_i \phi(b_i) + \left(C_t - \sum_{i=1}^{k} r_i\right) \phi(b_{k+1}), & \text{otherwise.} \end{cases} \tag{8}$$

The value of $\bar{\gamma}$ is illustrated as the shaded areas in Figs. 2a and 2b. Clearly, $\gamma \leq \gamma^* \leq \bar{\gamma}$, where γ^* is the solution to the original knapsack problem (7). The value of $\bar{\gamma}$ is the *revenue upper bound*.

Note that in a cloud market, the available capacity of a datacenter is usually enormous compared with a single user's request. In this case, Algorithm 2 nearly achieves the maximum revenue, as stated in Proposition 5.



(a) $\sum_{i=1}^{j} r_i \leq C_t$. (b) $\sum_{i=1}^{j} r_i > C_t$.

Fig. 2. Revenue upper bound in two cases, shown as the shaded area.

Proposition 5: If $C_t \geq \alpha \bar{n}$ for some $\alpha > 1$, Algorithm 2 is $\frac{\alpha}{\alpha-1}$-competitive. Specifically, $\gamma \geq (1 - \frac{1}{\alpha})\bar{\gamma} \geq (1 - \frac{1}{\alpha})\gamma^*$.

Proof: It is easy to see that $\gamma = \gamma^* = \bar{\gamma}$ when $\sum_{i=1}^{j} r_i \leq C_t$. For the case of $\sum_{i=1}^{j} r_i > C_t$, it suffices to prove $\gamma \geq (1 - \frac{1}{\alpha})\bar{\gamma}$. Due to the truthfulness, $r_i = n_i$ and $b_i = v_i$. We then have

$$\begin{aligned} \bar{\gamma} - \gamma &= \left(C_t - \textstyle\sum_{i=1}^{k} n_i\right) \phi(v_{k+1}) \\ &\leq \bar{n} \cdot \phi(v_{k+1}) \\ &\leq \tfrac{1}{\alpha} C_t \phi(v_{k+1}), \end{aligned}$$

where the first equality is derived from (8). Now inspect Fig. 2b. The value of $C_t\phi(v_{k+1})$ can be explained as the area of a rectangle with width C_t and height $\phi(v_{k+1})$, which is clearly no more than the shaded area $\bar{\gamma}$. Therefore $\bar{\gamma} - \gamma \leq \frac{1}{\alpha}\bar{\gamma}$. This is exactly the statement. ∎

We therefore conclude that Algorithm 2 is asymptotically optimal as α becomes sufficiently large, *i.e.*, $\alpha \gg 1$.

Even in the case where a datacenter does not have too much capacity left available, the revenue gap between Algorithm 2 and the optimal design is still acceptable, as stated below.

Proposition 6: In all cases, $\gamma^* - \gamma \leq \bar{\gamma} - \gamma \leq \bar{n}\bar{v}$.

Proof: It suffices to consider the case when $\sum_{i=1}^{j} r_i > C_t$. By (8), we have $\bar{\gamma} - \gamma = \left(C_t - \sum_{i=1}^{k} n_i\right) \phi(v_{k+1}) \leq \bar{n}\phi(\bar{v})$. By noting that $\phi(\bar{v}) = \bar{v}$, we conclude the proof. ∎

Take On-Demand Instances in Amazon EC2 as an example. For a typical user, n_i is usually less than 100, while v_i is usually in the range of $[0.08, 0.10]$ (m1.small, Linux [1]). By Proposition 6, this implies that even unfairly compared with the impractical optimal design, Algorithm 2 only results in a revenue loss of \$10 per period in the worst case.

Due to the simplicity, truthfulness, and performance guarantees, we argue that Algorithm 2 is the best single-stage auction in practice.

V. MULTISTAGE MECHANISM DESIGN

Based on our single-stage analysis in Sec. IV, we are now ready to present our design of multistage auctions.

A. Myopic Design

Perhaps the most straightforward design is a *myopic* strategy — iteratively carrying out the optimal single-stage auction as described in Sec. IV, without considering future effects. This is a reasonable design in two cases. (1) There is always sufficient capacity available to accommodate user requests; (2) the provider does not have sufficient information and is unable to predict future demands. In the first case, such a myopic design is optimal because supply shortages never happen. In

the second case, since the future is unpredictable, the provider makes the best effort by maximizing the currently obtainable revenue.

B. Optimal Capacity Control

When demand prediction is possible and the available capacity is limited, the provider can increase its revenue via capacity control. It can optimally reserve some capacity for future high-bid requests and reject current low bidders.

Suppose at time t, the available resources are capable of holding Q additional virtual instances. However, only C_t of them are allocated to accommodate current demands, while the remaining $Q - C_t$ capacity is reserved for future high-bid requests. Let $\Gamma_{\mathcal{M}}(C_t)$ be the overall revenue of carrying out a mechanism $\mathcal{M}$ at time t. By (3), we know $\Gamma_{\mathcal{M}}(C_t) = \sum_{i=1}^{N_t}\sum_{j=1}^{n_i} p_i l_{i,j}$, where $l_{i,j}$ is the life cycle of instance j held for user i. We assume life cycles are i.i.d. exponential. 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[5]. We now calculate the expected overall revenue as follows.

$$\begin{aligned}\mathbf{E}[\Gamma_{\mathcal{M}}(C_t)] &= \mathbf{E}\left[\sum_{i=1}^{N_t}\sum_{j=1}^{n_i} p_i l_{i,j}\right] \\ &= \sum_{i=1}^{N_t}\sum_{j=1}^{n_i} p_i \cdot \mathbf{E}[l_{i,j}] \\ &= \frac{1}{q}\sum_{i=1}^{N_t} p_i n_i = \frac{1}{q}\gamma_{\mathcal{M}}(C_t). \end{aligned} \tag{9}$$

Here, $\gamma_{\mathcal{M}}(C_t)$ is the single-stage revenue collected in the current period.

