# When Cloud Meets eBay: Towards Effective Pricing for Cloud Computing

Qian Wang†, Kui Ren†, and Xiaoqiao Meng‡
†Department of ECE, Illinois Institute of Technology, Chicago, IL 60616. Email: {qian, kren}@ece.iit.edu
‡IBM T.J. Watson Research Center, Hawthorne, NY 10532. Email: xmeng@us.ibm.com

***Abstract*—The rapid deployment of cloud computing promises network users with elastic, abundant, and on-demand cloud services. The pay-as-you-go model allows users to be charged only for services they use. Current purchasing designs, however, are still primitive with significant constraints. *Spot Instance*, the first deployed auction-style pricing model of Amazon EC2, fails to enforce fair competition among users in facing of resource scarcity and may thus lead to untruthful bidding and unfair resource allocation. Dishonest users are able to abuse the system and obtain (at least) short-term advantages by deliberately setting large maximum price bids while being charged only at lower *Spot Prices*. Meanwhile, this may also prevent the demands of honest users from being satisfied due to resource scarcity. Furthermore, *Spot Instance* is inefficient and may not adequately meet users' overall demands because it limits users to bid for each computing instance individually instead of multiple different instances at a time.**

**In this paper, we formulate and investigate the problem of cloud resource pricing. We propose a suite of computationally efficient and truthful auction-style pricing mechanisms, which enable users to fairly compete for resources and cloud providers to increase their overall revenue. We analytically show that the proposed algorithms can achieve truthfulness without collusion or $(t, p)$-truthfulness tolerating a collusion group of size $t$ with probability at least $p$. We also show that the two proposed algorithms have polynomial complexities $O(nm + n^2)$ and $O(nm)$, respectively, when $n$ users compete for $m$ different computing instances with multiple units. Extensive simulations show that, in a competitive cloud resource market, the proposed mechanisms can increase the revenue of cloud providers, especially when allocating relatively limited computing resources to a potentially large number of cloud users.**

## I. Introduction

Cloud Computing, the new term for the long dreamed vision of computing as a utility, enables convenient, on-demand network access to a centralized pool of configurable computing resources, *e.g.*, networks, servers, applications, and services, that can be rapidly deployed with great efficiency and minimal management overhead [1], [2]. By providing resizable compute capacity through virtual computing environments, cloud infrastructure services allow cloud users to use web service interfaces to launch instances and choose the number, the size and the configuration of the compute instances they need for their applications [3]. However, while technologists naturally focus on powerful features for building flexible, scalable, failure resilient applications, the important role of pricing, which could be a necessary condition of realizing successful cloud services, has so far been largely overlooked.

In cloud computing, the ultimate goal of pricing is to bring real social and economic benefits to cloud users and give them the flexibility to optimize the costs. Currently, most of existing cloud service providers adopt a fixed-price pricing model. As an example, Amazon EC2 utilizes *On-Demand Instance* that allows users to pay a fixed rate by the hour with no commitment and/or *Reserved Instance* that allows users to pay a low, one-time fee, in turn receiving a significant discount on the hourly usage charge [3]. However, these fixed-pricing models have an obvious drawback. That is, they cannot effectively reflect the underlying trends in demand and supply for the computing resources. On the one hand, underpricing may cause stock outs to lose users who are less price sensitive and would have paid a higher price than offered. On the other hand, overpricing can lead to a waste of resources and lower revenues.

Recently, an auction-style pricing model, called *Spot Instances*, was proposed by Amazon EC2 to bid for unused large capacity [3]. The use of *Spot Instance* in purchasing and consuming cloud service is the first step on a large scale towards "market pricing" for computing resources based on offer and demand. In *Spot Instance*, the *Spot Price* is dynamic and varies depending on supply and demand fluctuations. Cloud users then can specify the instance type, the number of computing instances they want to run, bid on unused Amazon EC2 capacity and run those instances as long as their bids exceed the current *Spot Price*. Compared to *On-Demand Instance*, *Spot Instance* may allow users to lower their Amazon EC2 costs. This purchasing design, however, is still primitive with significant constraints. First, it fails to enforce fair competition among users in facing of resource scarcity and may lead to untruthful bidding and thus unfair resource allocation. This is because *Spot Instance* implicitly assumes an unlimited amount of computing resources such that all bidders are satisfied if their bids exceed the current *Spot Price*. However, when there exists only a limited amount of computing resources, dishonest users are able to abuse the system and obtain (at least) short-term advantages. For example, since *Spot Price* changes once per hour and in many cases less frequently [3], a dishonest user then can deliberately set a large maximum price bid while being charged only at the lower *Spot Price* for a short-term use. Furthermore, *Spot Instance* limits users to bid for each computing instance individually instead of multiple different instances at a time, which is inefficient and may not adequately meet users' overall demands. As a final point on *Spot Instance*, instances being used will be terminated immediately when

*Spot Price* increases above the maximum bid price. As a result, users's applications may be interrupted unexpectedly without notice.

In this paper, we propose to design truthful and computationally efficient auction mechanisms for cloud resource pricing. By truthfulness, we mean that regardless of the declarations of other bidders, any bidder has the incentive to reveal his true valuation of the cloud resources he requests. By efficiency, we mean that the auction mechanism should have a polynomial time complexity. We formulate the cloud service pricing problem as an multi-unit combinatorial auction (MUCA) problem, where there are multiple types of computing instances for sale, and for each instance there are multiple units/copies. We focus on the case where each user/bidder desires a number of units of each instance. It is well-known that the classic auction mechanism–VCG mechanism enforces truthful combinatorial auctions [4]. However, the problem of finding the optimal allocation, *i.e.*, to maximize the sum of the declared valuations, has shown to be NP-hard [5]. Therefore, we resort to greedy allocation mechanisms that are computationally efficient and still achieve a reasonably effective optimization. To ensure truthfulness, we follow a generalized routine from [6] and propose a critical-value based payment scheme, which makes the MUCA a truthful mechanism when coupled with the greedy instance allocation scheme. Prior works in the area of truthful mechanism design usually assume that the bidders do not collude with each other [6], [7], [8], [9], [10]. However, collusive bidders can adopt a non-truthtelling strategy to maximize their total utility. To address this problem, we propose a collusion-resistant cloud resource auction mechanism by using the random rounding technique in [11]. The proposed algorithm uses a randomized posted price strategy that offers bidders a "take it or leave it" price independent of bids of other bidders. We analytically show that truth-telling is an optimal strategy. The main contributions of this paper are:

We identify the general requirements of designing effective auction mechanisms for cloud computing resources. We formulate the cloud pricing problem as an MUCA problem and propose a computationally efficient and fine-grained cloud computing resource auction mechanism with truth-telling as a dominant strategy. The cloud user/bidder can request any combination of computing instances and only pay the minimum it needs to win the auction. We analytically show that the algorithm achieves truthfulness and has a polynomial complexity of $O(nm + n^2)$ when $n$ users compete for $m$ different computing instances with multiple units.

