A Truthful (1-ϵ)-Optimal Mechanism for On-demand Cloud Resource Provisioning

Xiaoxi Zhang[†], Chuan Wu[†], Zongpeng Li[§], Francis C.M. Lau[†]
[†]Department of Computer Science, The University of Hong Kong, Email: {xxzhang2,cwu,fcmlau}@cs.hku.hk
[§]Department of Computer Science, University of Calgary, Canada, Email: zongpeng@ucalgary.ca

***Abstract*—On-demand resource provisioning in cloud computing provides tailor-made resource packages (typically in the form of VMs) to meet users' demands. Public clouds nowadays provide more and more elaborated types of VMs, but have yet to offer the most flexible dynamic VM assembly, which is partly due to the lack of a mature mechanism for pricing tailor-made VMs on the spot. This work proposes an efficient randomized auction mechanism based on a novel application of smoothed analysis and randomized reduction, for dynamic VM provisioning and pricing in geo-distributed cloud data centers. This auction, to the best of our knowledge, is the first one in literature that achieves (i) truthfulness in expectation, (ii) polynomial running time in expectation, and (iii) $(1-\epsilon)$-optimal social welfare in expectation for resource allocation, where ϵ can be arbitrarily close to 0. Our mechanism consists of three modules: (1) an exact algorithm to solve the NP-hard social welfare maximization problem, which runs in polynomial time in expectation, (2) a perturbation-based randomized resource allocation scheme which produces a VM provisioning solution that is $(1-\epsilon)$-optimal and (3) an auction mechanism that applies the perturbation-based scheme for dynamic VM provisioning and prices the customized VMs using a randomized VCG payment, with a guarantee in truthfulness in expectation. We validate the efficacy of the mechanism through careful theoretical analysis and trace-driven simulations.**[1]

I. Introduction

Cloud computing services are proliferating in today's Internet. They try to cater to users' resource demands of any scale at any time. To be flexible at meeting users' demands, cloud platforms such as Amazon EC2, Microsoft Azure and GoGrid exploit advanced virtualization technologies to pack CPU, RAM, and disk resources into virtual machine (VM) instances of various types. Undoubtedly, the more variety of VM types they can provide, the better they could meet the wide range of users' demands. For example, Amazon EC2 has been expanding the variety of VM instances they provide, which now spans 7 categories and 23 types [1]. However, the increased variety on the provider's side still often falls short of addressing the user's needs precisely, which could lead to a waste of resources and an unjustifiably inflated payment by the users. For example and suppose there is a user who needs to run a computationally intensive job (*e.g.*, MapReduce), and he wishes to acquire 16 vCPU units and 16 GB memory [2] in EC2's Singapore data center in order to process 160 GB usage data; the best match is a `c3.4xlarge` instance, which unfortunately is far from a perfect match, and the result is a waste of roughly half of the allocated memory and SSD storage.

Current virtualization technology is in fact ready for real-time, on-demand VM partitioning and provisioning (*e.g.*, by utilizing credit-based CPU scheduler and memory ballooning [3]). What is lacking seems to be an effective pricing mechanism that can decide on the spot the charges for the customized VMs. The current representative charging models, *e.g.*, long-term reservation, fixed on-demand instance pricing and spot instance pricing employed by Amazon EC2, are not suitable for dynamically assembled VMs. Under fixed pricing, it is impossible for the cloud provider to come up with the appropriate prices, *a priori*, for any VM type that could possibly be assembled according to the user's needs. Furthermore, fixed pricing fails to cater to the changing supply and demand in the market; either overpricing or underpricing would jeopardize the social welfare of the overall system as well as the provider's revenue. Amazon's spot instance market [4] represents the first attempt at a more market-driven pricing system, which however still lacks the truthfulness guarantee and service guarantee [5][6]. Some recent work further studied auction mechanism design for cloud resource provisioning from different perspectives [6][7][8][9]. However, most of them model VMs as type-oblivious commodities, and would fail to function property with VMs that are dynamically assembled.

Besides pricing, packing available resources in a data center to maximally cater to users' VM demands translates into an NP-hard combinatorial optimization problem, which presents a tough challenge in VM auction design. The VCG mechanism [10], essentially the only type of auction that guarantees both truthfulness and economic efficiency (social welfare maximization), requires an exact optimal allocation. When polynomial-time approximation algorithms are applied instead, VCG loses its truthfulness property [11]. To achieve truthfulness with an approximation algorithm, researchers have exploited the concept of *critical bids* [12], or resorted to some LP decomposition technique [13][14][15]. The approximation ratio of these auctions with respect to social welfare optimality depends on the efficiency of the approximation algorithm employed, which is typically much larger than 1 [14][15].

This work aims to leverage a set of recent, novel techniques from smoothed analysis [16] and randomized reduction, to design a randomized, highly efficient auction mechanism for

[1]This work was supported in part by a grant from RGC under the contract HKU 718513 and the Natural Sciences and Engineering Research Council of Canada (NSERC).

provisioning and pricing customized VM instances in a geo-distributed cloud. The resulting combinatorial VM auction is sufficiently expressive for cloud users to request the necessary custom-made VM instances in bundles in different data centers for their job execution. To the best of our knowledge, this is the first VM auction that achieves (i) truthfulness (in expectation), (ii) polynomial running time (in expectation), and (iii) $(1-\epsilon)$-optimal social welfare (in expectation) for resource allocation in a geo-distributed cloud, where $\epsilon \in (0,1)$ is a tunable parameter that can approach zero.

Our proposed auction mechanism consists of three main modules: (1) an exact algorithm to solve the NP-hard social welfare maximization problem, which runs in polynomial time in expectation and is based on smoothed analysis. It serves as the basis for resource allocation in (2); (2) a perturbation-based randomized resource allocation scheme that produces a VM provisioning solution that is $(1-\epsilon)$-optimal in social welfare in expectation; and (3) an auction mechanism that applies the perturbation-based scheme to dynamic VM provisioning, and prices the customized VMs using a randomized VCG payment, which guarantees truthfulness in expectation. The following are the steps involved.

First, we formulate the social welfare optimization problem as an integer linear program and then prove its NP-hardness. Based on smoothed analysis, we randomly perturb the objective function of the optimization following a well designed perturbation framework, and propose an exact dynamic programming based algorithm to solve the perturbed problem. The algorithm finds a feasible solution to the original, unperturbed problem with polynomial running time in expectation. Furthermore, a transformation of this feasible solution yields a fractional solution to the original problem, which achieves $(1-\epsilon)$-optimal social welfare in expectation.

Second, we design a randomized resource allocation scheme, which outputs the allocation solution of the auction following a well designed distribution over a set of feasible solutions of the social welfare maximization problem, including the feasible solution produced in the above step. By designing the distribution in close connection with the perturbation framework, we are able to show that the expectation of such a randomized solution equals the fractional solution mentioned above, and hence it achieves $(1-\epsilon)$-optimal social welfare in expectation.