By making short predictions, at time t, a provider knows N_τ for $\tau \le t+w$, with w being some prediction window. Let $T = t+w$ and denote by $V_t(Q)$ the optimal revenue obtainable from periods $t, t+1, \ldots, T$ given that the available capacity is Q at time t. Suppose C_t capacity is allocated to time t. Then right before period $t+1$, there are $C - Q + C_t$ instances being held. If K of them are terminated by users, then at $t+1$, there will be $Q - C_t + K$ instances available, which can bring $V_{t+1}(Q - C_t + K)$ revenue to the provider. Since each hosted instance has a probability q to be terminated in the next period, we know K follows a binomial distribution with p.m.f. $P(K = k) = B(C - Q + C_t, k, q)$, where $B(n, k, q) = \binom{n}{k} q^k (1-q)^{n-k}$. Therefore, the expected revenue of allocating C_t instances to time t is $\mathbf{E}_K[V_{t+1}(Q - C_t + K)]$.

The provider's problem is to decide the optimal capacity allocation C_t^* for the current period such that the overall revenue $V_t(Q)$ is maximized. This can be formulated as a Markov Decision Process:

$$V_t(Q) = \mathbf{E}_{\mathbf{n},\mathbf{v}}\left[\max_{C_t=0,1,\ldots,Q}\left\{\frac{1}{q}\gamma_{\mathcal{M}}(C_t) + \nu_{t+1}(Q - C_t)\right\}\right], \tag{10}$$

where $\nu_{t+1}(\cdot)$ is defined by

$$\begin{aligned}\nu_{t+1}(m) &= \mathbf{E}_K[V_{t+1}(m+K)] \\ &= \sum_{k=0}^{C-m} B(C-m, k, q) V_{t+1}(m+k)\end{aligned} \tag{11}$$

for all $m \ge 0$. For $m < 0$, we define $\nu_{t+1}(m) = -\infty$. The boundary conditions are $V_{T+1}(Q) = 0, Q = 0, 1, \ldots, C$.

After the optimal allocation C_t^* is decided, the provider then auctions it to current users via a mechanism $\mathcal{M}$. From previous analysis in Sec. IV, we know that the best $\mathcal{M}$ is Algorithm 2. The provider now aims to solve equation (10) in which $\gamma_{\mathcal{M}}(C_t)$ is replaced by $\gamma(C_t)$, the single-stage revenue obtained by Algorithm 2.

Since no closed-form solution exists, a standard method is to apply *numerical dynamic programming*. The computation starts from period T and proceeds backward to t. For each Q and each realization $(\mathbf{n}, \mathbf{v})$, compute equation (10), which takes $O(C^2)$ operations. Repeat this process until the optimal C_t^* is obtained. Since w is usually small ($w \le 10$), it can be viewed as a constant. The overall computational complexity is therefore $O(C^3)$.

C. Reducing Computational Complexity

Directly solving (10) is computationally prohibitive. In reality, a large datacenter is usually capable of holding tens of thousands of instances of a certain type simultaneously, *i.e.*, $C = 10^4$. In this case, direct computation requires $O(10^{12})$ operations, which is almost impossible to complete within a short time. To overcome this problem, we design an approximation solution that can significantly reduce the complexity to $O(C^2)$ while achieving near-optimal performance.

Let $\bar{\gamma}(C_t)$ be the upper bound of single-stage revenue defined by (8). Instead of directly solving (10) to get $V_t(Q)$, we compute its upper bound $\bar{V}_t(Q)$ defined as follows.

$$\bar{V}_t(Q) = \mathbf{E}_{\mathbf{n},\mathbf{v}}\left[\max_{C_t=0,1,\ldots,Q}\left\{\frac{1}{q}\bar{\gamma}(C_t) + \bar{\nu}_{t+1}(Q - C_t)\right\}\right], \tag{12}$$

where $\bar{\nu}_{t+1}(\cdot)$ is similarly defined in (11) by replacing V_{t+1} by $\bar{V}_{t+1}$. The boundary conditions are $\bar{V}_{T+1}(Q) = 0, Q = 0, 1, \ldots, C$. Since $\bar{\gamma}(C_t)$ is the revenue upper bound for any single-stage mechanism, $\bar{V}_t(Q)$ is the upper bound of overall revenue for any multistage auction. We show later that $\bar{V}_t(\cdot)$ can be efficiently obtained within $O(C^2)$.

Denote by $\bar{C}_t^*$ the optimal solution to (12). That is, $\bar{V}_t(Q)$ is maximized when $\bar{C}_t^*$ capacity is allocated to period t. We then auction $\bar{C}_t^*$ instances to current users, using Algorithm 2.