We also propose a collusion-resistant cloud computing resource auction mechanism that has a polynomial complexity $O(nm)$. The proposed scheme can effectively defend against small-size collusions. To our best knowledge, this is the first cloud computing resource auction design that achieves efficiency and collusion resistance. We analytically show that truth-telling is an optimal strategy with probability at least $1-O(t/k_{min})$ when the size of collusion group is less or equal to a threshold $t$ and $k_{min}$ is the minimum of the available units of computing instances.

We conduct extensive simulations to show that, in a competitive cloud resource market, the proposed algorithms can increase the revenue of cloud providers, especially when allocating relatively limited computing instances to a potentially large number of cloud users.

## II. Related Work

Auctions are the quickest and most efficient methods of selling goods at market value. Combinatorial auctions, which have been widely studied in the literature [5], [8], [6], [7], [12], [11], allow the auctioneer to sell multiple different items simultaneously and allow the bidders to bid on any combination of items. As a pioneering work on combinatorial auctions, Vikcrey-Clarke-Groves (VCG) mechanism is optimal for each bidder to reveal his true valuations for maximizing the *social welfare* allocates items to the bidders who value them most. However, in many cases, the problem of finding the optimal allocation, *i.e.*, to maximize the sum of the declared valuations (*i.e.*, *social welfare*), has shown to be NP-hard [5]. To obtain computational efficiency, researchers work on approximation algorithms to design polynomial time truthful auction mechanisms. Unfortunately, the two requirements, *i.e.*, computational efficiency and truthfulness do conflict with each other in many cases [8]. For example, the combination of an approximation algorithm (instead of the exact optimization routine) and the VCG payment scheme causes the mechanism to lose the property of truthfulness. To resolve the conflict, many algorithmic techniques have been proposed, *e.g.*, greedy heuristics together with critical value based payment scheme [6], [8] and randomized rounding of the LP-relaxation [7], [12]. In this paper, we follow the generalized routine from [6] to design truthful and computationally efficient auction mechanisms for cloud computing resources, but focus on a multi-unit combinatorial auction (MUCA) problem where certain of the computing instances may have multiple identical copies. In many case studies of auctions run in practice, collusion is considered as a serious problem [11], [13]. In this paper, we further consider a general case of allocating a bundle of different computing instances among the bidders and utilize the framework from [11] to design a collusion-resistant instance auction mechanism.

Recently, researches on spectrum auctions have received much attention [14], [15], [13]. The use of auctions in spectrum allocation allows the system to achieve both fairness and efficiency. In [14], Zhou *et al.* used the greedy allocation and critical value based pricing approach in [6] and proposed a truthful and computationally efficient spectrum auction mechanism under the bidder interference constraints. In [13], the same authors also proposed a collusion-resistant spectrum scheme using APM from [11]. By first dividing bidders into multiple non-overlapping segments based on the interference constraints, the spectrum auction problem is essentially a single-unit auction problem with an unlimited frequency/channel supply. Different from them, in this paper, we consider a more complex multi-unit combinatorial auction

(MUCA) problem in cloud resource pricing system where each computing instance may have multiple identical copies. Perhaps more importantly, in our pricing model, we focus on the case that each distinct instance for sale only has a limited number of units (*i.e.*, a limited supply), instead of assuming there exists an unlimited supply.

## III. Problem Formulation

### A. The Auction Model

We consider a cloud service auction setting, where a single cloud provider (*e.g.*, Amazon) deliver $m$ types of instances (e.g., Amazon EC2 Instances). These instances are characterized by their sizes of CPU/memory (*i.e.*, small, medium, large and extra large etc.), operating systems (*i.e.*, Linux/UNIX, SUSE Linux and Microsoft Windows Server operating systems etc.) and time limit for instance usage (*i.e.*, the number of hours an instance can run). For each instance type $i$, there are a limited number of units, *i.e.*, $k_i$ units of instance type $i$. We assume there are $n$ users/bidders (we will use users and bidders interchangabely in the following discussion), where each bidder $j$ desires a number of units of each instance. Let $\mathcal{S}_j$ denote the instance "bundle" requested by a bidder. Obviously, $\mathcal{S}_j \in \mathcal{S} = \{0,\ldots,k_1\} \times \{0,\ldots,k_2\} \times \cdots \times \{0,\ldots,k_m\}$.

*Instance request*– In the tuple $(\mathcal{S}_j, \mathcal{B}_j)$, $\mathcal{S}_j = \{k_j^1, k_j^2, \ldots, k_j^m\}$ represents the number of units of each instance desired by user $j$, where $k_j^i$ denotes the number of units of instance type $i$ is desired by user $j$. Let $\overline{\mathcal{S}}_j$ denote the set of instances (bundle) corresponding to $\mathcal{S}_j$. $\mathcal{B}_j = \{b_j^1, b_j^2, \ldots, b_j^m\}$ represents the per-instance bid for different types of instances, where $b_j^i$ denotes the per-instance bid for instance type $i$ of bidder $j$ and $b_j^i = 0$ if instance type $i$ is not desired. Let $\overline{\mathcal{B}}_j = \sum_{i=1}^m k_j^i b_j^i$ denote bidder $j$'s total bid for his desired bundle. Let $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n\}$ and $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_n\}$ denote the instance bundles requested by bidders and the corresponding bid set, respectively.

*Bidder's total valuation*– $\mathcal{V}_j$ represents the maximum amount of money bidder $j$ is willing to pay for his desired bundle. Note that the bidder may choose to lie (*i.e.*, bid less or more). By definition, $\mathcal{V}_j$ is the sum of bidder's valuations for different instances $\mathcal{V}_j = \sum_{i=1}^m k_j^i v_j^i$, where $v_j^i$ denotes bidder $j$'s valuation for instance type $i$. In this paper, we consider the single-minded bidder $j$ [6], who is willing to pay his *valuations* $v_j^i$ and $\mathcal{V}_j$ of the desired bundle only if he receives the whole desired bundle (or any superset); otherwise he is willing to pay 0 for any other bundle, *i.e.*, $\mathcal{V}_j = 0$. We assume that each $\mathcal{V}_j$ is privately known to bidder $j$. Note that $\mathcal{V}_j$ does not depend on the other bidder's bids, and if bidder $j$ truthfully reports his valuation, $b_j^i = v_j^i$ and $\overline{\mathcal{B}}_j = \mathcal{V}_j$.

*Payment*– $\mathcal{P}_j$ represents the amount of money bidder $j$ is required to pay after auction mechanism is run. We assume bidder $j$ is charged 0 if he loses.

*Bidder's utility*– $\mathcal{U}_j$ represents the difference between bidder $j$'s valuation and payment, *i.e.*, $\mathcal{U}_j = \mathcal{V}_j - \mathcal{P}_j$. Rational bidders wish to maximize their utilities. Based on the definitions of valuation and payment, it is easy to see that if bidder $j$ does not receive his desired bundle, $\mathcal{U}_j = 0$. For ease of exposition, we use $x_j \in \{0,1\}$ as an indicator for whether the desired bundle is allocated to bidder $j$ or not and $\mathcal{X} = \{x_1, \ldots, x_n\}$ to indicate the winners of the auction, respectively. Thus, bidder $j$' utility can be written generally as $\mathcal{U}_j = \mathcal{V}_j x_j - \mathcal{P}_j$. Since the number of instance is limited, it is required that $\sum_{j=1}^n k_j^i x_j \le k_i$.

