A Mechanism Design and Learning Approach for Revenue Maximization on Cloud Dynamic Spot Markets

Asterios Tsiourvas*, Constantinos Bitsakos†, Ioannis Konstantinou‡, Dimitris Fotakis†, Nectarios Koziris†
*Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA
atsiour@mit.edu
†School of Electrical and Computer Engineering, National Technical University of Athens, Greece
{kbitsak, nkoziris}@cslab.ece.ntua.gr, fotakis@cs.ntua.gr
‡Department of Computer Science and Telecommunications, University of Thessaly, Greece
ikons@uth.gr

***Abstract*—Modern large-scale computing deployments consist of complex elastic applications running over machine clusters. A current trend adopted by providers is to set unused virtual machines, or else spot instances, in low prices to take advantage of spare capacity. In this paper we present a group of efficient allocation and pricing policies that can be used by vendors for their spot price mechanisms. We model the procedure of acquiring virtual machines as a truthful knapsack auction and we deploy dynamic allocation and pricing rules that achieve near-optimal revenue and social welfare. As the problem is NP-hard our solutions are based on approximate algorithms. First, we propose two solutions that do not use prior knowledge. Then, we enhance them with three learning algorithms. We evaluate them with simulations on the Google Cluster dataset and we benchmark them against the Uniform Price, the Optimal Single Price and the Ex-CORE mechanisms. Our proposed dynamic mechanism is robust, achieves revenue up to 89% of the Optimal Single Price auction, and computes the allocation in polynomial time making our contribution computationally tractable in real-time scenarios.**

***Index Terms*—Spot Instances, Mechanism Design, Learning, Revenue Maximization.**

I. Introduction

Nowadays, cloud computing services offer prompt on-demand access on multiple computing resources. Commercial platforms such as Amazon EC2 [1] and Microsoft Azure [2], invest on their infrastructure and provide a wide range of CPU, RAM and storage options.

One of the most popular platforms is Amazon EC2 Spot Instances [3]. Amazon sells unused EC2 capacity on low prices, called spot prices, through the EC2 Spot Instances system. When a user requests computing capacity, they specify the maximum amount they are willing to pay per hour per Virtual Machine (VM). If the maximum offer is greater or equal to the spot price, the user acquires/maintains their computing resources. Instead, if the maximum offer is less than the spot price, the VM can be reclaimed by EC2 in a short amount of time. The underlying mechanism of the pricing scheme is not revealed. Multiple studies have been conducted [4], [5] to learn more about the underlying pricing scheme. As the pricing policy is not revealed, middle to small sized vendors may have trouble moving from a fixed price scheme to an efficient dynamic pricing scheme [6]. In static fixed-price scenarios, users usually submit bids close to the fixed price. In this case the revenue of the provider may not be maximized as the bidders usually do not report their truthful valuation for the desired VMs. Therefore, users with lower actual valuations may acquire VMs instead of users with higher valuations, since both groups of users may submit similar bids. In this case the revenue is not maximized. To overcome these issues we make the following contributions.

We propose a dynamic allocation and pricing system based on Mechanism Design [7], [8] by modeling our system as a truthful knapsack auction. First, we solve the knapsack problem without taking into consideration previous data regarding users' bids. We provide two distinct solutions. The first is the classical greedy algorithm approach while the second is an alternate approximate algorithm combined with a randomized algorithm [9], [10]. Then, we apply learning methods to the previous solutions, such as the Maximum Likelihood Estimation [11], [12] to estimate the distribution of each user's valuation, a modification of the Expectation - Maximization algorithm [12], [13] for auctions that takes into consideration missing bids and a deep neural network to estimate the distribution of each user's valuation without making prior assumptions. We theoretically and experimentally evaluate our proposed algorithms, by providing provable guarantees for the approximate algorithms and by measuring their performance on realistic cluster datasets provided by Google [14].

II. Related Work

Auctions are efficient resource allocation mechanisms applied in different markets and as a result numerous studies have been conducted upon them. There are two types of auctions, the multi-unit [15] and the multi-item [16] auction. In a multi-unit auction multiple identical items are sold, while in a multi-item auction, users bid for bundles of different items.