The intuition of the above approximation comes from Proposition 5. Let $\widetilde{V}$ be the approximation's overall revenue collected over time. Since $\bar{C}_t^*$ capacity is auctioned via Algorithm 2, the expected overall revenue of the approximation is

[5]This is essentially a binomial model that is widely adopted in economics to model many random environments, such as option pricing [19].

calculated by

$$\mathbf{E}[\widetilde{V}] = \mathbf{E}\left[\sum_{t=1}^{\infty}\Gamma(\bar{C}_t^*)\right] = \mathbf{E}\left[\sum_{t=1}^{\infty}\frac{1}{q}\gamma(\bar{C}_t^*)\right], \tag{13}$$

where $\Gamma(\bar{C}_t^*)$ is similarly defined in (9) with $\mathcal{M}$ being Algorithm 2. Note that for a large datacenter, the allocated capacity is usually enormous compared with an individual's request, *i.e.*, in most cases, $\bar{C}_t^* > \alpha\bar{n}$ for some $\alpha \gg 1$. In this sense, Proposition 5 comes into play: $\gamma(\bar{C}_t^*) \geq (1-\frac{1}{\alpha})\bar{\gamma}(\bar{C}_t^*)$. Substituting it back to (13), we have,

$$\begin{aligned}\mathbf{E}[\widetilde{V}] &\geq (1-\frac{1}{\alpha})\cdot\mathbf{E}\left[\sum_{t=1}^{\infty}\frac{1}{q}\bar{\gamma}(\bar{C}_t^*)\right]\\ &= (1-\frac{1}{\alpha})\cdot\mathbf{E}[\bar{V}],\end{aligned} \tag{14}$$

where $\bar{V}$ is the upper bound of the overall revenue. This implies that the approximation closely approaches the optimal revenue for large datacenters. We later verify this point by extensive simulations in Sec. VI.

We now analyze the complexity of the above approximation. This is equivalent to discussing the computational complexity of solving (12). It is interesting to show that (12) contains some special structures. Taking use of these structures results in a significant reduction in the amount of required computation.

First, we see that $\bar{\gamma}(C_t)$ is concave w.r.t. C_t, as stated in Lemma 3. The proof sketch is given in Appendix C.

Lemma 3: Given $\mathbf{n}$ and $\mathbf{v}$, $\bar{\gamma}(C_t)$ defined in (8) is concave, *i.e.*, $\nabla\bar{\gamma}(C_t) = \bar{\gamma}(C_t) - \bar{\gamma}(C_t-1)$ is decreasing w.r.t. C_t.

Besides the concavity of $\bar{\gamma}(\cdot)$, Lemma 4 states that both $\bar{V}_t(\cdot)$ and $\bar{\nu}_{t+1}(\cdot)$ are also concave. The proof is given in Appendix C.

Lemma 4: For every $\tau = t, t+1, \ldots, T$, both $\bar{V}_\tau(\cdot)$ and $\bar{\nu}_{\tau+1}(\cdot)$ are increasing and concave in $\mathbb{Z}^* = \{0, 1, \ldots\}$.

These concavities lead to a special optimization structure as described in the following proposition. Its proof is given in Appendix C.

Proposition 7: For every realization $\mathbf{n}$ and $\mathbf{v}$ at time $\tau = t, t+1, \ldots, T$, let $\bar{C}_\tau^*(Q)$ be the optimal allocation such that $\bar{V}_\tau(Q)$ is maximized. Then we have

$$\bar{C}_\tau^*(Q+1) - 1 \leq \bar{C}_\tau^*(Q) \leq \bar{C}_\tau^*(Q+1). \tag{15}$$

Proposition 7 plays a key role in reducing computational complexity. It shows that previously calculated results can be leveraged by the following computation. For each period τ, $\bar{V}_\tau(Q)$ is sequentially computed as $Q = C, C-1, \ldots, 0$. Though the first computation ($Q = C$) requires searching the entire solution space $C_\tau = 0, 1, \ldots, C$, subsequent computations are much more efficient. By Proposition 7, once $\bar{C}_\tau^*(Q+1)$ is calculated, one can quickly determine $\bar{C}_\tau^*(Q)$ by choosing between two candidates, $\bar{C}_\tau^*(Q) = \bar{C}_\tau^*(Q+1)$ and $\bar{C}_\tau^*(Q) = \bar{C}_\tau^*(Q+1) - 1$, and the one resulting in higher revenue $\bar{V}_\tau(Q)$ is the optimal solution. The entire computation only takes $O(C^2)$ operations[6]. We note that the complexity reduction from $O(C^3)$ to $O(C^2)$ is significant, especially for large datacenters. As an example, when $C = 10000$, the above simplification speeds up the computation by 10000 times.

Finally, since a user does not have complete market information as the provider does, it cannot predict future changes and deduce how capacity allocation is carried out. As a result, its strategies are limited within one auction period. This implies the truthfulness of multistage design, as every single-stage auction is proved to be truthful in Sec. IV.