*Revenue*– $\mathcal{R}$ represents the sum of payments made to the auctioneer (*i.e.*, cloud service provider) by all the bidders, *i.e.*, $\mathcal{R} = \sum_{j=1}^n \mathcal{P}_j$.

### B. Definitions

*Definition 1:* (Truthfulness) An auction mechanism is *truthful* if for any bidder $j$, regardless of the declarations (instance requests) of the other bidders, $\mathcal{U}_j$ achieves the maximum when bidder $j$ truthfully bids its true valuation, *i.e.*, $b_j^i = v_j^i$ and $\overline{\mathcal{B}}_j = \mathcal{V}_j$. In other words, truthfully revealing his valuation of the desired bundle is a *dominant strategy* for each bidder.

*Definition 2:* ($(t,p)$-truthfulness) An auction mechanism is $(t,p)$-truthful if no coalition of size $t$ or fewer can increase their total utilities by using a non-truthtelling strategy with probability $p$ or higher, where $p$ is a function of some input parameters of the auction mechanism.

*Definition 3:* (Critical value) A *critical value* $v_c$ is the minimum value that bidder $j$ should bid in order to win $\overline{\mathcal{S}}_j$. Formally, $\forall \overline{\mathcal{B}}_j$, if $\overline{\mathcal{B}}_j > v_c$ then bidder $j$ wins; $\forall \overline{\mathcal{B}}_j$, if $\overline{\mathcal{B}}_j < v_c$ then bidder $j$ loses. A *critical bidder* of bidder $j$ is the bidder whose bid is the first bid following bidder $j$ that has been denied but would have been grated were it not for the presence of $j$.

*Definition 4:* (Monotonicity) An auction (allocation) mechanism is *monotone* if, increasing the bid and/or decreasing the size of the desired bundle will not cause a bidder to lose while fixing the bids from all the other bidders, denoted by $\overline{\mathcal{B}}_{-j}$. Formally, $\forall$ bidder $j$ and $\forall$ $\overline{\mathcal{B}}_{-j}$, if $\overline{\mathcal{B}}_j$ is a winning bid then $\overline{\mathcal{B}}_j' \ge \overline{\mathcal{B}}_j$ and/or $\overline{\mathcal{S}}_j' \subseteq \overline{\mathcal{S}}_j$ also makes a winning bid.

Definition 1 is used to characterize the truth-revealing property of Algorithm 1 and Definition 2 is used to characterize the truth-revealing property of Algorithm 2 when user collusion may exist. Definitions 3 and 4 are important properties for constructing truthful auction mechanisms.

## IV. The Proposed Approach: An Efficient and Truthful Mechanism for Cloud Computing Resource Auction

In this section, we present a computationally efficient truthful mechanism for cloud computing resource auction. The proposed auction mechanism consists of two schemes: greedy resource allocation scheme and payment scheme. We show that combining these two schemes together can achieve truthfulness with polynomial computational complexity.

### A. Resource Allocation

In a full-fledged cloud computing environment, service providers are envisioned to provide various different configurations of computing resources to a large number of heterogeneous cloud users. As mentioned earlier, polynomial-time

**Algorithm 1** An computationally efficient and truthful mechanism for cloud service auction

```
Input: $\mathcal{S}, \mathcal{B}, \mathcal{W}$
Output: $\mathcal{X}, \mathcal{P}$
 1: Calculate $\overline{\mathcal{B}}$ based on $\mathcal{B}$
 2: Sort bids in $\overline{\mathcal{B}}$ such that $\frac{\overline{\mathcal{B}}_1}{\sqrt{\sum_{i=1}^m k_1^i w_i}} \geq \dots \geq \frac{\overline{\mathcal{B}}_n}{\sqrt{\sum_{i=1}^m k_n^i w_i}}$
 3: Initialize $x_1 = \cdots = x_n = 0$  %greedy allocation
 4: for $j = 1, \dots, n$ do
 5:   if $\forall q \in \{1, \dots, m\}, \sum_{t=1}^{j-1} k_t^q x_t + k_j^q \leq k_q$ then
 6:     $x_j = 1$
 7:   end if
 8: end for
 9: Initialize $\mathcal{P}_1 = \mathcal{P}_2 = \dots = \mathcal{P}_n = 0$  %payment
10: for $j = 1, \dots, n$ do
11:   if $x_j = 1$ then
12:     $\overline{\text{INS}_1} = \sum_{t=1}^{j-1} k_t^1 x_t, \dots, \overline{\text{INS}_m} = \sum_{t=1}^{j-1} k_t^m x_t$
13:     for $s = j+1, \dots, n$ do
14:       if $\forall q \in \{1, \dots, m\}, \overline{\text{INS}_q} + k_s^q \leq k_q$ then
15:         $\overline{\text{INS}_1} = \overline{\text{INS}_1} + k_s^1, \dots, \overline{\text{INS}_m} = \overline{\text{INS}_m} + k_s^m$
16:         if for some $q \in \{1, \dots, m\}$, $\overline{\text{INS}_q} + k_j^q > k_q$ then
17:           $\mathcal{P}_j = \frac{\overline{\mathcal{B}}_s \sqrt{\sum_{i=1}^m k_j^i w_i}}{\sqrt{\sum_{i=1}^m k_s^i w_i}}$
18:           break;
19:         end if
20:       end if
21:     end for
22:   end if
23: end for
24: return $\mathcal{X}$, $\mathcal{P}$
```

algorithms do not exist for computing the optimal solution in MUCA, we thus propose a greedy resource allocation scheme, which achieves reasonable economic- and computational-efficiency.

The allocation scheme is shown in the first part of Algorithm 1. The cloud provider/auctioneer first computes $\overline{\mathcal{B}} = \{\overline{\mathcal{B}}_1, \dots, \overline{\mathcal{B}}_n\}$ based on $\mathcal{B}$ in the instance request, where $\overline{\mathcal{B}}_j = \sum_{i=1}^m k_j^i b_j^i$. Then it ranks the bids in descending order based a ranking metric called *average price per weighted instance*:

$$\frac{\overline{\mathcal{B}}_j}{\sqrt{\sum_{i=1}^m k_j^i w_i}}, \quad \text{for } j \in 1, \dots, n.$$

Note that we abuse the notation and assume $\overline{\mathcal{B}}_1/\sqrt{\sum_{i=1}^m k_1^i w_i} \geq \dots \geq \overline{\mathcal{B}}_n/\sqrt{\sum_{i=1}^m k_n^i w_i}$. This criterion takes into account the amount of bid value $\overline{\mathcal{B}}_j$, the number of instances requested $k_j^i$ and the weight of the instance $w_i \in \mathbb{R}^+$. The reason that we introduce instance weight is to distinguish between different instances, *e.g.*, the bidder should bid more for an instance with high CPU speed or large memory. Based on the sorted list, the auctioneer chooses the most promising bid and continues to find the next most promising bid in the list that satisfies the constraint while not affecting the allocated ones. Finally, it examines the following bids in order and grants it if it does not conflict with the updated instance constraint, and denies it otherwise.