*Work done while the author was with CSLab, NTUA

2159-6190/21/$31.00 ©2021 IEEE
DOI 10.1109/CLOUD53861.2021.00057

A special category of multi-unit auctions is the knapsack auction. In knapsack auctions, there is a limited pool of items and users bid to acquire a group of them. Knapsack auctions in which users bid truthfully to acquire goods have been studied thoroughly [7], [9], [10] and approximate solutions have emerged.

In cloud environments, dynamic resource provisioning systems for VMs which are based on auction theory have been studied. In [17] and [18], the dynamic resource allocation in a cloud market through multi-item auctions of heterogeneous VMs was studied. In these works, randomized auctions were used to maximize the social welfare of the system. Contrary to our approach, these solutions do not maximize the revenue of the provider and focus solely on social welfare. In [19] the authors suggest that spot pricing used by Amazon is truthful in markets with a single provider and show that users can increase their utility by being untruthful in a federated cloud environment. In [20] the authors proposed a multi-unit and single price auction for IaaS cloud resources that is envy-free and with high probability truthful mechanism that generates near optimal profit for the provider. In [21], online truthful auctions in IaaS clouds are designed to maximize users' social welfare along with the cloud provider's net profit during the running span of the system.

III. System Model And Preliminaries

We consider an auction-based resource provisioning model. In this setting, the cloud provider (auctioneer) has a pool of W similar VMs and there are n users who want to acquire a bundle of them. Each user i has a private valuation v_i for each VM and submits a bid b_i to acquire w_i VMs. The price p_i that the user i pays is calculated by the auctioneer. Given a feasible set X which contains vectors in the form of $(x_1, ..., x_n)^T$, where $x_i \in [0, 1]$ represents the percentage of the VMs that the user acquires at the end of the auction, our goal is to find an allocation $\mathbf{x} \in X$ that maximizes the revenue generated for the auctioneer under the knapsack constraint $\sum_{i=1}^{n} w_i x_i \leq W$. Each user i wants to maximize their utility defined as $u_i = \begin{cases} v_i - p_i, & i \text{ wins} \\ 0, & i \text{ loses} \end{cases}$. To motivate a user to participate in an auction, the relationship $p_i \leq b_i$ should hold. We define **social welfare** as the sum of users' valuations constrained upon the fraction of the VMs that each user finally acquires, i.e. $SW = \sum_{i=1}^{n} v_i w_i x_i$, while the revenue generated for the cloud provider is defined as $R = \sum_{i=1}^{n} p_i$. We provide the definition of truthful auctions, as all of our solutions fulfill this property, as well as some other useful definitions.

Definition 1. *An auction is truthful if for any user i, reporting their true valuation as their bid b_i, maximizes their expected utility, regardless of the bids submitted by other users.*

Definition 2. *If an auction is truthful and guarantees non negative utility then it is Dominant-Strategy Incentive Compatible (DSIC).*

Definition 3. *An allocation rule $\mathbf{x}$ is called implementable if a payment rule $\mathbf{p}$ exists in order for the auction to be DSIC.*

Given that $\mathbf{b}_{-i}$ is the vector $\mathbf{b} = (b_1, ..., b_n)^T$ of all bids, but with the ith component removed, we define the monotone allocation rule.

Definition 4. *An allocation rule $\mathbf{x}$ is monotone if for every user i and bids $\mathbf{b}_{-i}$ from the other users, the allocation $x_i(z, \mathbf{b}_{-i})$ to i is non decreasing in their bid z.*

According to the definition above, a monotone allocation rule guarantees that while a user increases their bid, the goods received will only increase or stay the same for constant $\mathbf{b}_{-i}$. Given all these definitions we present Myerson's Lemma [7], [22], a powerful tool that specifies the prices in order for the auction to be truthful.

Lemma 1. *According to Myerson's Lemma:*

- *An allocation rule $\mathbf{x}$ is implementable if and only if it is monotone.*
- *If $\mathbf{x}$ is monotone, then there is a unique payment rule for which the auction mechanism ($\mathbf{x}$,$\mathbf{p}$) is DSIC and $p_i(\mathbf{b}) = 0$, whenever $b_i = 0$.*
- *The payment rule is given by the formula $p_i(b_i, \mathbf{b}_{-i}) = \sum_{j=1}^{l} z_j \cdot$ [jump in $x_i(\cdot, \mathbf{b}_{-i})$ at z_j]*