Third, we combine the randomized resource allocation scheme with a randomized VCG payment, and complete our auction design with truthfulness guaranteed in expectation.

An interesting result of our mechanism design, which arises from the contrast between the following pair of facts, represents perhaps the most surprising outcome of this work: (i) For the social welfare maximization problem we formulate, **even if truthful bids are given for free**, no (deterministic) polynomial-time algorithm can guarantee $(1-\epsilon)$-approximation for arbitrarily tunable ϵ [17]. (ii) The (randomized) VM auction designed in our work is both polynomial-time and $(1-\epsilon)$-optimal in expectation, and can simultaneously elicit truthful bids from selfish cloud users.

The above strong properties of our VM auction are made possible by unleashing fully the power of randomization, through the art of calculated random perturbation (for computational efficiency) and associated perturbation (for truthfulness). While there exist separate literature on applying randomization for efficient algorithm design and for truthful mechanism design respectively, to the best of our knowledge, this work is the first one that applies the same carefully prepared randomization scheme twice in subtly different forms and in a coordinated fashion to achieve low algorithm complexity and truthful mechanism design in the same auction framework. We believe that this new technique can be generalized to become applicable to a rich class of combinatorial auctions in which social welfare maximization can be modeled as a linear integer program that is NP-hard (otherwise our technique is unnecessary) but not too hard (which admits a smoothed polynomial time algorithm) to solve.

We discuss related work in Sec. II and present the system model in Sec. III. Sec. IV gives the complete auction design. Sec. V presents trace-driven simulation studies and Sec. VI concludes the paper.

II. Related Work

Resource provisioning in cloud computing has been extensively studied with different focuses. Beloglazov *et al.* [18] aim at minimizing the energy consumption in computing task scheduling. Alicherry *et al.* [19] study VM allocation in distributed cloud systems, taking into consideration the communication cost. Joe-Wong *et al.* [20] seek to balance efficiency and fairness for allocating resources of multiple types. None of them however focus on dynamic VM assembly and provisioning, which is the focus of our work.

Auction mechanisms have been applied to achieve efficient resource allocation in cloud systems. Zaman *et al.* [7] design a truthful auction based on an approximation algorithm for resource allocation, but without proving the performance of the resource allocation algorithm. Wang *et al.* [5] propose a truthful VM auction based on a greedy allocation algorithm and a well-designed payment method; the derived allocation solution approximates the optimal solution with a large approximation ratio which depends on the number of VMs. Zhang *et al.* [9] and Wang *et al.* [6] design online cloud auctions but they only consider a single type of VM instances, ignoring dynamic provisioning of different VMs. Similar to our work, Zhang *et al.* [14] and Shi *et al.* [15] address dynamic VM provisioning, and design truthful auctions by applying an LP decomposition technique, which achieve 2.72- and 3.30-approximation of optimal social welfare, respectively. This work departs from the existing literature by applying smoothed analysis and randomized reduction techniques to randomized auction design, which achieves a much better approximation to the optimal solution, *i.e.*, $(1-\epsilon)$-optimal social welfare (where ϵ can be very close to zero), while retaining truthfulness and computation efficiency in expectation.

A key technique we adopt in this paper is a novel use of smoothed analysis in designing an algorithm to produce a good

solution to the social welfare maximization in polynomial time in expectation. Smoothed analysis is a technique for analyzing the time complexity of an algorithm for an NP-hard problem, that exactly solves a perturbed instance of the problem based on a small, random perturbation, in order to show that the algorithm can be efficient in expectation despite its possible worst case complexity [16]. It has been argued that complexity analysis on the expectation over some distribution of the instances is more convincing than that of the average case, and more practical than that of the worst case [16]. Smoothed analysis has been applied recently in areas such as combinatorial programming [21], computational geometry [22], game analysis [23]. Dughmi *et al.* [24] focus on social welfare maximization problems with an FPTAS, and design a randomized reduction method to convert the FPTAS into a truthful mechanism. Unlike theirs, our NP-hard social welfare maximization problem does not have a deterministic FPTAS; even so, we are able to show, surprisingly, that we can still achieve a randomized, truthful, $(1-\epsilon)$-approximation mechanism with expected polynomial complexity by applying the carefully designed permutation framework.

III. System Model

A. Cloud Resource Aucion

Consider an IaaS cloud consisting of D geo-distributed data centers. The cloud provider offers users K types of resources, including CPU, RAM and storage, to be packaged as VMs for lease. Let $[X]$ denote the set of integers $\{1, 2, \ldots, X\}$. The overall capacity of resource of type $k \in [K]$ in data center (DC) $d \in [D]$ is $c_{k,d}$.

The cloud provider acts as an auctioneer and sells custom-made VMs to N cloud users through auctions. Assume M is the maximum number of VM types that the users may possibly request to run their jobs, and each VM of type $m \in [M]$ consumes an r_m^k amount of type-k resource, for all $k \in [K]$. Note that we allow flexible VM assembly on demand and the numbering of VM types is purely for the ease of presentation.

The N cloud users act as bidders in the auction. Let L be the total number of bids submitted by all the cloud users, and we use $i \in [L]$ to denote the index in the set of bids. Each bid B_i contains a list of desired quantities of VMs of different types, as well as the bidder's reported valuation for this bundle of VMs. More specifically, there is an $(MD+1)$-tuple of elements in a bid: $q_{m,d}^i, \forall m \in [M], \forall d \in [D]$, and b_i, where $q_{m,d}^i$ is the number of type-m VM instances the corresponding bidder requires in data center d, and b_i is the bidder's reported valuation for this bundle. Each cloud user $n \in [N]$ can submit multiple bids, but at most one bid can be successful. This assumption is reasonable given that any need for concurrently acquiring VM bundles in two or more bids can be expressed as a separate bid with a combined bundle. Let $\mathcal{B}_n$ denote the set of bids submitted by user n. We have $0 \leq |\mathcal{B}_n| \leq L$.

Upon receiving user bids, the cloud provider computes the outcome of the auction, including both (i) its VM allocation scheme, $\vec{x} = \{x_1, ..., x_L\}$, where binary variable x_i is 1 if bid i is successful and 0 otherwise, and (ii) a payment p_i for each winning bid i. Let v_i denote the true valuation of the bidder submitting bid i. The utility u_i acquired due to this bid is then:

$$u_i(B_i, \mathcal{B}_{-i}) = \begin{cases} v_i - p_i & \text{if } B_i \text{ is accepted} \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{B}_{-i}$ is the set of all bids in the auction except B_i. We summarize important notations in Table I for ease of reference.