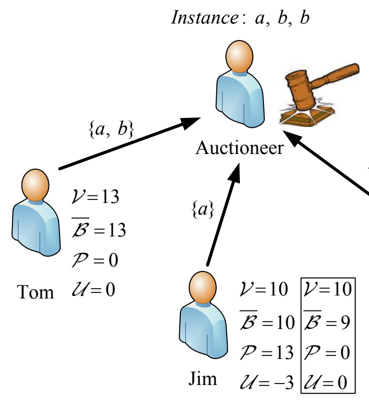

Fig. 1. An example of multi-unit combinatorial auction (MUCA) shows that the direct combination of greedy instance allocation and the classic Clarke's payment scheme does not make a truthful auction mechanism.

The greedy approach works well and is very speedy. It is well known that the classical Clarke's payment scheme coupled with the optimal allocation scheme enforces truthfulness. However, the greedy allocation together with the Clarke's payment scheme makes the pricing mechanism untruthful [6]. Consider an example in Fig. 1, assume there are two different instances $a$ and $b$ and three users/bidders Tom, Jim and Bob. The instances $b$ has two units available. If the three users bid truthfully, Tom will bid 13 for $\{a, b\}$, Jim will bid 10 for $a$ and Bob will bid 8 for $\{b\}$. For simplicity, we assume instances $a$ and $b$ have the same weight. Based on the ranking metric, the greedy instance allocation scheme in Algorithm 1 will satisfy Jim and Bob's requests and deny Tom's request. For this allocation, we compute their payments using Clarke's payment scheme, *i.e.*, the payment of bidder $j$ is computed by $\mathcal{P}_j = \sum_{i \neq j} \overline{\mathcal{B}}_i(\mathcal{A}^*_{\sim j}) - \sum_{i \neq j} \overline{\mathcal{B}}_i(\mathcal{A}^*)$, where $\mathcal{A}^*_{\sim j}$ is an allocation that maximizes the sum of the bids/valuations if bidder $j$ had not participate in the auction (*i.e.*, $\max_{\mathcal{A}} \sum_{i \neq j} \overline{\mathcal{B}}_i(\mathcal{A})$), and $A^*$ is the the proposed allocation. Note that in VCG pricing mechanism, $\mathcal{A}^*$ is the optimal allocation while it is the greedy allocation in our proposed pricing mechanism (refer to [6] for the detailed procedures). The results are shown in Fig. 1. As we can see, Jim should pay 13 for $a$, which is more than the amount he declared. The resulting utility is thus -3. To maximize his utility, Jim would have been better off lying, *e.g.*, bidding 9 for $a$. Now the greedy instance allocation scheme would have satisfied Tom and Bob's requests and denied Jim's request. By bidding untruthfully, Jim can increase his utility from -3 to 0. This simple example clearly shows that the use of Clarke's payment scheme with a greedy instance allocation cannot make truthfelling a dominant strategy in MUCA.

## *B. Payment Scheme*

Before presenting the payment scheme, we first introduce a notion of *critical value*. As defined in Definition 3, a *critical value* $v_c$ is the minimum value that bidder $j$ should bid in order to win $\overline{\mathcal{S}}_j$. In fact, $\forall \overline{\mathcal{B}}_j$, if $\overline{\mathcal{B}}_j > v_c$ then $j$ wins; $\forall \overline{\mathcal{B}}_j$, if $\overline{\mathcal{B}}_j < v_c$ then $j$ loses.

Our general payment rule is defined as follows:

1) A bidder whose bid is denied pays 0;
2) A bidder who receives the desired bundle pays the *critical value*;
3) A bidder who does not have a *critical bidder* pays 0.

The payment rule is closely related to the allocation scheme. In our design, finding the *critical value* of a bidder $j$ is equivalent to finding the *critical bidder* of bidder $j$, whose bid is the first bid following bidder $j$ that has been denied but would have been granted were it not for the presence of $j$. However, if a bidder does not have a *critical bidder*, he will pay zero, *e.g.*, in the cases that the bidders who have ranks after him are all allocated or he appears at the bottom of the sorted list. The proposed greedy payment scheme is shown in the second part of Algorithm 1. In order to find the *critical bidder* of each granted bidder $j$, we assume $j$'s bid is not in the sorted bid list. Then we identify the first bidder in the remaining bid set that if he is granted (*i.e.*, satisfy the resource constraints of each instance) it will also make the instance resource unavailable for bidder $j$. By definition, this bidder (we denote it as $s$) is indeed the *critical bidder* of $j$. We can compute the *critical value* (*i.e.*, payment) of $j$ as

$$\mathcal{P}_j = \frac{\overline{\mathcal{B}}_s\sqrt{\sum_{i=1}^m k_j^i w_i}}{\sqrt{\sum_{i=1}^m k_s^i w_i}}.$$

In the analysis section, we show that, if the instance allocation scheme satisfy certain properties we can always find critical values for the granted bids.

Now we reconsider the example in Fig. 1. We have seen that if all bidders truthfully reveal their bids/valuations, Jim and Bob's bids are granted and Tom's bid is denied. We still assume instance $a$ and $b$ has the same weight. Under the critical value based payment scheme, the critical bid associated with Jim is Tom's bid. Therefore, Jim pays $\frac{13}{\sqrt{2}} \approx 9.19$ and his resulting utility is $10 - 9.19 = 0.81$. If Jim under-bids at 9, his bid is denied and the resulting utility is 0. Notice that by paying the critical value, Jim has no incentive to bid untruthfully.

## *C. Analysis*

We first show that in Algorithm 1 truthtelling strategy is a dominant strategy for all bidders.

**Theorem** *1:* Algorithm 1 is *truthful.*

*Proof Sketch*: Due to space limitations, we only provide a sketch of the proof here using Theorem 9.6 in [6]. The details of the proof are in the full version [16]. We start by showing that the proposed auction mechanism satisfies the following properties: 1) *monotonicity* property; 2) *critical* payment rule; 3) *exactness* and 4) *participation*.

We first examines the *monotonicity* property. The ranking metric we adopt in the instance allocation scheme is $\overline{\mathcal{B}}_j/\sqrt{\sum_{i=1}^m k_j^i w_i}$. As we can see, for a fixed instance weight setting, this metric is monotonic in $\overline{\mathcal{B}}_j$ and anti-monotonic in $k_j^i$. Therefore, increasing the bid and/or decreasing the size of "current bundle" can only increase the bid's ranking in the sorted $\overline{\mathcal{B}}$. Due to the greedy allocation, this will not cause a bidder to lose, making it easier to win the auction than before given any fixed settings of other bids.