Using Myerson's Lemma we construct truthful mechanisms. This mechanism rewards the users with the greatest valuations. Our concern is to create mechanisms that maximize the expected revenue of the cloud provider, i.e. $\mathbb{E}[\sum_{i=1}^{n} p_i(\mathbf{v})]$. To solve this problem, we study the *average-case* model. We assume that the private valuation v_i of user i is drawn from the distribution F_i. Our problem is transformed into maximizing the expected revenue $\mathbb{E}_{\mathbf{v}\sim\mathbf{F}}[\sum_{i=1}^{n} p_i(\mathbf{v})]$ of a DSIC mechanism ($\mathbf{x}$,$\mathbf{p}$), where $\mathbf{F} = F_1 \times F_2 \times ... \times F_n$. By observing this formula, it is not clear how to maximize it. Therefore, we work towards an equivalent formula that is easier to handle. This second formula incorporates the concept of virtual valuations.

Definition 5. *For a user i with valuation distribution F_i and valuation v_i, their virtual valuation is defined as*

$$\phi_i(v_i) = v_i - \frac{1 - F_i(v_i)}{f_i(v_i)}$$

If the auctioneer had prior knowledge about the exact value of the user's valuation v, then the optimal price would be v, as it is the upper bound on the price the user can pay for product. This is the first part of the virtual valuation formula. Since the valuation is not constant, but follows the F distribution, the expected revenue is influenced by the randomness of the procedure. Actually, the second term of the virtual valuation is the expected loss in revenue incurred due to the randomness of the buyer's valuation. We present the following theorem.

Theorem 1. *For every auction environment with valuation distributions $F_1, ..., F_n$ and every DSIC mechanism*

$$\mathbb{E}_{\mathbf{v}\sim\mathbf{F}}[\sum_{i=1}^{n} p_i(\mathbf{v})] = \mathbb{E}_{\mathbf{v}\sim\mathbf{F}}[\sum_{i=1}^{n} \phi_i(v_i) \cdot x_i(\mathbf{v})]$$

The proof is presented in [7]. The right term in this equation is the second equivalent formula that is easier to maximize. To maximize this formula for each $\mathbf{v}$ we choose a proper $x(\mathbf{v})$ that

maximizes the *virtual welfare* $\sum_{i=1}^{n} \phi_i(v_i)x_i(\mathbf{v})$. If this rule is monotone, it can be extended to a DSIC mechanism and by the previous theorem this mechanism maximizes the expected revenue. The monotonicity of the virtual welfare maximizing rule depends on the valuation distributions. We define regular distributions.

Definition 6. *A distribution F is regular if the corresponding virtual valuation function $\phi(v)$ is non decreasing.*

The following lemma provides the condition under which the virtual welfare allocation rule is monotone.

Lemma 2. *If every user's valuation is drawn from a regular distribution, then the virtual welfare maximizing rule is monotone.*

As a consequence, with regular valuation distributions we extend the virtual welfare maximizing allocation rule to a DSIC mechanism using Myerson's Lemma and we construct a truthful auction that maximizes the expected revenue. We present the procedure.

1) Transform the truthfully reported valuation v_i of the user i into the virtual valuation $\phi_i(v_i)$.
2) Choose the feasible allocation $(x_1, ..., x_n)^T$ that maximizes $\sum_{i=1}^{n} \phi_i(v_i)x_i(\mathbf{v})$.
3) Charge payments according to Myerson's payment rule.

Contrary to valuations, virtual valuations can be negative. The virtual welfare maximizing mechanism rewards only the users with non-negative virtual valuations. The valuation v_i that corresponds to virtual valuation $\phi_i(v_i) = 0$ can be interpreted as a reserve price, formally $\phi_i^{-1}(0)$. If for user i, $v_i < \phi_i^{-1}(0)$ the user is not rewarded. Finally, we define randomized auctions [23] which are used widely in revenue maximization [24] and are proved to be efficient in the case of the unlimited supply problem, i.e. $\sum_i^n w_i \leq W$.

Definition 7. *In a randomized auction the procedure of calculating the allocation and pricing rule is a random process.*

The revenue, the pricing rule and the allocation rule are random variables in a randomized auction.

IV. Algorithms for Revenue Maximization

A. Non - Learning Algorithms

We propose non-learning algorithms for vendors that do not have sufficient past data about the users in order to apply the learning algorithms. This collection of non-learning algorithms can be applied successfully for the start-up stage of the spot instance service. When a significant amount of data is collected, the vendor may choose to migrate to learning algorithms. Our first proposed solution, is based upon the classical greedy solution of the knapsack problem. This solution uses the *per-unit value* of each user as the main criterion to create a feasible allocation.