VI. Simulation Results

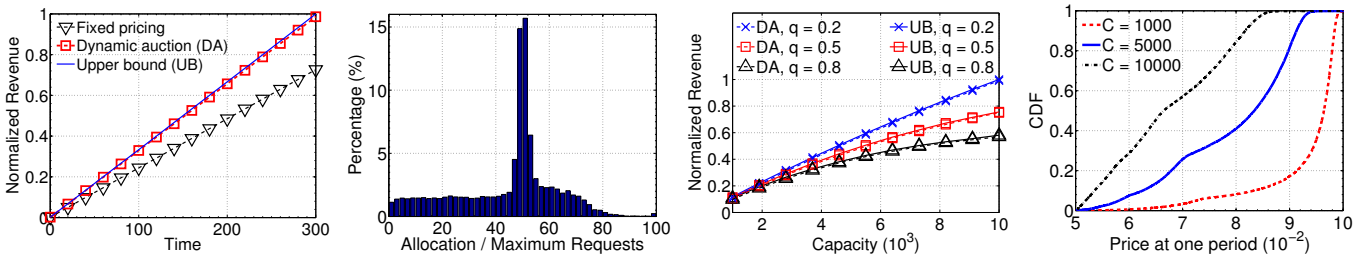
We evaluate the revenue performance of the proposed dynamic auction via extensive simulations. For convenience, denote $U_{\mathbb{Z}}(a,b)$ and $U_{\mathbb{R}}(a,b)$ the uniform distributions defined in integer and real domains, respectively, in the range of $[a,b]$.

We adopt a typical scenario where $C = 10000$. That is, the datacenter is able to host up to 10000 virtual instances of a certain type 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 r_i instances with price b_i. We set $N_t \sim U_{\mathbb{Z}}(1,300)$, $r_i \sim U_{\mathbb{Z}}(1,100)$, and $b_i \sim U_{\mathbb{R}}(0.05, 0.1)$. We enable short predictions and set the prediction window $w = 5$. It is worth mentioning that though the evaluation presented in this section are based on i.i.d. settings, similar results are also observed in non-i.i.d. cases. Each result below has been averaged over 1000 runs.

Revenue vs. Time. We first evaluate the proposed dynamic auction by comparing its revenue against traditional pay-as-you-go with a fixed price in a long time span (at least two weeks in practice). Since it is almost impossible to predict long future's demands, a widely adopted way is to set a price p^* such that $p^*(1-F_v(p^*))$ is maximized [20]. 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. 3a illustrates the revenue performance of both dynamic auction and fixed pricing, where q is set to 0.5. Since i.i.d. demands are adopted in the simulation, the revenue is observed to be linearly dependent on the time. We see that the designed auction accurately captures market dynamics and outperforms fixed pricing, leading to 30% revenue enhancement. Also note that our design closely approaches the theoretical revenue upper bound, $\bar{V}$, with a performance gap that is less than 1%. This is further verified by Fig. 3b, where a histogram is presented to show statistics of the ratio of allocated capacity $\bar{C}_t^*$ to the maximum individual request $\bar{n}$. Specifically, over 85% of such ratios exceed 20, and more than 70% exceed 40. These statistics support the argument made in Sec. V: in general, the allocated capacity is much larger than any single user's request, which leads to a strong performance guarantee

[6]The complexity $O(C^2)$ includes the calculation of binomial coefficients used in (11), since it does not incur additional complexities by using the following property: $B(n,k,q) = pB(n-1,k-1,q)+(1-q)B(n-1,k,q)$.



(a) Normalized revenue vs. time. (b) Histogram of $\bar{C}_t^*/\bar{n}$. (c) Normalized revenue vs. capacity. (d) CDF of calculated prices.

Fig. 3. Performance evaluation of dynamic auctions (DA) and the revenue upper bounds (UB) in 300 simulation periods, with $C = 10000$, $w = 5$, $N_t \sim U_{\mathbb{Z}}(1, 300)$, $r_i \sim U_{\mathbb{Z}}(1, 100)$, and $b_i \sim U_{\mathbb{R}}(0.05, 0.1)$. Unless explicitly stated, q is set to 0.5. All data are averaged over 1000 runs.

for the designed mechanism. Similar results are also observed when other q's are adopted.

Revenue vs. Capacity. We next verify the structural results obtained in Sec. V 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. 3c illustrates the results, where the solid line represents the theoretical revenue upper bound obtained by solving (12), i.e., $\mathbf{E}[\bar{V}]$ in (14), 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 3%. Also, the gap is found decreasing when more capacity is available in the datacenter. In this case, more capacity is allocated to each period, so that a better performance guarantee is achieved, which validated our statement that our design is preferred by large datacenters.

Another important observation is that all three upper bounds are increasing and concave functions of the capacity, verifying the structural analysis made in Lemma 4. It is interesting to see that the concavity is gradually increased when the market becomes more dynamic (*i.e.*, with larger q). Note that only q is changed in the setting, so that increasing q essentially decreases the entire workloads and revenues, as the expected usage time (lifetime) of every instance is shortened. In this case, adding capacity brings less marginal revenue to the provider, which increases the revenue concavity.

Capacity vs. Price Dynamics. Our final observation is that the price becomes more dynamic as capacity increases. Fig. 3d shows the c.d.f. 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 $C = 10000$).

VII. Conclusions

In this paper, we investigate the problem of designing efficient pricing policies to charge for cloud services. Specifically, we advocate the use of dynamic auctions to quickly adapt to market changes while offering guaranteed services. To this end, we analytically characterize the revenue of truthful auctions, and formulate their optimal design as a capacity control problem in a leasing market. Further, our analysis shows the drawback of optimal design: it is too complicated for users to understand and is almost impractical to exercise in practice. For this reason, we turn to a near-optimal auction with a simple structure. Theoretical analysis shows that the design closely approaches the optimal revenue as long as the capacity of datacenter is large, which is naturally the case in reality. Finally, we derive optimization structures of the capacity control problem and reduce the computational complexity from $O(C^3)$ to $O(C^2)$, which is significant for cloud providers with large datacenter capacities. All our theoretical results are further validated by extensive simulation studies.