We next examines the property of *critical* payment. Recall that in our allocation scheme, a bidder either gets exactly the bundle (*i.e.*, desired set of instances) or nothing, *i.e.*, he never gets only part of what he requested. This property is called *exactness* [6]. By Lemma 9.1 in [6], there always exists a *critical value* $v_c$ for each bidder. Now let us verify if the payment $\mathcal{P}_j$ of a satisfied bidder equals $v_c$. Consider the process of finding the *critical bidder* of a satisfied bidder $j$ in the greedy payment scheme. The *critical* bidder is identified if and only if step 14 and 16 both hold, and if this is true, execution of **for** loop is terminated. That implies there exist enough instances to satisfy bidder $j$ before the execution of this loop. That is, bidder $j$ would have received his desired bundle if he ranks above the critical bidder $s$ and he loses the bid otherwise. Clearly, if bidder $j$ submits a bid above the payment value $\overline{\mathcal{B}}_j > \mathcal{P}_j = \overline{\mathcal{B}}_s\sqrt{\sum_{i=1}^m k_j^i w_i}/\sqrt{\sum_{i=1}^m k_s^i w_i}$, then

$$\frac{\overline{\mathcal{B}}_j}{\sqrt{\sum_{i=1}^m k_j^i w_i}} > \frac{\overline{\mathcal{B}}_s\sqrt{\sum_{i=1}^m k_j^i w_i}}{\sqrt{\sum_{i=1}^m k_s^i w_i}}\frac{1}{\sqrt{\sum_{i=1}^m k_j^i w_i}} = \frac{\overline{\mathcal{B}}_s}{\sqrt{\sum_{i=1}^m k_s^i w_i}}.$$

We can conclude that, the payment value $\mathcal{P}_j$ of a satisfied bidder $j$ is exactly the minimum value needed for winning, *i.e.*, *the critical value*. Notice that an unsatisfied bidder pays 0 in our auction scheme. This property is called *participation*. By Theorem 9.6 in [6], the cloud instance auction mechanism in Algorithm 1 satisfies exactness, monotonicity, participation and critical, and thus is truthful. ■

**Theorem** *2:* Algorithm 1 with criterion $\overline{\mathcal{B}}_j/\sqrt{\sum_{i=1}^m k_j^i w_i}$ approximates the optimal solution with a factor of $\sqrt{\frac{w_{max}}{w_{min}}\sum_{i=1}^m k_i}$.

*Proof:* Here, the optimal solution is the one that maximizes the sum of the declared valuations, *i.e.*, *social welfare*. Due to space limitations, the detailed proof is provided in the full version [16]. ■

We next analyze the computational complexity of Algorithm 1. In the initialization process, it takes time of the order of $O(n)$ to compute the ranking metric. In the greedy allocation scheme, it takes time of the order of $O(n\log n)$ to sort the bids and time of of the order of $O(nm)$ to allocate the instances. In the greedy payment scheme, it takes time of the order of $O(nm+n^2)$ to compute the payments for all bidders. Thus, the overall complexity of Algorithm 1 is $O(nm+n^2)$.

**Algorithm 2** A collusion-resistant mechanism for cloud service auction

```
Input: $\mathcal{S}, \mathcal{B}, \mathcal{W}$
Output: $\mathcal{X}, \mathcal{P}$
1: Calculate $\widehat{\mathcal{B}} = \{\widehat{\mathcal{B}}_1, \ldots, \widehat{\mathcal{B}}_m\}$ based on $\mathcal{B}$, where $\widehat{\mathcal{B}}_i = \{b_1^i, \cdots, b_n^i\}$
2: Calculate $\widehat{\mathcal{S}} = \{\widehat{\mathcal{S}}_1, \ldots, \widehat{\mathcal{S}}_m\}$ based on $\mathcal{S}$, where $\widehat{\mathcal{S}}_i = \{k_1^i, \ldots, k_n^i\}$
3: for $i = 1, \ldots, m$ do
4:     Run AEM*($k_i, \widehat{\mathcal{B}}_i, \widehat{\mathcal{S}}_i$)  %auction for each instance
5:     Record $\mathcal{X}'_i = \{x'_{i,1}, \ldots, x'_{i,n}\}$ and $p_i$
6: end for
7: for $j = 1, \ldots, n$ do
8:     if $\forall q \in \{1, \ldots, m\}, x'_{i,1} = 1$ then
9:         $x_j = 1$
10:        $\mathcal{P}_j = \sum_{i=1}^{m} k_j^i p_i$
11:    else
12:        $x_j = 0$
13:        $\mathcal{P}_j = 0$
14:    end if
15: end for
16: return $\mathcal{X}$, $\mathcal{P}$
```

## V. The Proposed Approach: A Collusion-resistant Mechanism for Cloud Computing Resource Auction

In many case studies of auctions run in practice, collusion is considered as a serious problem [11], [13]. In this section, we propose a collusion-resistant mechanism for cloud computing resource auction and show that it can achieve $(t, p)$-truthfulness.

### A. Collusion-resistant Instance Allocation and Payment

We first consider an example to show that bidders can form coalitions and exchange side-payments to each other so as to increase the total utility of the bidders in the coalition. In Fig. 1, we have seen that Tom's bid is used to compute the critical value/payment of Jim. If Tom and Jim form a coalition group, they try to increase their total utility by colluding and exchange side-payments with each other. For this purpose, Tom under-bids at 12 without affecting the allocation (*i.e.*, winners and losers) and increasing his utility (still 0). However, Jim is now charged by a lower critical value, *i.e.*, $\frac{12}{\sqrt{2}} \approx 8.49$. The total utility of this coalition group increase from $0.82$ to $1.51$. Obviously, Jim will give Tom a side-payment between $0$ and $0.69$ in order to induce him to join the coalition.

In practice, a group of bidders may maximize the total utility of the coalition group with a non-truthtelling strategy. In this section, we design a collusion-resistant auction mechanism for cloud service auction. Particularly, we focus on a special case of MUCA problem, where each bidder $j (j \in \{1, \ldots, n\})$ requests only one unit of a cloud instance, *i.e.*, $k_j^1 = k_j^2 = \ldots = k_j^m = 1$. We propose to decompose this MUCA problem into multiple $k$-unit single-instance auction problems, where each type of instance is auctioned independently among the bidders. Then in each $k$-unit single-instance auction, we use the AEM auction in [11], where a randomized "take it or leave it" price is posted for interested bidders. The bidders bidding no less than this price are considered as potential winners. In the final step, we greedily choose the final winners that are satisfied by all the auctions they participate.