Definition 8. *The greedy knapsack algorithm (GK) does the following:*

1) *Sort the users in a per-unit value descending order.*
2) *Select the users in the aforementioned ordering until the provider cannot support more requests.*
3) *The set of winners is either the solution described above, or the user with the highest bid, depending on which of the two solutions achieves the highest social welfare.*

The allocation rule produced by GK is monotone and therefore we can apply Myerson's Lemma. Also, it is a $\frac{1}{2}$-approximation for the knapsack problem, meaning that for every instance of the knapsack problem, the algorithm returns a feasible solution with total value at least $\frac{1}{2}$ times the maximum possible. The second proposed non-learning solution [9] is based on approximate and randomized algorithms. If $\sum_{i=1}^{n} w_i > W$, we encounter the case of the limited supply where we apply the following algorithm.

Definition 9. *The limited supply approximate algorithm does the following:*

1) *Remove the users that request a bundle of VMs greater than half of the available capacity.*
2) *Sort the users in a per-unit value descending order.*
3) *Select the winners in the aforementioned ordering until the provider cannot support more requests.*
4) *Charge the winners a price equal to the amount of their acquired VMs multiplied by the largest per-unit value that was left out of the knapsack.*

The aforementioned algorithm is truthful. On input N it selects a feasible set H satisfying $OPT(H) \geq OPT(N)/3 - h$, where OPT is the profit obtained by the best monotone pricing function for users' actual valuations and h is an upper bound on the highest user's valuation. In the unlimited supply case, which is a typical case for large cloud providers, we present the Unlimited Supply Auction (USK) algorithm which is the convex combination of two randomized algorithms. Before we present the first algorithm, we provide an essential definition.

Definition 10. *A monotone pricing rule with exponential intervals is a monotone pricing rule in which the winners can be partitioned into equal priced intervals over their valuations such that the i^{th} interval (in decreasing order of weights) contains at least 2^{i-1} winners.*

We present the first algorithm, called Random Sampling Knapsack (RSK).

Definition 11. *The RSK does the following:*

1) *Partition the users into two sets A and B uniformly.*
2) *Compute the optimal monotone pricing rule with exponential intervals (restricting price to powers of two) for each partition. The pricing rules for each set are π_A and π_B.*
3) *Use π_A for B and π_B for A.*

The second algorithm is called general attribute algorithm (GAA) [10] and its steps are presented below.

Definition 12. *The GAA, given the parameters α, p does the*

following:

1) *List users in order of decreasing per-unit value* $\frac{b_i}{w_i}$.
2) *Given that* h *is the upper bound on the highest user's valuation, we define the variable* ***experts*** = $\lfloor \log_\alpha h \rfloor + 1$ *that represents the number of virtual experts that offer a feasible price for each VM bid of each user.*
3) *For each expert j, we define a virtual score* $s_j = k\alpha^j$ *with probability* $(1-p)^k p$*, where* k *is the total amount of Bernoulli trials that were made until the first event of probability* p *arrives.*
4) *Starting from the first user, we select for each user* i *the expert* j *with the highest virtual score (in case of tie we choose arbitrary, but constantly) and we offer the user the price* α^j.
5) *We then update the scores of every expert that could make a sale (*$a^j \leq b_i$*) according to the rule* $s_j \leftarrow s_j + a^j$.
6) *We repeat this procedure for every user.*

Both RSK and GAA are truthful [10], [9]. We now present the complete USK method for revenue maximization.

Definition 13. *The Unlimited Supply Knapsack algorithm does the following:*

1) *Perform the first step of RSK.*
2) *With probability* p*, run the GAA on sets* A *and* B *and with probability* $1-p$ *run the remaining steps of RSK.*

It is proved [9] that USK is truthful and the revenue generated is $\alpha OPT - \gamma h(\log\log\log n_A + \log\log\log n_B + c)$, where α, γ, c are constants.