TABLE I: Notation

N	# of users	L	# of bids
M	# of VM types	K	# of resource types
D	# of data centers	B_i	the ith bid
v_i	true valuation of bid i	u_i	utility of bid i
P	perturbation matrix	$\mathcal{B}_n$	bid set of user n
b_i	bidding price of bid i	$\bar{b}_i$	perturbed b_i

ϵ	parameter in $(0, 1)$
r_m^k	amount of type-k resource in a type-m VM
$q_{m,d}^i$	# of type-m VMs in DC d requested in bid i
$R_{k,d}^i$	demand of type-k resource in DC d in bid i
$c_{k,d}$	capacity of type-k resource in DC d
$s(\vec{x})$	social welfare under allocation solution $\vec{x}$
$C_{k,d}(\vec{x})$	demand for type-k resource in DC d under $\vec{x}$
θ_i	parameter in $[0, \epsilon/L]$
$\Omega(\vec{x})$	distribution based on $\vec{x}$, ϵ and $\vec{\theta}$
x_i	to accept (1) or reject (0) bid i
$\vec{x}^*$	optimal allocation solution of ILP (1)
$\vec{x}^p$	optimal allocation solution of ILP (3)
$\vec{x}^f$	fractional solution perturbed from $\vec{x}^p$
$\vec{y}^\epsilon$	auction's final allocation solution
$p_i(\vec{y}^\epsilon)$	payment of bid i under $\vec{y}^\epsilon$

B. Goals of Mechanism Design

We pursue the following desirable properties in our mechanism design. (i) *Truthfulness*: The auction mechanism is truthful if for any user n, declaring its true valuation of the VM bundle in each of its bids always maximizes its utility, regardless of other users' bids. Truthfulness ensures that selfish buyers are automatically elicited to reveal their true valuations of the VMs they demand, simplifying the bidding strategy and the auction design. (ii) *Social welfare maximization*: The social welfare is the sum of the cloud provider's revenue, $\sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} p_i x_i$, and the aggregate users' utility $\sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} (v_i - p_i) x_i$. Since the cloud provider's revenue and the payment from the users cancel out, the social welfare is equivalent to the overall valuation of the winning bids $\sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} v_i x_i$, which equals $\sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} b_i x_i$ under truthful bidding. Different from existing work that achieve only approximate social welfare optimality with a ratio much larger than 1, we seek to achieve $(1-\epsilon)$-optimality where ϵ is a tunable parameter that can be arbitrarily close to 0. (iii) *Computational efficiency*: A polynomial-time resource allocation algorithm is desirable for the auction to run efficiently in practice. Our auction mechanism leverages the power of randomization to break through the inapproximability

barrier of the social welfare maximization problem which does not have a deterministic FPTAS. Consequently, we target at polynomial time complexity of the mechanism *in expectation*.

Next, we formulate the social welfare maximization problem, which gives rise to the optimal resource allocation solution for the cloud provider to address users' VM demands, assuming truthful bidding is guaranteed. Here $R^i_{k,d} = \sum_{m=1}^{M} q^i_{m,d} r^k_m$ denotes the overall demand of type-k resource in data center d in bid i.

$$\text{maximize} \quad \sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} b_i x_i \tag{1}$$

subject to:

$$\sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} x_i R^i_{k,d} \le c_{k,d}, \quad \forall k \in [K], \forall d \in [D], \tag{1a}$$

$$\sum_{i\in\mathcal{B}_n} x_i \le 1, \quad \forall n \in [N], \tag{1b}$$

$$x_i \in \{0,1\}, \quad \forall i \in \mathcal{B}_n, \forall n \in [N].$$

Constraint (1a) states that the overall demand for each type of resource in the winning bids should not exceed the overall capacity of the resource in each data center. Constraint (1b) specifies that each user can win at most one bid.

Theorem 1. *The social welfare maximization problem defined in the integer linear program (ILP) (1) is NP-hard and there does not exist a deterministic FPTAS for the problem.*

The proof is given in our technical report [25].

IV. Auction Design

At a high level, our strategy for truthful VM auction design is to apply a randomized VCG-like payment mechanism that works in concert with a randomized allocation algorithm, with the latter achieving optimal social welfare in expectation. Such randomized auctions leverage maximal-in-distributional range (MIDR) algorithms, which are known to be a powerful tool for designing (randomized) truthful mechanisms [26]. More specifically, if we can design an MIDR allocation rule (*i.e.*, a randomized allocation algorithm that chooses an allocation solution randomly from a set of feasible solutions of the social welfare maximization problem, following a distribution that is independent of the bidders' bids, and leads to the largest expected social welfare as compared to all other such distributions in a range), then we can combine a randomized VCG payment scheme following a similar distribution to obtain an auction mechanism that is truthful in expectation. To achieve the other two goals of our auction design, the allocation algorithm should run in polynomial time and be $(1-\epsilon)$-optimal in social welfare, both in expectation.

Next we first design an exact algorithm to solve the social welfare maximization problem in Sec. IV-A. Then we apply a perturbation framework to design the randomized allocation algorithm in Sec. IV-B, making use of the exact algorithm, based on a novel application of smoothed analysis techniques. Then we describe the payment scheme in Sec. IV-C. The missing proofs of some of the lemmas and theorems can be found in our technical report [25].

A. An Exact Algorithm for Social Welfare Maximization

The basic idea of the exact algorithm is to enumerate all the feasible allocation solutions excluding those absolutely "bad" ones, and then select the optimal allocation solution $\vec{x}$ that achieves maximum aggregate bidding price (corresponding to maximum social welfare under truthful bidding) among the set of "good" feasible solutions. The set of "good" solutions are defined to be those *Pareto optimal* solutions which are not dominated by any other feasible solutions, and the "bad" ones are those dominated by at least one Pareto optimal solution. This is in line with classical dynamic programming approaches for enumerating Pareto optimal solutions in traditional combinatorial optimization [27].

Let $s(\vec{x}) = \sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} b_i x_i$ denote the social welfare under allocation solution $\vec{x}$, and $C_{k,d}(\vec{x}) = \sum_{n\in[N]} \sum_{i\in\mathcal{B}_n} x_i R^i_{k,d}$ be the total demand for type-k resource in data center d under $\vec{x}$. The Pareto optimal solutions are defined as follows.

Definition (Pareto Optimal Allocation) An allocation solution $\vec{x}$ is Pareto optimal if it satisfies all the constraints in ILP (1), and there does not exist a feasible solution $\vec{x}'$ that dominates $\vec{x}$, *i.e.*, $\nexists \vec{x}'$ such that $s(\vec{x}') \ge s(\vec{x})$ and $C_{k,d}(\vec{x}') \le C_{k,d}(\vec{x}), \forall k \in [K], \forall d \in [D]$, with at least one inequality being strict among the above, as well as $\sum_{i\in\mathcal{B}_n} x'_i \le 1$, $\forall n \in [N]$.