The proposed auction mechanism is shown in Algorithm 2. Based on the instance request, the auctioneer extracts the bid set for each instance. It then divides bidders into $m$ groups and conducts auction for $m$ instances one by one(Note that since each instance is independently auctioned, the instance weight can be eliminated here). For each group, we use a modified version of AEM [11], which can achieve collusion resistance to small-size coalition group. For each instance $i$ with $k_i$ limited supply, AEM first uses random sampling to estimate the $k_i$-th highest bid value among the interested bidders. The auctioneer samples each bidder in the bidder set with probability $\frac{1}{k_i}$ and chooses $p$ as the highest bid in the sampling set. In the second step, it estimates the number of bids above $p$ using a *consensus function* $n_p = \lceil \#_p(\widehat{\mathcal{B}}_i) \rceil_{\{2^{i+y}: i \in \mathbb{Z}\}}$, where $y$ is uniformly chosen from $[0, 1]$. In the final step, AEM uses a posted price mechanism while ensuring that the instance is not oversold with high probability. Specifically, if $n_p \leq k_i$ it offers the common price $p$ for all bidders; Otherwise, for each bidder $j$ it either offers a price $z_j = p$ independently with probability $\frac{k_i}{2n_p}$ or does not sell the instance at all with the remaining probability, *i.e.*, offer a price that equals infinity. If the instance is oversold when $n_p > k_i$, AEM uses the classic $k_i$-Vickrey auction [17] to sell the instances to the highest $k_i$ bidders at the $k{+}1$-st highest bid value. As we can see, the random sampling and the *consensus estimation* indeed introduce randomness to the process of price/payment determination, which makes it difficult for the colluding bidders to maximizing their total utility by harvesting the bid changes. It is also clear to see that, no matter what case it is, a common "take it or leave it" price is posted for all interested bidders for one particular instance. Based on AEM, we propose AEM* and focus on the case where a bidder $j$ desires at most $k_j^i = 1$ units of instance $i$. Since we focus on single-minded bidders in this paper, we consider bidder $j$ as a potential winner only if he is satisfied. After step 3, the AEM* examines whether each bidder is satisfied with $k_j^i (k_j^i \in \{0, 1\})$ unit. The output is an indicator vector $\mathcal{X}'_i = \{x'_{i,1}, \ldots, x'_{i,n}\}$, where $x'_{i,j} = 1$ indicates that bidder $j$ is satisfied and $x'_{i,j} = 0$ otherwise. Finally, the auctioneer checks whether each bidder $j$ is totally satisfied for all instances he desired or not. If yes, it outputs $x_j = 1$ and $\mathcal{P}_j = \sum_{i=1}^{m} k_j^i p_i$; otherwise, it outputs $x_j = 0$ and $\mathcal{P}_j = 0$.

### B. Analysis

We now show that the proposed auction mechanism is $(t, p)$-truthful. We have the following theorem:

**Theorem** *3:* Algorithm 2 is $(t, p)$-truthful with probability $p = 1 - O(t/k_{min})$ when $t/k_{min} \ll 1$, where $t$ is the size of coalition group and $k_{min} = \min\{k_1, \ldots, k_m\}$.

*Proof:* When there exists one single instance with $k$-unit limited supply for auction. Lemma 15 in [11] shows that AEM is $t$-truthful with probability $P = 1 - O(t/k)$. In our auction problem, we have $m$ instances for auction, each of which has $k_i$-unit limited supply. In Algorithm 2, because the bidders for each instance auction are grouped based on their preferences instead of the bid values, the $t$-truthfulness is achieved within the auction group of each instance due to the use of AEM. Consider any coalition group of size less or equal to $t$ across multiple auction groups. Assume $t_i$ is the size of the coalition group in the instance auction group $i$ and $\sum_{i=1}^{m} t_i \leq t$. Because each auction group is $t_i$-truthful with probability $P_i = 1 - O(t_i/k_i)$ and the auction on each instance is conducted independently, the collusion across multiple auction groups cannot manipulate their bids to maximize their total utility of the coalition group. Therefore, Algorithm 2 is $t$-truthful. We next bound the probability. Let $\widetilde{P}_i = 1 - P_i$. When $t/k_{min}$ is very small, the probability that Algorithm 2 is $t$-truthful is

$$\begin{aligned} P \geq \prod_{i=1}^{m} P_i &= \prod_{i=1}^{m} (1 - \widetilde{P}_i) \\ &\geq 1 - \sum_{i=1}^{m} \widetilde{P}_i \end{aligned}$$

Since $\widetilde{P}_i = O(t_i/k_i)$, we have $\widetilde{P}_i = O(t_i/k_{min})$. Therefore, $\sum_{i=1}^{m} \widetilde{P}_i = O(\sum_{i=1}^{m} t_i/k_{min})$. Note that $\sum_{i=1}^{m} t_i \leq t$, thus $\sum_{i=1}^{m} \widetilde{P}_i = O(t/k_{min})$. When $t/k_{min} \ll 1$, we then have

$$P \geq p = 1 - \sum_{i=1}^{m} \widetilde{P}_i = 1 - O(t/k_{min}).$$

By Definition 2, Algorithm 2 is $(t,p)$-truthful with probability $p = 1 - O(t/k_{min})$. ■

We next analyze the computational complexity of Algorithm 2. The AEM* takes time of the order of $O(n)$ to auction each instance. Since there are $m$ instances, the overall complexity of instance auction is $O(nm)$. Then it takes time of the order of $O(n)$ to determine the final winners. Thus, the overall complexity of Algorithm 2 is $O(nm)$.

### C. Discussion

It is worth noting that, in Algorithm 1, we assume each bidder desires at least one instance (otherwise he has no incentive to bid) and the number of units of each instance is limited. Under this condition, if the *instance request* $\{\mathcal{S}_j, \mathcal{B}_j\}$ is uniformly generated, the *average revenue* would increases linearly as the number of instances increases. This is because more different instances will be allocated to winning bidders. However, when the number of units $k_i$ is relatively large compared to the number of bidders/users, the resulting revenue will decrease since the increase of pool of winning bidders makes it hard to find their corresponding *critical bidders*. In the extreme case, the revenue will go to zero. This is equivalent to case when there exists an unlimited number of instances and each bidder will get allocated. We argue that, for generating a high revenue, Algorithm 1 suits best for application scenarios where there exists a relatively large cloud users compared to the number of computing instances.

In Algorithm 2, we concentrate on a special case where each bidder requests only one unit of each instance. Obviously, the algorithm can be easily modified to apply to the scenario where each bidder requests one unit of an instance from a particular instance set he is interested instead of all. Another point to note is that, the revenue of Algorithm 2 is much less than that of Algorithm 1. This is mainly due to the fact that the payment schemes of the two algorithms are quite different. We argue that such revenue loss is the price to pay to achieve collusion resistance.

## VI. Performance Evaluation

In this section, extensive simulations are conducted to evaluate the performance of the two proposed cloud instance auction mechanisms in terms of *revenue* and *user satisfactory*. As defined above, *Revenue* is the sum of payments of all winners. *User satisfactory* has been considered as an important performance metric to measure the user experience for applications based on auction-style pricing [13]. It is usually defined as the ratio of the number of winning bidders to the total number of bidders, *i.e.*, the percentage of bidders who allocated the desired bundles.