Appendix A
Analysis of Truthfulness of Algorithm 1

For every user i, fix all others' submissions. Clearly, maximizing u_i is equivalent to maximizing $v_i - p_i$.

Denote by p_i^t and p_i^u the charged price to user i when i bids truthfully and untruthfully, respectively. We first show $u_i(n_i, v_i) \geq 0$ for all $i = 1, 2, \ldots, N_t$. Since $\gamma_{-i}^* \leq \gamma^*$, we have $(\gamma_{-i}^* - \gamma^*)/n_i + \phi(v_i) \leq \phi(v_i)$. Due to the monotonicity of $\phi(\cdot)$, we know $p_i^t \leq v_i$. In this sense, $u_i(n_i, v_i) = \sum_{\tau=t}^{\infty} n_{i,\tau}(v_i - p_i^t) \geq 0$.

We now prove the truthfulness of Algorithm 1.

Proof of Proposition 4: It suffices to consider the four cases of possible outcomes of bidding truthfully or untruthfully, as listed in Table I. For every user i, fix all others' submissions. Denote by $\mathbf{x}^t = (x_i^t)$ (resp. γ_t^*) and $\mathbf{x}^u = (x_i^u)$ (resp. γ_u^*) the auction outcomes (resp. optimal revenue) when user i bids truthfully and untruthfully, respectively.

Case 1: In this case, $\mathbf{x}^t = \mathbf{x}^u$. We then have

$$\begin{aligned}\gamma_t^* - n_i\phi(v_i) &= \sum_{j\neq i} n_j\phi(b_j)x_j^t \\ &= \sum_{j\neq i} n_j\phi(b_j)x_j^u = \gamma_u^* - n_i\phi(b_i)\,. \end{aligned} \tag{16}$$

Hence, we know

$$p_i^t = \phi^{-1}\left(\frac{\gamma_{-i}^* - \gamma_t^* + n_i\phi(v_i)}{n_i}\right) = \phi^{-1}\left(\frac{\gamma_{-i}^* - \gamma_u^* + n_i\phi(b_i)}{n_i}\right) = p_i^u\,, \tag{17}$$

which implies

$$u_i(n_i, v_i) = \sum_{\tau=t}^{\infty} n_{i,\tau}(v_i - p_i^t) = \sum_{\tau=t}^{\infty} n_{i,\tau}(v_i - p_i^u) = u_i(n_i, b_i)\,.$$

Case 2: In this case, $u_i(n_i, b_i) = 0 \le u_i(n_i, v_i)$ and the statement trivially holds.

Case 3: In this case, we show $p_i(b_i) \ge v_i$ and $u_i(n_i, b_i) \le 0$. This is equivalent to proving $\gamma_u^* - n_i\phi(b_i) + n_i\phi(v_i) \le \gamma_{-i}^*$. Since user i loses by submitting v_i, we know $x_i^t = 0$ and $\gamma_{-i}^* = \gamma_t^*$.

We now prove by contradiction. Suppose otherwise $\gamma_u^* - n_i\phi(b_i) + n_i\phi(v_i) > \gamma_t^*$. Then we have

$$\begin{aligned}\text{LHS} &= \sum_{j\neq i} n_j\phi(b_j)x_j^u + n_i\phi(v_i)\\ &= \sum_{j\neq i} n_j\phi(b_j)x_j^u + n_i\phi(v_i)x_i^u\\ &> \text{RHS} = \sum_{j\neq i} n_j\phi(b_j)x_j^t + n_i\phi(v_i)x_i^t\,,\end{aligned}$$

where the second equality is derived from the fact that user i wins by bidding untruthfully (*i.e.*, $x_i^u = 1$). Also note that $\sum_{i=1}^{N_t} n_i x_i^u \le C_t$. One can conclude that $\mathbf{x}^u$ is a better auction outcome with higher revenues than $\mathbf{x}^t$ when user i bids truthfully. Clearly, this contradicts the fact that $\mathbf{x}^t$ is the best auction outcome.

Case 4: The statement trivially holds for this case. ∎

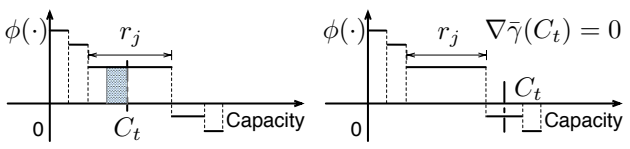
APPENDIX B
ANALYSIS OF OVERBOOKING BEHAVIOURS

In this section, we analyze overbooking behaviours of users. We prove Lemma 2 and show that overbooking does not increase user utility. To simplify the analysis, we artificially insert a virtual user l with $(r_l, b_l) = (\infty, \phi^{-1}(0))$. We then avoid discussing two cases in Algorithm 2 since $\sum_{i=1}^{j} r_i = \infty$ and the uniform price computed is b_{k+1}.

We first prove Lemma 1.