We find out all the Pareto optimal solutions using a dynamic programming approach: Let $\mathcal{P}(i)$ be the set of all Pareto optimal solutions when we only consider the first i bids in set $[L]$ (the bids in $[L]$ are ordered in any fashion). Let i-dimensional vector $\vec{x}^{(i)}$ denote a Pareto optimal solution in $\mathcal{P}(i)$. We compute $\mathcal{P}(i)$ from $\mathcal{P}(i-1)$, and eventually obtain $\mathcal{P}(L)$ which is the set of Pareto optimal solutions of ILP (1).

We show the following property of the Pareto optimal solution sets:

Lemma 1. *If $\vec{x}^{(i)}$ is a Pareto optimal solution in $\mathcal{P}(i)$, then the vector obtained by removing the last element $x^{(i)}_i$ from $\vec{x}^{(i)}$ is a Pareto optimal solution in $\mathcal{P}(i-1)$, $\forall i = 2, \ldots, L$.*

Let $\mathcal{P}(i-1)+1$ denote the set of i-dimensional solutions obtained by simply adding 1 as the ith element to each solution vector in $\mathcal{P}(i-1)$ (removing infeasible solutions), and $\mathcal{P}(i-1)+0$ be the set obtained by adding 0 as the ith element. Given Lemma 1, we know that any solution in $\mathcal{P}(i)$ must be contained in set $\mathcal{P}(i-1)+0 \cup \mathcal{P}(i-1)+1$. In the algorithm given in Alg. 1, we start with $\mathcal{P}(1)$, which contains two Pareto optimal solutions 1 (accept B_1) and 0 (reject B_1), if the resource demands in bid B_1 do not exceed the respective capacity limits, and contains only one Pareto optimal solution 0, otherwise. Then we construct $\mathcal{P}(i), i = 2, \ldots, L$, by eliminating infeasible or non-Pareto-optimal solutions from $\mathcal{P}(i-1)+0 \cup \mathcal{P}(i-1)+1$. Finally, the exact allocation solution of ILP (1) is obtained as the solution in $\mathcal{P}(L)$ that achieves

the maximum social welfare. The computation complexity of

Algorithm 1: The Exact Algorithm for ILP (1)

Input: $\vec{b}, \vec{R}, \vec{c}$
Output: exact optimal solution $\vec{x}$
if $C_{k,d}(\{1\}) \leq c_{k,d}, \forall k \in [K], \forall d \in [D]$ **then**
  $\mathcal{P}(1) = \{0, 1\}$;
else
  $\mathcal{P}(1) = \{0\}$;
for $i = 2, ..L$ **do**
  for all $\vec{x}^{(i-1)} \in \mathcal{P}(i-1)$ **do**
    $\vec{x}^{(i)} = \{\vec{x}^{(i-1)}, 1\}$;
    if $\vec{x}^{(i)}$ satisfies Constraints (1a) and (1b) **then**
      Put $\vec{x}^{(i)}$ into $\mathcal{P}(i-1)+1$;
  Merge $\mathcal{P}(i-1)+0$ and $\mathcal{P}(i-1)+1$ into $\mathcal{P}(i)'$;
  Prune the solutions dominated by others in $\mathcal{P}(i)'$ to obtain $\mathcal{P}(i) = \{\vec{x}^{(i)} \in \mathcal{P}(i)' | \nexists \vec{x}^{(i)\prime} \in \mathcal{P}(i)' : \vec{x}^{(i)\prime}$ dominates $\vec{x}^{(i)}\}$;
return $\vec{x} = \text{argmax}_{\vec{y} \in \mathcal{P}(L)} s(\vec{y})$

the exact Alg. 1 is polynomial in the number of Pareto optimal solutions in $\mathcal{P}(L)$, as given in Theorem 2, which is based on Lemma 2.

Lemma 2. *The number of Pareto optimal solutions $|\mathcal{P}(i)|$ does not decrease with i, i.e., $|\mathcal{P}(1)| \leq ... \leq |\mathcal{P}(L)|$.*

The proof is given in our technical report [25].

Theorem 2. *The computation complexity of Alg. 1 is $O(KDL|\mathcal{P}(L)|^2)$.*

The proof is given in our technical report [25].

The algorithm runs in exponential time in the worst case, since there can be exponentially many Pareto optimal solutions to check in the worst case. In what follows, however, we will show that this exact algorithm is efficient in practice and can be used as a building block in a perturbation framework, for producing a randomized allocation algorithm which runs in polynomial time in expectation.

B. The Randomized $(1-\epsilon)$-Approx. Allocation Algorithm

The basic idea of our randomized algorithm that can efficiently solve the social welfare maximization problem (1) is to obtain a set of feasible allocation solutions that achieve $(1-\epsilon)$-optimal social welfare in expectation, following a well-designed distribution, and then randomly output an allocation solution from this set following this distribution. To achieve computation efficiency, the set of feasible solutions are to be computed in polynomial time in expectation, including one from the random perturbation of the social welfare maximization problem, based on smoothed analysis techniques [16][21]. We apply a pair of associated random perturbation schemes—which is the most salient feature of this work and for the first time in the literature—for smoothed polynomial time algorithm design and for randomized auction design respectively. The random perturbation is carefully designed, in close connection with the distribution to sample feasible solutions, to achieve $(1-\epsilon)$-optimal social welfare of (1) in expectation.

Given an arbitrary parameter $\epsilon \in (0, 1)$ and L random variables $\{\theta_1, \theta_2, \ldots, \theta_L\}$ that are independently and identically chosen from $[0, \frac{\epsilon}{L}]$ (following any distribution that is not necessarily uniform). Let $\vec{\theta} = \{\theta_1, \ldots, \theta_L\}$. We perturb the weight b_i in the objective function of ILP (1), *i.e.*, each bidding price, independently to:

$$\bar{b}_i = (1-\epsilon)b_i + \frac{\theta_i \sum_{j=1}^{L} b_j}{L}, \forall i \in [L].$$

Let

$$P = (1-\epsilon)I + \frac{\vec{\theta}\vec{1}^T}{L} \tag{2}$$

be the perturbation matrix, where I is the $L \times L$ identity matrix. Then we can express the perturbation as $\vec{\bar{b}} = P\vec{b}$. The perturbed social welfare maximization problem is:

$$\begin{array}{ll} \text{maximize} & \sum_{n\in[N]} \sum_{i \in \mathcal{B}_n} \bar{b}_i x_i \\ \text{subject to:} & \text{constraints (1a)(1b)(1c).} \end{array} \tag{3}$$