We assume each bidder's valuation per type of instance is uniformly distributed over $(0, 1]$. When generating the valuations, we also take into account the effect of *weights* on different instances and choose uniformly random $w_i$ over $(0, 1]$. In Algorithm 1, we assume each bidder's desired number of units of each instance $k_j^i$ is uniformly distributed over $[0, r_{max}]$, where $r_{max}$ is the maximum number of units required by each bidder. In Algorithm 2, we introduce a tuning factor $t_f$ to let the users have more flexibility to control their bid requests. The tuning factor is defined as the *acceptable* percentage of $m$ instances allocated to the bidders. When $t_f = 1$, we say the bid request is *strict*, *i.e.*, a bidder is satisfied and will pay for the allocated instances if and only if he receives all $m$ different instances. Without loss of generality, we also assume the number of available units $k_i (i \in \{1, \ldots, m\})$ is same for all instances, *i.e.*, $k_1 = \cdots = k_m$. For each simulation setting, we calculate the results averaged over $1000 \sim 5000$ rounds.

### A. Evaluation of Algorithm 1

In Fig. 2, we plot the revenue and user satisfaction for the proposed auction mechanisms under different $r_{max}$, $k_i$ and $t_f$. Fig. 2(a) shows that the revenue increases almost linearly as the number of available units per instance $k_i$ increases until an inflection point is reached. After the inflection point, the revenue drops quickly to zero. This is due to the use of *critical value* based payment scheme. As $k_i$ increases, more and more users are satisfied, *i.e.*, getting their desired bundle allocated. Meanwhile, it is also more and more difficult to find the *critical bidders* of these winners, which results less payments after $k_i$ reaches the threshold. In the extreme case, the revenue

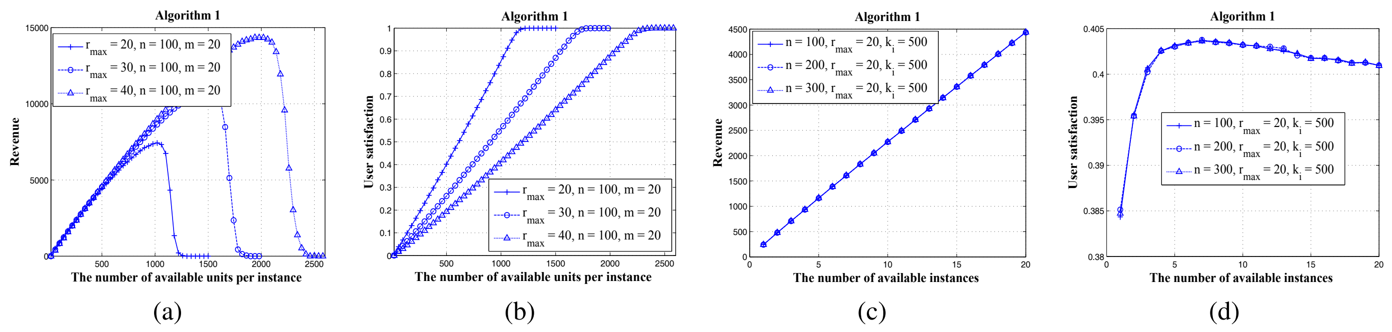

Fig. 2. The effect of $r_{max}$, $k_i$ and $t_f$ on the performance of the proposed auction mechanisms for Algorithm 1.

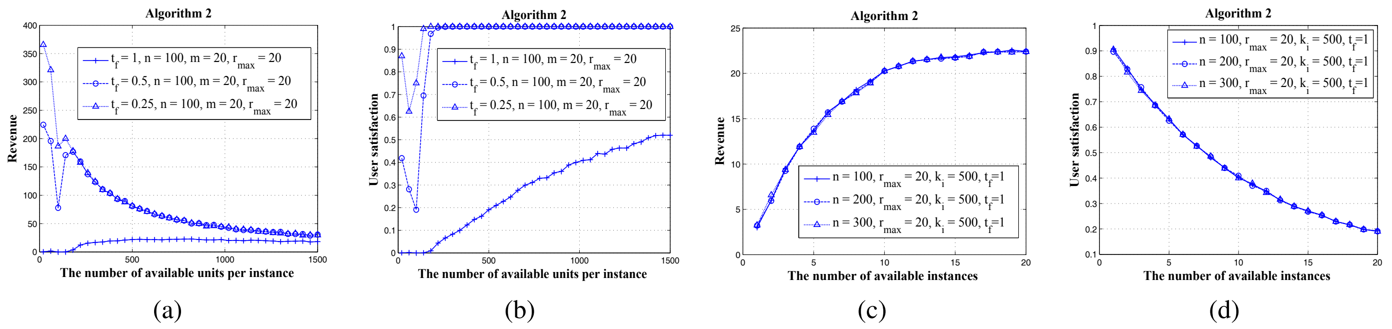

Fig. 3. The effect of $r_{max}$, $k_i$ and $t_f$ on the performance of the proposed auction mechanisms for Algorithm 2.

becomes zero when all bidders are satisfied. This phenomenon can be clearly seen in Fig. 2(b), the user satisfaction is one after reaching certain value of $k_i$. We can also see that if we vary $r_{max}$, the inflection points changes and a larger $r_{max}$ is corresponding to a larger threshold value. In Fig. 2(c) and (d), we fix $k_i = 500$ (*i.e.*, a not fully satisfied case) and study the effect of $m$ on revenue generation and user satisfaction. As we can see, the revenue increases linearly as the number of available instance $m$ increases. However, the user satisfaction will stop increasing much and maintain around a certain level when $m$ further increases. This is because although the winners pay for more different instances allocated to them, the pool of the winners will not change much under fixed $k_i$ and $m$. Due to the same reason, the further increase of $n$ would not help to increase the revenue and user satisfaction level. To show the relationship between the two important performance metrics, Fig. 4 plots the revenue against user satisfaction for Algorithm 1. The simulation is conducted by varying $k_i$ from 20 to 2500. As we can see, there exists a very nice and high level of user satisfaction (*e.g.*, around 0.9 in this setting) at which the maximum revenue can be achieved. Under this satisfaction level, the increase of number of bidders will lead to more revenues.