B. Learning Algorithms

In our first learning approach, we use classical estimation theory to estimate the parameters of each user's valuation distribution. Given a vector of samples $\mathbf{X} = (x_1, ..., x_n)^T$ from a single distribution and assuming that the underlying distribution $p(\mathbf{x}|\boldsymbol{\theta})$ is described by a vector of parameters $\boldsymbol{\theta}$, we want to make an accurate estimation $\hat{\boldsymbol{\theta}}$ of this vector. To estimate the vector $\boldsymbol{\theta}$, we apply the Maximum Likelihood Estimation method [11].

In our second learning approach, we estimate users' valuation distribution by taking into account *"missing bids"*. When users with valuation lower than the current spot price enter the system to acquire spot instances, they do not make an offer, as they will certainly not acquire a VM. However, their valuations are essential in estimating the distribution. Therefore, we use a modification of the Expectation - Maximization algorithm (EM) [13], [25] to estimate the missing bids. We assume that each user follows the valuation distribution $f(x|\boldsymbol{\theta})$ and the arrival rate of the users follows the distribution $g(x|\boldsymbol{\lambda})$. Given the set of samples D we split the data in $D = (\mathbf{x}_v, \mathbf{x}_h)$ where $\mathbf{x}_v$ represents visible data and $\mathbf{x}_h$ hidden data. We assume that m total users exist, n of them make an offer and $m-n$ are the *"missing bids"*. Our goal is to find the vector $\hat{\boldsymbol{\theta}}$ that maximizes the function $Q(\theta) = \int \log(p(\mathbf{x}_h, \mathbf{x}_v|\boldsymbol{\theta}))p(\mathbf{x}_h|\mathbf{x}_v, \boldsymbol{\theta}^{(old)}, \boldsymbol{\lambda}^{(old)})d\mathbf{x}_h$ generated in step E of the EM algorithm. For its computation we present the following method for generating hidden bids. The method resembles the procedure of a user bidding for a spot instance.

Definition 14. *The hidden bid generating method is the following:*

1) *Set as the spot price the lowest of the observed offers.*
2) *Generate a sample* $m > n$ *from the distribution* $g(x|\boldsymbol{\lambda})$ *that represents the total number of users that arrived to the system at the specific round.*
3) *Draw* $m-n$ *random offers, smaller than the spot price, that represent the missing bids from* $f(x|\boldsymbol{\theta})$.

In our third learning approach, we use deep learning (DL) to estimate directly the virtual valuation of each user without any assumption on the distributions. The method is based on [26], [27]. We estimate the virtual valuations with a two layer min - max neural network. According to [26] this type of neural network is proved to approximate uniformly to an arbitrary degree of accuracy $\epsilon > 0$ any continuous, bounded, differentiable function which is monotonic in all variables. Since, the virtual valuations that correspond to regular distributions satisfy these criteria the network is suitable for approximating virtual valuations. Regarding the architecture, we define K groups, each of which consists of J linear functions with slope $w_{kj}^i = e^{\alpha_{kj}^i}$, where $\alpha_{kj}^i \in [-B, B]$ with $B > 0$, and intercepts β_{kj}^i, where $k = 1, ..., K$ and $j = 1, ..., J$. We approximate every virtual valuation function as $\hat{\phi}_i(b_i) = \min_{k\in[K]} \max_{j\in[J]} w_{kj}^i b_i + \beta_{kj}^i$. To decide the winners of the knapsack auction, we add a virtual user with virtual valuation equal to zero in order to not grant VMs to users with negative virtual valuations. We pass each virtual valuation through the softmax function to decide the winners. The loss function is the negative of the expected revenue, described formally as $\widehat{rev}(\boldsymbol{x}, \boldsymbol{p}) = \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{n} x_i(v^{(l)})p_i(v^{(l)})$. It is proved that network preserves the truthful property [27]. We proceed by presenting the experimental results.

V. Experimental Results

A. Simulation Environment and Performance Evaluation

We evaluate our algorithms using trace-driven simulation against realistic cluster usage traces provided by Google [14]. We study the popular a1.medium VM [3] with 1 CPU and 2Gb of Ram. We transform each job request into a bidding bundle by calculating the number of a1.medium VMs that combined make up the resource demands in the job request. Each user's i valuation follows the regular Lognormal(μ, σ^2) distribution with $\mu = 2 + 0.05i - \alpha \cdot \mathcal{N}$ and $\sigma = 0.25$, where $\alpha = 0.01(init - remaining)$, $init$ is equal to the initial amount of the VMs that each user requests, $remaining$ is equal to the number of the remaining VMs that the user needs to complete their workload and $\mathcal{N}$ is the absolute value of a random variable that follows the standard normal distribution. The $\alpha \cdot \mathcal{N}$ is a random noise term that represents the case in which when the majority of a user's requests are fulfilled and their workload comes close to its end, the user usually reduce