We solve the perturbed social welfare maximization problems using the exact algorithm (Alg. 1), and derive the optimal solution $\vec{x}^p$ and optimum value of the perturbed objective function $POPT = \vec{\bar{b}}^T \vec{x}^p$. We will show that the expected running time to solve the randomly perturbed ILP is polynomial in Theorem 3 and Theorem 4. Let $\vec{x}^*$ be the optimal solution of ILP (1), and $OPT = \vec{b}^T \vec{x}^*$ be the optimal social welfare. We have

$$\begin{aligned} POPT &= (P\vec{b})^T \vec{x}^p \geq (P\vec{b})^T \vec{x}^* = \vec{b}^T((1-\epsilon)I + \frac{\vec{1}\vec{\theta}^T}{L})\vec{x}^* \\ &\geq (1-\epsilon)\vec{b}^T\vec{x}^* = (1-\epsilon)OPT, \end{aligned} \tag{4}$$

i.e., the optimal objective value of the perturbed problem is at least $(1-\epsilon)$ of the optimal social welfare of the original problem, which is very close as long as the perturbation, decided by ϵ, is small enough. Given that $(P\vec{b})^T\vec{x}^p = \vec{b}^T(P^T\vec{x}^p)$, we also obtain a potential solution $\vec{x}^f = P^T\vec{x}^p$ to the original problem, which achieves $(1-\epsilon)$-optimal social welfare. However, the bad news is that $\vec{x}^f$ may well be fractional due to the fractional entries in P^T, and hence not a feasible solution of ILP (1) (not to mention whether it satisfies other constraints in (1) or not). We hence cannot directly use $\vec{x}^f$ as the allocation solution to our social welfare maximization problem (1), but design a random sampling approach to produce a feasible allocation solution from a set of feasible solutions of (1) following a well-designed distribution, such that the expectation of the randomly produced solution is $\vec{x}^f$, which achieves $(1-\epsilon)$-optimal social welfare in expectation.

Let $\vec{l_i}$ denote a solution of (1) that accepts only the ith bid and rejects all the other bids, *i.e.*, $l_i^i = 1$ and $l_i^{i'} = 0, \forall i' \neq i$. We can easily see that $\vec{l_i}, \forall i \in L$, are feasible solutions to (1). Note that $\vec{x}^p$ is a feasible solution to (1) as well, since the constraints in ILP (3) and ILP (1) are the same. The set of feasible solutions to sample from hence is $\{\vec{x}^p, \vec{l_1}, \ldots, \vec{l_L}, \vec{0}\}$, where $\vec{0}$ is a L-dimensional all-zero vector. The final allocation

solution of (1), denoted by $\vec{y}^{\epsilon}$, is randomly produced following the distribution $\Omega(\vec{x}^p)$ below:

$$\Omega(\vec{x}^p) = \begin{cases} Pr[\vec{y}^{\epsilon} = \vec{x}^p] = 1-\epsilon, \\ Pr[\vec{y}^{\epsilon} = \vec{l}_i] = \frac{\sum_{j=1}^{L} \theta_j x_j^p}{L}, \forall i \in \{1,...,L\}, \\ Pr[\vec{y}^{\epsilon} = \vec{0}] = 1 - Pr[\vec{y}^{\epsilon} = \vec{x}^p] - \sum_{i=1}^{L} Pr[\vec{y}^{\epsilon} = \vec{l}_i]. \end{cases} \tag{5}$$

We can verify that the probability of each candidate is positive and with a summation exactly 1. We then have that the expectation of $\vec{y}^{\epsilon}$ is

$$E[\vec{y}^{\epsilon}] = (1-\epsilon)\vec{x}^p + (\frac{\sum_{j=1}^{L} \theta_j x_j^p}{L})(\sum_{i=1}^{L} \vec{l}_i) = P^T \vec{x}^p = \vec{x}^f. \tag{6}$$

Given the above, the design of all the candidates in $\Omega(\vec{p})$ and probability assignment of each of them aim to make the expectation equal to $P^T\vec{x}^p$. The high level idea is using $\Omega(\vec{x}^p)$ to randomly perturb $\vec{x}^p$ to $\vec{x}^f$ where P^T of $\vec{x}^p$ compensates the perturbation P of $\vec{b}$ in (2). According to the critical property $(P\vec{b})^T\vec{x}^p = \vec{b}^T(P^T\vec{x}^p)$, i.e., the perturbation of the objective function is equal to the perturbation of the solution, $\vec{x}^f$ enables the $(1-\epsilon)$-approximation, as indicated in (4). Here, $\vec{y}^{\epsilon}$ is equal to x^p with a high probability of $1-\epsilon$, which is in accordance with the $1-\epsilon$ part in (2). Each of the base vectors $\vec{l}_i$ and the zero vector $\vec{0}$ is chosen as a candidate to make $\Omega(\vec{x}^p)$ diffuse enough, such that an expected polynomial number of Pareto optimal solutions to (3) can be guaranteed, which will be proved in Theorem 3.

We summarize the above steps in Alg. 2, which is our randomized algorithm for computing a $(1-\epsilon)$-approximate solution to social welfare optimization problem (1).

Algorithm 2: The $(1-\epsilon)$-Approx. Algorithm for ILP (1)

1 **Input:** $\epsilon \in (0,1), \vec{b}, \vec{R}, \vec{c}$
2 **Output:** $(1-\epsilon)$-approximate allocation solution $\vec{y}^{\epsilon}$
3 Choose $\theta_1, ..., \theta_L$ independently and identically in the interval $[0, \frac{\epsilon}{L}]$;
4 Produce perturbation matrix: $P = (1-\epsilon)I + \frac{\vec{\theta}\vec{1}^T}{L}$;
5 Compute $\vec{x}^p =$ Algorithm $1(P\vec{b}, \vec{R}, \vec{c})$;
6 Produce distribution $\Omega(\vec{x}^p)$ according to (5);
7 **return** A sample $\vec{y}^{\epsilon}$ according to $\Omega(\vec{x}^p)$ in (5).

Alg. 2 achieves the following properties.

(i) The expected running time of the randomized Alg. 2 is polynomial. Although the worst-case computation complexity of the exact Algorithm Alg. 1 is exponential due to exponentially many Pareto optimal solutions in the worst case (Theorem 2), we show that the algorithm runs efficiently in practice, based on smoothed analysis [16][21]. The reason is that the expected number of the Pareto optimal solutions of the perturbed social welfare maximization problem (3) is polynomial, and hence the exact algorithm runs in polynomial time in expectation when applied to the perturbed problem—perturbed with a P generated randomly as in (2). According to smoothed analysis, Alg. 1 is said to run in smoothed polynomial time.

Theorem 3. *The expected number of Pareto optimal solutions of the perturbed social welfare maximization problem (3) is upper bounded by $1 + \frac{L^4}{\epsilon}$, where the perturbation matrix P is produced according to (2) with $\{\theta_1, \theta_2, \ldots, \theta_L\}$ independently and identically chosen from $[0, \frac{\epsilon}{L}]$.*

The proof is given in our technical report [25].

Theorem 4. *The expected running time of the randomized algorithm Alg. 2 is polynomial.*

The proof is given in our technical report [25].

(ii) Alg. 2 achieves $(1-\epsilon)$-optimal social welfare.