## B. Evaluation of Algorithm 2

In Fig. 3(a) and (b), for Algorithm 2 we plot the revenue and user satisfaction as $k_i$ changes under different $t_f$. It is shown that when $t_f$ is small, a larger revenue can be generated. This is due to the fact that bidders can be satisfied with only a fraction of computing instances instead of all (*i.e.*, $t_f = 1$). As $k_i$ increases, although it is more likely that more bidders become winners, the decrease of the sampling probability $1/k_i$ may cause the posted price $p$ to be very small. This directly leads to the decrease of total revenue. Note that for $t_f < 1$ a sudden revenue drop occurs when $k_i = n = 100$. This value of $k_i$ is exactly the point at which all bidders will be satisfied if their bids were higher than $p$ when $t_f = 1$. As we can see, when $t_f = 1$ the further increase of $k_i$ will have little effect on the revenue since the instances are abundant for allocation. Therefore, the revenue generation becomes stable afterwards. In Fig. 3(b), it is shown that when $t_f = 1$ it is hard to get a high satisfaction level since the probability of wining $m$ auctions for a bidder is very small. In comparison, by setting $t_f$ to a small value, user satisfaction can quickly reach one when there exist enough units of each instance. We next study the effect of $m$ on revenue and user satisfaction. As shown in Fig. 3(c), although a winner can obtain more instances such that the total revenue is increased, the pool of winners will shrink as $m$ increases. This result can be clearly seen from Fig. 3(d). Similarly to Algorithm 1, the increase of $n$ almost has no effect on the performance when $m$ and $k_i$ are fixed. This is also due to the fact that the pool of winners would not change in this case. In Fig. 5, we plot the revenue against user satisfaction for Algorithm 2. We see that there also exist an optimal user satisfaction level at which maximum revenue can be obtained. However, compared to Algorithm 1, both the revenue and the user satisfaction level are lower. We claim that this is the price to pay to achieve collusion resistance.

## C. Discussion

Based on the above results, we see that the cloud provider could choose a reasonable number of units per instance ($k_i$)

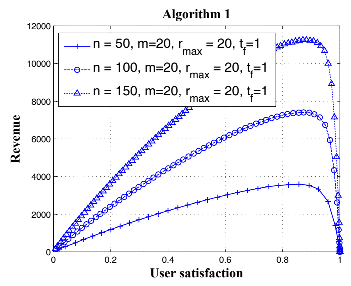

Fig. 4. Revenue V.S. user satisfaction for Algorithm 1.

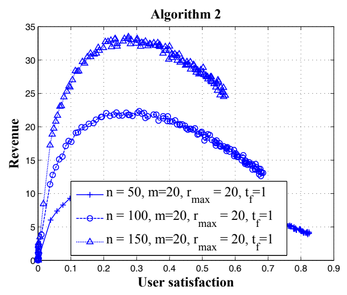

Fig. 5. Revenue V.S. user satisfaction for Algorithm 2.

in light of a rough estimation of users' requests in order to generate a high revenue. Compared to *Spot Instance*, our proposed auction mechanisms focus on application scenarios where computing resources are relatively limited when facing a large number of heterogeneous cloud users. In addition, our mechanisms can achieve *truthfulness* without collusion or $(t, p)$-truthfulness tolerating a collusion group of size $t$ with probability at least $p$. This significant difference comes from the fact that our charging scheme is totally different from that of *Spot Instance*. While *Spot Instance* posts a price for users to decide bid or not, our auction mechanisms determine the payments after all users put their bids.

## VII. Concluding Remarks and Future Work

In this paper, we formulated and investigated the problem of cloud resource pricing. We proposed a suite of computationally efficient and truthful auction-style pricing mechanisms, which enable users to fairly compete for resources and cloud providers to increase their overall revenue. We analytically showed that the proposed algorithms can achieve truthfulness without collusion or $(t, p)$-truthfulness tolerating a collusion group of size $t$ with probability at least $p$. We also showed that the two proposed algorithms have polynomial complexity. Experiments show that, in a competitive cloud resource market, the proposed mechanisms can increase the revenue of cloud providers when allocating relatively limited computing resources to a potentially large number of cloud users.

Several important open problems remain for future research. First, in the first truthful auction mechanism, we assume that the number of available units per instance is limited compared to the large number of users. It is important to explore the extreme case where an unlimited computing resources/instances are available to the cloud users. We still aim to achieve both efficiency and truthfulness for the proposed mechanisms. Second, in the $(t, p)$-truthful auction mechanism with collusion resistance, we assume each user requests only one unit of each computing instance. In practice, it is more desirable to explore the general problem of allocating an arbitrary "bundle" of instances to the bidders other than a unit of each. However, to design a $(t, p)$-truthful algorithm in this case is much more complex. We also plan to identify other advanced techniques other than "random rounding" to obtain good collusion resistance while generating a higher revenue.

## Acknowledgment

This work was partially supported by the US National Science Foundation under grants CNS-1054317 and CNS-1116939.

## References

[1] D. Parkhill, "The challenge of the computer utility," Addison-Wesley Educational Publishers Inc., US, 1966.

[2] P. Mell and T. Grance, "Draft nist working definition of cloud computing," Referenced on June. 3rd, 2009 Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2009.

[3] Amazon.com, "Amazon elastic compute cloud," Online at http://aws.amazon.com/ec2/, 2009.

[4] H. R. Varian, "Economic mechanism design for computerized agents," in *Proc. of USENIX'95 WS on EC*, Berkeley, CA, USA, 1995, pp. 2–2.

[5] M. H. Rothkopf, A. Pekec, and R. M. Harstad, "Computationally manageable combinational auctions," *Manage. Sci.*, vol. 44, pp. 1131–1147, August 1998.

[6] D. Lehmann, L. I. Oćallaghan, and Y. Shoham, "Truth revelation in approximately efficient combinatorial auctions," *J. ACM*, vol. 49, pp. 577–602, September 2002.

[7] A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos, "An approximate truthful mechanism for combinatorial auctions with single parameter agents," in *Proc. of SODA'03*, Philly, PA, USA, 2003, pp. 205–214.

[8] A. Mu'alem and N. Nisan, "Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract," in *Proc. of AI'02*, Menlo Park, CA, USA, 2002, pp. 379–384.

[9] Y. Bartal, R. Gonen, and N. Nisan, "Incentive compatible multi unit combinatorial auctions," in *Proc. of TARK'03*, New York, NY, USA, 2003, pp. 72–87.

[10] R. Gonen and D. Lehmann, "Optimal solutions for multi-unit combinatorial auctions: branch and bound heuristics," in *Proc. of EC'00*, New York, NY, USA, 2000, pp. 13–20.

[11] A. V. Goldberg and J. D. Hartline, "Collusion-resistant mechanisms for single-parameter agents," in *Proc. of SODA'05*, Philadelphia, PA, USA, 2005, pp. 620–629.

[12] R. Lavi and C. Swamy, "Truthful and near-optimal mechanism design via linear programming," in *Proc. of FOCS'05*, Washington, DC, USA, 2005, pp. 595–604.

[13] X. Zhou and H. Zheng, "Breaking bidder collusion in large-scale spectrum auctions," in *Proc. of MobiHoc'10*, New York, NY, USA, 2010, pp. 121–130.

[14] X. Zhou, S. Gandhi, S. Suri, and H. Zheng, "ebay in the sky: strategy-proof wireless spectrum auctions," in *Proc. of Mobicom'08*, New York, NY, USA, 2008, pp. 2–13.

[15] J. Jia, Q. Zhang, Q. Zhang, and M. Liu, "Revenue generation for truthful spectrum auction in dynamic spectrum access," in *Proc. of MobiHoc'09*, New York, NY, USA, 2009, pp. 3–12.

[16] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing," Illinois Institute of Technology, Chicago, IL, Tech. Rep., December 2011, http://www.ece.iit.edu/∼qian/TR_ICNP11.pdf.

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