their bid. In this way, we test the robustness of our algorithms and observe how well they adapt to valuations with varying distributions. Each user has a total workload $WL \sim \mathcal{U}[10, 20]$. We simulate $R = 72$ discrete rounds. Our cluster consists of 40 VMs and 10 to 30 users participate in each round. We chose these settings to simulate the unlimited supply case (10 users), the limited supply case (30 users) and the combination of both cases (20 users). To evaluate the performance of our algorithms we compute the revenue, the social welfare and the CPU utilization achieved.

B. Baselines

We compare our algorithms with the following three auction mechanisms. The first auction mechanism is the Optimal Single Price auction $\mathcal{F}$ that is defined as follows.

Definition 15. *Let* $\mathbf{d}$ *be the vector consisting of each user's* i *per unit bid* b_i *and request for capacity* w_i*. Given that* $\mathbf{d}$ *is sorted in descending order based on bids, the auction* $\mathcal{F}$ *on input* $\mathbf{d}$ *determines the value* k *such that* $b_k \sum_{i=1}^{k} w_i$ *is maximized. All users with* $b_i \geq b_k$ *win at price* b_i*. The revenue generated on input* $\mathbf{d}$ *is:*

$$\mathcal{F}(\mathbf{d}) = \max_i b_i \sum_{j=1}^{i} w_j$$

OPT is not truthful [20]. The revenue in a single-round and single-price auction is at most OPT. Therefore, our goal is to create auction mechanisms that are competitive with $\mathcal{F}$, i.e. generate revenue as close to OPT.

The second benchmark is the Uniform Price Auction (UPA), in which each user is served exhaustively until the capacity is exhausted or there are no more requests. Every user is charged the per-unit price of the lowest winning bid.

The third benchmark is the Online Extended Consensus Revenue Estimate Mechanism (Online Ex-CORE) Auction algorithm proposed by Toosi et. al [20]. This algorithm is a renowned online algorithm for pricing by cloud providers that offer Infrastucture as a Service (IaaS) capabilities that maximizes profit and balances resource supply and demand.

C. Evaluation of Non-Learning Algorithms

We begin with the comparison between the two non-learning algorithms. For brevity, we call approximate knapsack algorithm (AK), the second proposed non-learning solution that is based on approximate (limited supply case) and randomized algorithms (unlimited supply case).

In Table I, we observe that the AK algorithm generates more revenue than the greedy knapsack (GK) algorithm in every case. As mentioned in sec. IV, GK favors social welfare. Therefore, since the maximization of social welfare is contrary to revenue maximization, in GK the revenue is not optimized. As expected, GK achieves better social welfare and CPU utilization than AK in every setup. One key observation is that the randomized algorithms used for the unlimited supply case (10 users) grant high revenue.

TABLE I: Metrics - Non Learning Algorithms

Method	Metric	10 users	20 users	30 users
GK	Revenue	631.72	18,165.43	37,848.43
	Social Welfare	**16,404.03**	**38,130.29**	**63,213.33**
	CPU Util (%)	**61.18**	**98.05**	**99.65**
AK	Revenue	**7,319.82**	**21,167.37**	**38,766.2**
	Social Welfare	15,326.63	36,551.79	62,311.5
	CPU Util (%)	56.67	93.02	95.24

D. Evaluation of Learning Algorithms

First, we present the simulation results regarding the effectiveness of our first two learning solutions, the MLE and the modified EM. We assume that each user's valuation follows the same distribution (i.e. the log-normal distribution with $\mu = 2$, $\sigma = 0.25$), the arrival rate of the users follow the Poisson distribution with $\lambda = 20$ and a random portion (between $10\% - 50\%$) of the total bids is not observed. The Kullback-Leibler (KL) divergence [28] from the actual distribution for both methods can be seen in Table II.

TABLE II: MLE versus EM

Missing Bids (%)	MLE KL	EM KL
10	**0.441157**	**0.441157**
20	0.862317	**0.356689**
30	1.306289	**0.622689**
40	1.801786	**0.438974**
50	2.419784	**0.235461**

In almost every case, the modified EM algorithm makes a better estimation than the MLE.