Proof sketch of Lemma 1: First, we show $p(r_i, b_i)$ is increasing w.r.t r_i for all b_i. It is easy to see that if user i loses by submitting (r_i, b_i), then increasing r_i does not change the produced price. For the case where i wins by bidding (r_i, b_i), increasing r_i decreases k, leading to an increased b_{k+1}. In either case, $p(r_i, b_i)$ is increasing w.r.t. r_i.

We next see $p(r_i, b_i)$ is increasing w.r.t. b_i for all r_i. Let k be calculated the same as that in Algorithm 2 when user i bids $b_i = \underline{v}$. Then as b_i increases, the ranking of user i goes above. The determined price $p(r_i, b_i)$ remains unchanged until user i becomes the $(k+1)$-th bidder. The price then starts to increase until user i wins, after which $p(r_i, b_i)$ is stable. ∎



(a) $C_t \le \sum_{i=1}^{j} r_i$. (b) $C_t > \sum_{i=1}^{j} r_i$.

Fig. 4. $\nabla\bar{\gamma}(C_t)$ in two cases, shown as the shaded area.

We then need the following technical lemma.

Lemma 5: If bidder i wins by submitting (r_i, b_i), then it also wins by submitting (r_i', b_i), where $r_i' < r_i$.

Proof: It is easy to see that user i wins if and only if $\phi(b_i) > 0$ and $\sum_{l=1}^{i} r_l \le C_t$. Because changing r_i to r_i' does not change $\phi(b_i)$, user i remains the ith highest bidder. Noting that $\sum_{l=1}^{i-1} r_l + r_i' \le \sum_{l=1}^{i} r_l \le C_t$, we conclude that i also wins by submitting $r_i' < r_i$. ∎

We now complete the proof of Lemma 2.

Proof of Lemma 2: Since the statement trivially holds if i loses by requesting more instances than needed, we consider the case where i wins by submitting (r_i, b_i), where $r_i > n_i$. By Lemma 5, bidder i also wins by bidding (n_i, b_i). Then by Lemma 1, $p(r_i, b_i) \ge p(n_i, b_i)$. By (2), we have

$$\begin{aligned}u_i(r_i, b_i) &= \sum_{\tau=t+1}^{\infty} n_{i,\tau}(v_i - p(r_i, b_i)) + v_i n_i - p(r_i, b_i)\cdot r_i\\ &< \sum_{\tau=t+1}^{\infty} n_{i,\tau}(v_i - p(n_i, b_i)) + v_i n_i - p(n_i, b_i)\cdot n_i\\ &= u_i(n_i, b_i).\end{aligned}$$

∎

APPENDIX C
ANALYSIS OF OPTIMIZATION STRUCTURES

In this section, we analyze the optimization structures in problem (12). We start by proving Lemma 3.

Proof Sketch of Lemma 3: We explain the pictorial proofs of Figs. 4a and 4b. Given $\mathbf{n}$ and $\mathbf{v}$, let j be defined the same as that in Algorithm 2. Figs. 4a and 4b intuitively show $\nabla\bar{\gamma}(C_t)$ for two cases when $C_t \le \sum_{i=1}^{j} r_i$ and $C_t > \sum_{i=1}^{j} r_i$, respectively. Clearly, increasing C_t decreases $\nabla\bar{\gamma}(C_t)$. ∎

We next prove Lemma 4. This requires the following two technical lemmas.

Lemma 6: Suppose f is a convex (resp. concave) discrete function. Define g by $g(n) = \sum_{k=0}^{n} B(n, k, q) f(k)$. Then $g(\cdot)$ is also convex (resp. concave).

Proof: For notational simplicity, we write $f(n)$ by f_n and $g(n)$ by g_n. Without loss of generality, we assume $f_0 = 0$. Let $\nabla g_n = g_n - g_{n-1}$ and $\nabla^2 g_n = \nabla g_n - \nabla g_{n-1}$ be the first- and second-order forward differences of $g(\cdot)$, respectively. Similar definition also applies to ∇f_n and $\nabla^2 f_n$. When there is no confusion, we simply write $B(n, k, q)$ as $B_{n,k}$. We next prove

$$\nabla^2 g_{n+1} = q^2 \sum_{k=0}^{n-1} B_{n-1,k}\nabla^2 f_{k+2}. \tag{18}$$

By definition, we have

$$\begin{aligned}\nabla g_n &= g_n - g_{n-1} \\ &= \sum_{k=0}^{n} B_{n,k} f_k - \sum_{k=0}^{n-1} B_{n-1,k} f_k \\ &= q \sum_{k=1}^{n-1} \left[B_{n-1,k-1} - B_{n-1,k}\right] f_k + q^n f_n. \end{aligned} \tag{19}$$

Here the third equality holds because $f_0 = 0$ and

$$B_{n,k} = qB_{n-1,k-1} + (1-q)B_{n-1,k}. \tag{20}$$

For convenience, we define $B_{n,k} = 0$ for all $k < 0$. The following equation is obtained by plugging (19) into (18) and applying (20):

$$\begin{aligned}&\frac{1}{q^2}\nabla^2 g_{n+1} \\ &= \sum_{k=1}^{n-1} [B_{n-1,k-2} + B_{n-1,k} - 2B_{n-1,k-1}] f_k \\ &\quad + [B_{n-1,n-2} - 2B_{n-1,n-1}] f_n + B_{n-1,n-1} f_{n+1}. \end{aligned} \tag{21}$$