Theorem 5. *Alg. 2 is a $(1-\epsilon)$-approximation randomized algorithm for the social welfare maximization problem (1).*

Proof. We have shown $E[\vec{y}^{\epsilon}] = \vec{x}^f$. Hence $E[\vec{b}^T\vec{y}^{\epsilon}] = \vec{b}^T\vec{x}^f = \vec{b}^T(P^T\vec{x}^p) = POPT \geq (1-\epsilon)OPT$, based on (4). □

C. The Truthful-in-Expectation VM Auction

Recall the MIDR mechanism [26] we introduced at the beginning of this section. We have designed the randomized allocation algorithm which chooses an allocation solution following the distribution $\Omega(\vec{x}^p)$ in (5). This distribution is independent of the cloud users' bids. We next show that it leads to the largest expected social welfare among a compact set of such distributions, such that we can combine a randomized VCG payment scheme following a similar distribution, to obtain an auction mechanism that is truthful in expectation.

Theorem 6. *The randomized allocation Alg. 2 is an MIDR allocation rule for the social welfare maximization problem (1).*

Proof. An MIDR allocation rule of a social welfare maximization problem returns an allocation solution that is sampled randomly from a distribution over a feasible set of the problem, which achieves the largest expected social welfare, among random solutions produced following distributions in a distributional range, which is a fixed compact set of probability distributions over the feasible set that are independent of the users' bids [26].

Let $\mathcal{T}$ denote the set of all feasible solutions of ILP (1). For each feasible solution $\vec{x} \in \mathcal{T}$, we can obtain a distribution $\Omega(\vec{x})$ in the same way as the distribution in (5) by replacing all the $\vec{x}^p$ with $\vec{x}$ shown in $\Omega(\vec{x}^p)$. Then let $\vec{y}$ denote the random allocation solution produced following $\Omega(\vec{x})$, i.e., $\vec{y} \sim \Omega(\vec{x})$. Given ϵ and $\theta_1, \ldots, \theta_L$, the distribution $\Omega(\vec{x})$ is dependent on feasible solution $\vec{x}$, but independent of the users' bids. $\mathcal{R} = \{\Omega(\vec{x}), \forall \vec{x} \in \mathcal{T}\}$ is a compact set including all the distributions indexed by feasible solution $\vec{x}$. Using $\mathcal{R}$ as the distributional range, we have

$$\begin{aligned} E_{\vec{y}^{\epsilon} \sim \Omega(\vec{x}^p)}[\vec{b}^T\vec{y}^{\epsilon}] &= \vec{b}^T(P^T\vec{x}^p) = \max_{\vec{x} \in \mathcal{T}} \vec{b}^T(P^T\vec{x}) \\ &= \max_{\vec{x} \in \mathcal{T}} E_{\vec{y} \sim \Omega(\vec{x})}[\vec{b}^T\vec{y}]. \end{aligned} \tag{7}$$

The first equation is due to (6). The second equation is because $\vec{x}^p$ is the optimal solution of the perturbed ILP (3), and the set of feasible solutions of ILP (1) and ILP (3) are the same. The third equation is due to $E_{\vec{y}\sim\Omega(\vec{x})}[\vec{y}] = P^T\vec{x}$, which can be readily obtained according to (6). Hence the solution $\vec{y}^\epsilon$ selected following distribution $\Omega(\vec{x}^p)$ in Alg. 2 achieves the largest expected social welfare, among all the solutions produced following distributions in the distributional range $\mathcal{R}$, leading to an MIDR allocation rule. □

Now we describe our randomized VCG payment, which is based on the allocation solution $\vec{y}^\epsilon$, as follows:

$$p_i(\vec{y}^\epsilon) = \vec{b}^T_{-i}\vec{y}^\epsilon_{-i} - (\vec{b}^T\vec{y}^\epsilon - b_i y^\epsilon_i), \forall i \in [L]. \tag{8}$$

Here $\vec{b}_{-i}$ denotes the bidding price vector where the ith bidding price is set to 0; $\vec{y}^\epsilon_{-i}$ is the random allocation solution output by Alg. 2 with the input bidding price vector $\vec{b}_{-i}$. Hence $\vec{b}^T_{-i}\vec{y}^\epsilon_{-i}$ is the social welfare when the ith bid is excluded from the auction. Further recall $\vec{y}^\epsilon$ is the output of Alg. 2 with the full bidding price vector $\vec{b}$. Let y^ϵ_i be the ith element of $\vec{y}^\epsilon$. Hence $\vec{b}^T\vec{y}^\epsilon - b_i y^\epsilon_i$ is the social welfare achieved by all the other bids except bid i, when all the bids are considered in the auction.

Theorem 7. *The randomized allocation algorithm Alg. 2 combined with the randomized VCG payment in (8) yields an auction mechanism that runs in polynomial time in expectation, is truthful in expectation, and achieves* $(1-\epsilon)$*-optimal social welfare in expectation.*

Proof. According to the principles of MIDR algorithms [26], to render a truthful-in-expectation mechanism, we should combine an MIDR allocation rule with a VCG-like payment as follows:

$$p'_i = E[\vec{b}^T_{-i}\vec{y}^\epsilon_{-i} - (\vec{b}^T\vec{y}^\epsilon - b_i y^\epsilon_i)], \tag{9}$$

where the expectation is computed as follows in more details: $E_{\vec{y}^\epsilon_{-i}\sim\Omega(\vec{x}^p_{-i})}[\vec{b}^T_{-i}\vec{y}^\epsilon_{-i}] - E_{\vec{y}^\epsilon\sim\Omega(\vec{x}^p)}[\vec{b}^T\vec{y}^\epsilon - b_i y^\epsilon_i]$. Here $\vec{x}^p_{-i}$ is the optimal solution of the perturbed ILP (3), produced by line 5 of Alg. 2, when the input bidding price vector is $\vec{b}_{-i}$.

Since it may not be always possible to compute the expectation in (9) efficiently, it has been proved [24] that instead of using (9), we can use a randomized payment rule to yield the truthfulness in expectation as well, as long as the expected payment of the randomized payment rule equals p'_i in (9).

The expectation of our random payment in (8) is exactly

$$\begin{aligned} & E[p_i(\vec{y}^\epsilon)] \\ = \; & E_{\vec{y}^\epsilon_{-i}\sim\Omega(\vec{x}^p_{-i})}[\vec{b}^T_{-i}\vec{y}^\epsilon_{-i}] - E_{\vec{y}^\epsilon\sim\Omega(\vec{x}^p)}[\vec{b}^T\vec{y}^\epsilon - b_i y^\epsilon_i] \\ = \; & p'_i. \end{aligned}$$

Hence the random payment in (8) renders truthfulness in expectation and can be computed in polynomial time in expectation. Combining Theorem 4 and Theorem 5, this theorem is proved. □

We summarize our complete randomized auction mechanism in Alg. 3.