TABLE III: Metrics - Learning Greedy Algorithm

Method	Metric	10 users	20 users	30 users
GK MLE	Revenue	7,036.86	20,934.58	37,371.74
	Social Welfare	9,005.6	29,720.14	59,197.02
	CPU Util (%)	26.88	66.18	86.80
GK EM	Revenue	**9,245.85**	**21,312.89**	37,443.99
	Social Welfare	12,457.65	30,540.33	62,055.03
	CPU Util (%)	41.11	69.02	94.41
GK Deep	Revenue	1,531.12	18,377.67	**37,858.16**
	Social Welfare	**16,404.03**	**38,130.29**	**63,213.33**
	CPU Util (%)	**61.18**	**98.05**	**99.65**

We proceed with our evaluation by presenting and comparing the results of the learning enhanced GK algorithm (Table III). We observe that the GK combined with learning increases significantly its revenue in the first two cases. In the case of the unlimited supply, the revenue is increased up to $9,245.86/631.72 \approx \mathbf{\times 14.5}$ **times** when using the EM method. We observe that in the limited supply case the EM generates the best results, while in the unlimited supply case the DL approach performs best. The DL method grants the best results in the rest of the metrics in every case.

Next, we present the results of the learning-enhanced AK algorithm in Table IV. We observe that the highest metrics are granted by the DL approach. We believe that the other two learning methods fail, as they wrongly assume the type of distribution and that each user's valuation distribution is the same. For example, if these two methods assumed that the users' valuations follow the normal distribution instead of the

TABLE IV: Metrics - Learning Approx. Algorithm

Method	Metric	10 users	20 users	30 users
AK MLE	Revenue	4,254	15,046.37	34,460.91
	Social Welfare	8,551.91	27,769.91	56,398.72
	CPU Util (%)	25.59	61.56	81.42
AK EM	Revenue	6,273	15,651.22	37,299.54
	Social Welfare	12,110.65	28,585.78	59,885.36
	CPU Util (%)	39.20	64.55	89.65
AK Deep	Revenue	**7,319.82**	**21,167.37**	**38,766.2**
	Social Welfare	**15,326.63**	**36,551.79**	**62,311.5**
	CPU Util (%)	**56.67**	**93.02**	**95.24**

log-normal, the revenue would be even worse as seen in Table V. Finally, we present the metrics by the benchmark methods.

TABLE V: Revenue - Wrong Distribution Assumptions

Method	10 users	20 users	30 users
GK MLE	6,928.56	19,608.73	37,041.24
GK EM	8,206.45	20,674.11	36,415.11
AK MLE	4,091	13,828.77	26,963.23
AK EM	4,843	14,427.32	30,678.94

TABLE VI: Metrics - Benchmark Auctions

Method	Metric	10 users	20 users	30 users
OPT	Revenue	**11,525.15**	**25,411.8**	**43,394.11**
	Social Welfare	14,764.58	34,905.06	58,558.23
	CPU Util (%)	61.14	94.68	95.97
UPA	Revenue	9,891.2	21,628.22	36,519.09
	Social Welfare	**16,404.03**	**38,120.3**	**63,213.33**
	CPU Util (%)	**61.18**	**98.05**	**99.65**
Ex-CORE	Revenue	3,331.16	11,341.12	15,952.45
	Social Welfare	4,440.76	16,372.15	32,907.44
	CPU Util (%)	13.96	35.69	58.54

By observing Table VI, we notice that the approximate algorithms, both the original and the DL approach, surpass the UPA and Ex-CORE achieved revenue for the limited and the unlimited supply case. Moreover, regarding OPT, our learning algorithms achieve up to **89**$\% \approx 38,766.2/43,394.11$ as great revenue. The social welfare and the CPU utilization achieved by our greedy learning algorithms surpass the results of OPT and Ex-CORE auction and are similar to the UPA. It can be seen, that regardless of the metric, our proposed algorithms can provide feasible, computationally tractable solutions that perform well against different benchmarks.