Fixing n, we define α_m by

$$\alpha_m = \sum_{k=1}^{m-1} [B_{n-1,k-2} + B_{n-1,k} - 2B_{n-1,k-1}] f_k, \tag{22}$$

and define β_m by

$$\beta_m = [B_{n-1,m-2} - 2B_{n-1,m-1}] f_m + B_{n-1,m-1} f_{m+1}. \tag{23}$$

Noticing that $\alpha_m = 0$ for all $m < 2$, we can write (21) by

$$\frac{1}{q^2}\nabla^2 g_{n+1} = \alpha_n + \beta_n. \tag{24}$$

We now investigate the structure of $\alpha_m + \beta_m$ for all m. First, we have

$$\begin{aligned}\beta_m &= [B_{n-1,m-2} - 2B_{n-1,m-1}] f_m + B_{n-1,m-1} f_{m+1} \\ &= [B_{n-1,m-2} - B_{n-1,m-1}] f_m + B_{n-1,m-1} \nabla f_{m+1} \\ &= -B_{n-1,m-1} f_{m-1} + B_{n-1,m-2} f_m \\ &\quad + B_{n-1,m-1} \nabla^2 f_{m+1}. \end{aligned} \tag{25}$$

Substituting (25) back to $\alpha_m + \beta_m$ leads to the following equalities:

$$\begin{aligned}&\alpha_m + \beta_m \\ &= \alpha_{m-1} + \beta_m \\ &\quad + [B_{n-1,m-3} + B_{n-1,m-1} - 2B_{n-1,m-2}] f_{m-1} \\ &= \alpha_{m-1} + B_{n-1,m-1} \nabla^2 f_{m+1} \\ &\quad + [B_{n-1,m-3} - 2B_{n-1,m-2}] f_{m-1} + B_{n-1,m-2} f_m \\ &= \alpha_{m-1} + \beta_{m-1} + B_{n-1,m-1} \nabla^2 f_{m+1} \qquad (26) \\ &= \sum_{k=0}^{m-1} B_{n-1,k} \nabla^2 f_{k+2}, \qquad (27) \end{aligned}$$

where (27) is obtained by recursively applying (26). With (27) and (24), we see (18) holds, which concludes the proof. ■

Lemma 7: Let f_1 and f_2 be two discrete functions that are concave, and define the function f by

$$f(n) = \max_{k=0,1,\ldots,n} \{f_1(k) + f_2(n-k)\}. \tag{28}$$

Then f is also concave.

Proof: For any discrete function h, let $\bar{h}$ be its linear interpolation. That is, for all x, let $n = \lfloor x \rfloor$. We have

$$\bar{h}(x) = h(n) + \Delta h(n) \cdot (x - n). \tag{29}$$

Given n, we define $g_k = f_1(k) + f_2(n-k)$. It is easy to see that $\bar{g}(x) = \bar{f}_1(x) + \bar{f}_2(n-x)$. Since $\bar{g}$ is a linear interpolation of a discrete function g, we know its maximum value is achieved at the integeter point, *i.e.*,

$$\begin{aligned} f(n) &= \max_{k=0,1,\ldots,n} g_k \\ &= \max_{0 \le x \le n} \bar{g}(x) \\ &= \max_{0 \le x \le n} \{\bar{f}_1(x) + \bar{f}_2(n-x)\}. \end{aligned} \tag{30}$$

Let $f_0(y)$ be defined by

$$f_0(y) = \max_{0 \le x \le y} \{\bar{f}_1(x) + \bar{f}_2(y-x)\}. \tag{31}$$

Since both f_1 and f_2 are concave, their linear interpolations, $\bar{f}_1$ and $\bar{f}_2$, are also concave. By [21] (Rule 9 of Theorem 3.1.5), we know f_0 is concave and piecewise linear. Also note that $f_0(n) = f(n)$ for all integer n. We conclude that f_0 is a linear interpolation of f, *i.e.*, $f_0 = \bar{f}$. In this case, f is concave. ■

We now move to prove Lemma 4.

Proof of Lemma 4: We omit the proof of monotonicity and focus only on the concavity statement. We prove by induction.

Basis: Show that the statement holds for $\tau = T$. In this case, since $\bar{\gamma}(\cdot)$ is increasing, we know that $\bar{V}_T(Q) = \mathbf{E}[\frac{1}{q}\bar{\gamma}(Q)]$, which is concave by Lemma 3. It is easy to see that $\bar{\nu}_{T+1}(m) = 0$ for all $m \ge 0$. We conclude that the statement holds for the basis.

Inductive step: Suppose the statement holds for τ. We now show it also holds for $\tau - 1$. Let $n = C - m$. We have

$$\begin{aligned} \bar{\nu}_\tau(m) &= \textstyle\sum_{k=0}^{C-m} B(C-m,k,q) \bar{V}_\tau(m+k) \\ &= \textstyle\sum_{k=0}^{n} B(n,k,q) \bar{V}_\tau(C-n+k) \\ &= \textstyle\sum_{k=0}^{n} B(n,n-k,1-q) \bar{V}_\tau(C-(n-k)) \\ &= \textstyle\sum_{l=0}^{n} B(n,l,1-q) \bar{V}_\tau(C-l). \end{aligned} \tag{32}$$

Due to the induction assumptions, $\bar{V}_\tau(\cdot)$ is concave. Applying Lemma 6 to (32), we show the concavity of $\bar{\nu}_\tau(\cdot)$.