Algorithm 3: The Randomized Auction Mechanism

1 **Input:** $\epsilon \in (0,1), \vec{b}, \vec{R}, \vec{c}$
2 **Output:** allocation solution $\vec{y}^\epsilon$ and payment $\vec{p}$
3 Compute $\vec{y}^\epsilon$=Algorithm 2$(\epsilon, \vec{b}, \vec{R}, \vec{c})$;
4 Compute payment $p_i(\vec{y}^\epsilon) = \vec{b}^T_{-i}\vec{y}^\epsilon_{-i} - (\vec{b}^T\vec{y}^\epsilon - b_i y^\epsilon_i)$, for all accepted bids $i \in [L]$;
5 **return** $\vec{y}^\epsilon$ and $\vec{p}$

V. Performance Evaluation

We evaluate our randomized auction using trace-driven simulations, exploiting Google cluster-usage data [28] which record jobs submitted to the Google cluster. Each job contains multiple tasks, with information on resource demands (CPU, RAM, Disk) of the tasks. We translate each job into a bundle bid, and each task in the job into a VM in the bundle. There are 100 types of VMs consisting of $K = 3$ types of resources in our experiments. Each task is mapped to a VM of a specific type by resource demands, and further mapped to a data centre randomly. We then generate the VM demands of the bundle, $q^i_{m,d}$, by counting the number of VMs of the same type mapped to the same data center among the tasks in the job. We estimate the unit prices of CPU, Disk and RAM, respectively, based on the prices of Amazon EC2 instances and their resource composition, and then set the bid price of each bundle based on the unit prices and the overall resource demands in the bundle, scaled by a random number in $[0.75, 1.5]$. In this way, we obtain a pool of bidding bundles from the Google cluster data. Each user randomly chooses at most $|\mathcal{B}_n|$ bundles from the pool to bid. We compute the capacity of type-k resource, $c_{d,k}$, in a data center based on the overall amount of this resource required in this data center in all the bid bundles submitted by the users, and scale it down using a random factor in [0, 0.5N/L], such that roughly no more than half of the users can win a bid under constraint (1b), without loss of generality. By default, the number of users is $N = 500$, the upper-bound on the number of bids a user can submit is $|\mathcal{B}_n| = 4$, the number of data centers is $D = 8$, and $\epsilon = 0.05$. We repeat each experiment for 50 times to obtain the average results.

A. Approximation Ratio

We first study the average approximation ratio achieved by our algorithm, computed by the social welfare achieved by Alg. 2 over the optimal social welfare by solving (1) exactly. Given $\epsilon = 0.05$, the theoretical expected approximation ratio is 0.95. Fig. 1 shows that the average approximation ratio is larger than the theoretical ratio and approaches the latter when the number of users (or bids) increases. According to (4) and (6) in Sec. IV, $1-\epsilon$ is a lower bound of the approximation ratio, and the results show that under practical settings the average ratio achieved is better. The reason why the ratio approaches the theoretical one when N is large can also be explained by (4) in Sec. IV, that the inequality tends to equality when N (and hence L) is large. The average ratio does

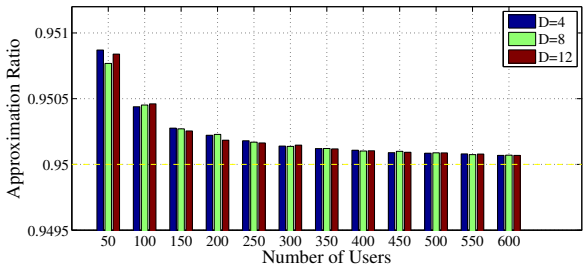

Fig. 1: Approx. Ratio of RPAA

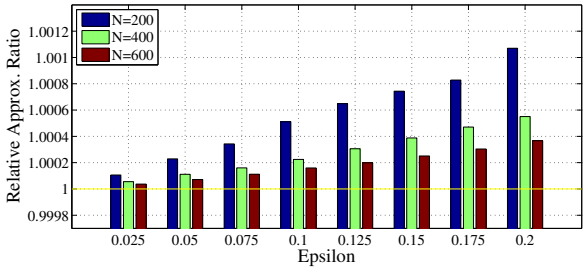

Fig. 2: Relative Approx. Ratio of RPAA

not vary much with different numbers of data centers, since the perturbation does not influence constraints (1a) where the number of data centers shows up.

Fig. 2 compares the average approximation ratio obtained in our experiments and the respective theoretical approximation ratios at different values of ϵ. We plot the relative approximation ratio, computed as $\frac{\text{average approx. ratio}}{1-\epsilon}$. The average approximation ratio we obtained outperforms the theoretical ratio a lot more when ϵ is larger, since $\theta_i, i = 1, \ldots, L$ are larger (they are selected in the interval $[0, \epsilon/L]$) and the gap is larger according to (4) in Sec. IV. The better performance at a smaller N can be explained similarly as that for Fig. 1.

B. Social Welfare Comparison with PDAA

We now compare the social welfare achieved by our Alg. 2 with the primal-dual approximation algorithm in [14] (which is essentially the algorithm used in [15] as well). The algorithm in [14] does not consider the distribution of VM demands in multiple data centers. We hence extend this algorithm to multiple data centers by expanding the dimensions of the capacity constraint from K to $K \times D$ to handle K types of resources distributed in D data centers, for a fair comparison. In the following figures, we use RPAA to represent our Randomized Perturbation-based Approximation Algorithm and PDAA to represent the Primal-Dual Approximation Algorithm in [14].

Fig. 3 and Fig. 4 show that our algorithm consistently outperforms the algorithm in [14] in terms of social welfare, under the same parameter settings. This validates our theoretical analysis: our algorithm is guaranteed to achieve a no-lower-than-$(1-\epsilon)$ approximation in social welfare, where the other algorithm achieves a ratio of around 2.72 [14].

Fig. 3 also indicates that the social welfare of both algorithms increases with the increase of N and $|\mathcal{B}_n|$. The resource capacity in our experiments is set to be roughly linear in the total resource demand of the users, and when N is large, more bids can be accepted and hence the social welfare is larger. When each user can submit more bids, the decision space for ILP (1) is larger, leading to a better social welfare.

Fig. 4 implies a negative correlation between the social welfare and D, which can be intuitively explained as follows: When the number of data centers is larger, the bid bundles that each user submits contain VMs scattered in more data centers. Had any resource demand in any data center not be satisfied, a bid will be rejected. Thus the chance of each bundle to be accepted decades, leading to a slight decrease of the social welfare.

C. User Satisfaction Comparison with PDAA

We next evaluate user satisfaction achieved by both algorithms, which is the percentage of users accepted as winners in the respective auctions. Fig. 5 and Fig. 6 show that user satisfaction achieved by our algorithm is about twice that of the other algorithm, which results from similar reasons as given in the comparison of social welfare. User satisfaction of both algorithms improves slightly with the increase of the number of bids a user submits, mainly because more choices of the bids provide a user a higher chance to win one, while the chance does not improve much since all the users now have more bids to submit. User satisfaction in Fig. 6 decreases slightly as more data centers are included, suffering from the same cause as explained for Fig. 4.