VI. Conclusion

In this paper, we presented how Mechanism Design and Learning Theory can enhance the generated revenue and the total social welfare on Cloud Dynamic Spot Markets. We modeled the procedure of acquiring VMs as a truthful knapsack auction. We applied learning methods, such as the MLE, a modified version of the EM algorithm and a deep neural network to estimate each user's valuation distribution. By estimating each user's valuation distribution, we modified the non-learning algorithms in a way that leads to revenue maximization. Evaluation on Google cluster dataset and comparison with well-known baselines, showed that learning methods achieve increased revenue in every possible setting up to 89% of the Optimal Single Price auction. Our solutions provide substantial results on every metric, while they compute the allocation in polynomial time, making our contribution computationally tractable for real scenarios.

We believe that the DL approach is the most robust learning method for estimating each user's virtual valuation, as it does not make any assumption about the underlying distribution of users' valuation. Taking all of the above into consideration, we conclude that the approximate knapsack solution combined with the DL method grants competitive results in every metric, while it is the most robust and can adapt accurately to every regular distribution.

References

[1] "Amazon EC2," https://aws.amazon.com/ec2/. [Online]. Available: https://aws.amazon.com/ec2/
[2] "Microsoft Azure Cloud Computing Platform & Services," https://azure.microsoft.com/en-us/. [Online]. Available: https://azure.microsoft.com/en-us/
[3] "Amazon EC2 Spot Instances," https://aws.amazon.com/ec2/spot/. [Online]. Available: https://aws.amazon.com/ec2/spot/
[4] M. Baughman *et al.*, "Deconstructing the 2017 Changes to AWS Spot Market Pricing," in *Workshop on Scientific Cloud Computing*, 2019.
[5] O. Agmon Ben-Yehuda *et al.*, "Deconstructing Amazon EC2 Spot Instance Pricing," *ACM Trans. Econ. Comput.*, 2013.
[6] G. Cachon and P. Feldman, "Dynamic versus Static Pricing in the Presence of Strategic Consumers," *University of Pennsylvania*, 2011.
[7] T. Roughgarden, *Twenty Lectures on Algorithmic Game Theory*, 1st ed. Cambridge University Press, 2016.
[8] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. The MIT Press, 1994.
[9] G. Aggarwal and J. D. Hartline, "Knapsack Auctions," in *SODA*, 2006.
[10] A. Blum and J. D. Hartline, "Near-optimal Online Auctions," in *SODA*, 2005.
[11] D. Bertsekas and J. Tsitsiklis, *Introduction to Probability*, ser. Athena Scientific optimization and computation series. Athena Scientific, 2008.
[12] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.
[13] T. Moon, "The expectation-maximization algorithm," *Signal Processing Magazine, IEEE*, 1996.
[14] "google/cluster-data: Borg cluster traces from Google." [Online]. Available: https://github.com/google/cluster-data
[15] S. Dobzinski and N. Nisan, "Mechanisms for Multi-Unit Auctions," *CoRR*, vol. abs/1401.3834, 2014.
[16] G. Demange, G. David, and M. Sotomayor, "Multi-Item Auctions," HAL, Post-Print, 1986.
[17] W. Shi *et al.*, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," *IEEE/ACM Transactions on Networking*, 2016.
[18] L. Zhang *et al.*, "Dynamic Resource Provisioning in Cloud Computing: A Randomized Auction Approach," in *IEEE INFOCOM*, 2014.
[19] M. Mihailescu and Y. Teo, "The Impact of User Rationality in Federated Clouds," *CCGrid*, 2012.
[20] A. N. Toosi *et al.*, "An Auction Mechanism for Cloud Spot Markets," *ACM Transactions on Autonomous and Adaptive Systems*, 2016.
[21] X. Zhang *et al.*, "Online Auctions in IaaS Clouds: Welfare and Profit Maximization With Server Costs," *IEEE/ACM Transactions on Networking*, 2017.
[22] R. B. Myerson, "Optimal Auction Design," *Math. Oper. Res.*, 1981.
[23] A. Mehta and V. V. Vazirani, "Randomized truthful auctions of digital goods are randomizations over truthful auctions," in *Proceedings of the 5th ACM conference on Electronic commerce*, 2004.
[24] A. V. Goldberg *et al.*, "Competitive Auctions," *Games and Economic Behavior*, 2006.
[25] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. Hoboken, NJ: Wiley, 2008.
[26] J. Sill, "Monotonic Networks," in *NeurIPS*, 1997.
[27] P. Dütting *et al.*, "Optimal Auctions through Deep Learning," *arXiv:1706.03459 [cs]*, 2017, arXiv: 1706.03459.
[28] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, 1951.