For $\bar{V}_{\tau-1}(\cdot)$, we inspect its definition (12). Since both $\bar{\nu}_\tau(\cdot)$ and $\bar{\gamma}(\cdot)$ are concave, applying Lemma 7 to (12) leads to the concavity of $\bar{V}_{\tau-1}$. ■

We are now ready to show the optimization structures by proving Proposition 7.

Proof of Proposition 7: For any discrete function f, define $\Delta f(n) = f(n+1) - f(n)$. The following lemma is useful.

Lemma 8: For any $k = 0, 1, \ldots, Q$, $\bar{C}_t^*(Q) \le k$ if $\frac{1}{q}\Delta\bar{\gamma}(k) \le \Delta\bar{\nu}_{\tau+1}(Q-k-1)$, and $\bar{C}_t^*(Q) > k$ otherwise.

Proof of Lemma 8: Let $L(C_t) = \frac{1}{q}\bar{\gamma}(C_t) + \bar{\nu}_{\tau+1}(Q - C_t)$. Then L is maximized at $\bar{C}_t^*$. Since both $\bar{\gamma}(\cdot)$ and $\bar{\nu}_{\tau+1}(\cdot)$ are concave (Lemma 4), $L(\cdot)$ is also concave. In this case, $\bar{C}_t^* \leq k$ if $\Delta L(k) \leq 0$, and $\bar{C}_t^* > k$ otherwise. Note that $\Delta L(k) = \frac{1}{q}\Delta\bar{\gamma}(k) - \Delta\bar{\nu}_{\tau+1}(Q - k - 1)$. We see that the statement holds. □

We are now ready to prove Proposition 7. Suppose $\bar{C}_t^*(Q) = k$. We first show $\bar{C}_t^*(Q+1) \leq k+1$ by (33).

$$\begin{aligned} \tfrac{1}{q}\Delta\bar{\gamma}(k+1) &\leq \tfrac{1}{q}\Delta\bar{\gamma}(k) \\ &\leq \Delta\bar{\nu}_{\tau+1}(Q-k-1) \\ &= \Delta\bar{\nu}_{\tau+1}((Q+1)-(k+1)-1). \end{aligned} \tag{33}$$

Here, the first inequality holds due to the concavity of $\bar{\gamma}(\cdot)$, and the second inequality is due to the fact that $\bar{C}_t^*(Q) = k$. Now applying Lemma 8 to (33), we see that $\bar{C}_t^*(Q+1) \leq k+1$.

Next, we prove $\bar{C}_t^*(Q+1) > k-1$. It suffices to consider the case where $k > 0$. We have

$$\begin{aligned} \tfrac{1}{q}\Delta\bar{\gamma}(k-1) &\geq \Delta\bar{\nu}_{\tau+1}(Q-(k-1)-1) \\ &\geq \Delta\bar{\nu}_{\tau+1}((Q+1)-(k-1)-1), \end{aligned} \tag{34}$$

where the first inequality is due to the fact that $\bar{C}_t^*(Q) > k-1$, and the second inequality is derived from the concavity of $\Delta\bar{\nu}_{\tau+1}(\cdot)$. By Lemma 8, this implies $\bar{C}_t^*(Q+1) > k-1$. ■

REFERENCES

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

[2] Windows Azure Pricing, http://www.microsoft.com/windowsazure/pricing/.

[3] Google AppEngine Pricing, http://code.google.com/appengine/docs/billing.html.

[4] R. B. Myerson, "Optimal Auction Design," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 58–73, 1981.

[5] J. Bulow and J. Roberts, "The Simple Economics of Optimal Auctions," *Journal of Political Economy*, vol. 97, no. 5, pp. 1060–1090, 1989.

[6] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[7] G. Aggarwal and J. Hartline, "Knapsack Auctions," in *Proc. 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006, pp. 1083–1092.

[8] A. Archer and É. Tardos, "Truthful Mechanisms for One-Parameter Agents," in *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001.

[9] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. Cambridge University Press, 2007.

[10] E. Friedman and D. Parkes, "Pricing WiFi at Starbucks: Issues in Online Mechanism Design," in *Proc. 4th ACM Conf. on Electronic Commerce (EC)*, 2003, pp. 240–241.

[11] D. C. Parkes and S. Singh, "An MDP-Based Approach to Online Mechanism Design," in *Proc. 17th Annual Conf. on Neural Information Processing Systems (NIPS)*, 2003.

[12] M. Hajiaghayi, R. D. Kleinberg, M. Mahdian, and D. C. Parkes, "Online Auctions with Re-usable Goods," in *Proc. 6th ACM Conf. on Electronic Commerce (EC)*, 2005.

[13] G. Vulcano, G. Van Ryzin, and C. Maglaras, "Optimal Dynamic Auctions for Revenue Management," *Management Science*, vol. 48, no. 11, pp. 1388–1407, 2002.

[14] J. Gallien, "Dynamic Mechanism Design for Online Commerce," *Operations Research*, vol. 54, no. 2, pp. 291–310, 2006.

[15] K. Talluri and G. Van Ryzin, *The Theory and Practice of Revenue Management*. Springer Verlag, 2005.

[16] 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.

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

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

[19] J. Hull, *Options, Futures and Other Derivatives*. Pearson Prentice Hall, 2009.

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

[21] J. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer Verlag, 2001.