Finally, we remark on the running time incurred by the two algorithms (figures omitted due to space limit). Figs. 3–6 illustrate that our algorithm outperforms the algorithm in [14] in social welfare and user satisfaction, which is mainly due to the much better approximation ratio achieved by our algorithm. The running time of PDAA is shown to be $O(L^6 \log L)$, where L is the number of bids, while our algorithm has an expected time complexity of $O(KDL^9(\frac{1}{\epsilon})^2)$ (details in the proof of Theorem 4 in [25]). Hence, our algorithm may sacrifice some of the computation efficiency for a much better approximation to the optimal social welfare. However, the difference is not substantial, and a polynomial running time is still guaranteed for our algorithm in practice.

VI. Concluding Remarks

This work presents a truthful and efficient auction mechanism for dynamic VM provisioning and pricing in geo-distributed cloud data centers. By employing smoothed analysis in a novel way and randomized reduction techniques, we develop a randomized mechanism that achieves truthfulness, polynomial running time, and $(1-\epsilon)$-optimal social welfare for resource allocation (all in expectation). We propose an exact algorithm which solves the NP-hard social welfare maximization problem in expected polynomial time, and apply a perturbation-based randomized scheme based on the exact algorithm to produce a VM provisioning solution that is $(1-\epsilon)$-optimal in social welfare in expectation. Combining the randomized scheme with a randomized VCG payment, we achieve an auction mechanism truthful in expectation. From a theoretical perspective, we achieve a randomized

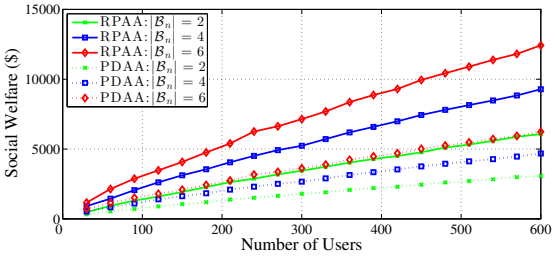

Fig. 3: Social Welfare of RPAA and PDAA

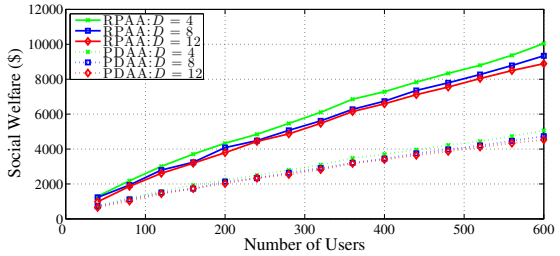

Fig. 4: Social Welfare of RPAA and PDAA

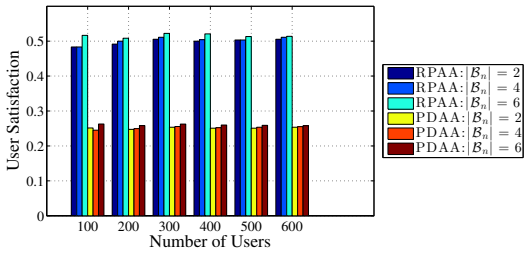

Fig. 5: User Satisfaction of RPAA and PDAA

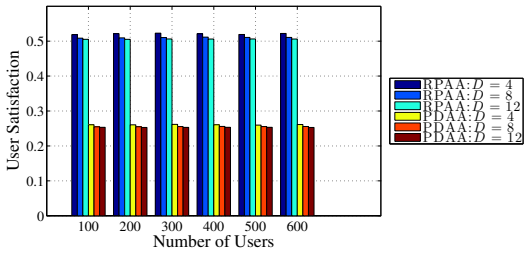

Fig. 6: User Satisfaction of RPAA and PDAA

fully polynomial-time-in-expectation $(1-\epsilon)$-approximation scheme for a strongly NP-hard problem which does not have a deterministic FPTAS. We believe that this new technique can be generalized to work for a rich class of combinatorial auctions, other than VM auctions. Trace driven simulations we conduct validate our theoretical analysis and reveals the superior performance of our mechanism as compared to an existing mechanism on dynamic VM provisioning.

REFERENCES

[1] "Amazon EC2 Instance Types," http://aws.amazon.com/ec2/instance-types/.

[2] "Recommended Memory Configurations for the MapReduce Service," https://ambari.apache.org/1.2.0/installing-hadoop-using-ambari/content/ambari-chap3-7-9a.html.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," in *Proc. of ACM SOSP*, 2003.

[4] "Amazon EC2 Spot Instances," http://aws.amazon.com/ec2/spot-instances/.

[5] Q. Wang, K. Ren, and X. Meng, "When Cloud Meets eBay: Towards Effective Pricing for Cloud Computing," in *Proc. of IEEE INFOCOM*, 2012.

[6] W. Wang, B. Liang, and B. Li, "Revenue Maximization with Dynamic Auctions in IaaS Cloud Markets," in *Proc. of IEEE/ACM IWQoS*, 2013.

[7] S. Zaman and D. Grosu, "Combinatorial Auction-Based Dynamic VM Provisioning and Allocation in Clouds," in *Proc. of IEEE CloudCom*, 2011.

[8] ——, "Combinatorial Auction-based Allocation of Virtual Machine Instances in Clouds," *Journal of Parallel and Distributed Computing*, vol. 73, no. 4, pp. 495–508, 2013.

[9] H. Zhang, B. Li, H. Jiang, F. Liu, A. V. Vasilakos, and J. Liu, "A Framework for Truthful Online Auctions in Cloud Computing with Heterogeneous User Demands," in *Proc. of IEEE INFOCOM*, 2013.

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

[11] A. Mu'alem and N. Nisan, "Truthful Approximation Mechanisms for Restricted Combinatorial Auctions," *Games and Economic Behavior*, vol. 64, no. 2, pp. 612–631, 2008.

[12] D. Lehmann, L. I. Oćallaghan, and Y. Shoham, "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM*, vol. 49, no. 5, pp. 577–602, 2002.

[13] R. Lavi and C. Swamy, "Truthful and near-optimal mechanism design via linear programming," in *Proc. of IEEE FOCS*, 2005, pp. 595–604.

[14] L. Zhang, Z. Li, and C. Wu, "Dynamic Resource Provisioning in Cloud Computing: A Randomized Auction Approach," in *Proc. of IEEE INFOCOM*, 2014.

[15] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," in *Proc. of ACM SIGMETRICS*, 2014.

[16] D. Spielman and S.-H. Teng, "Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time," in *Proc. of ACM STOC*, 2001.

[17] G. Gens and E. Levner, "Complexity of Approximation Algorithms for Combinatorial Problems: A Survey," *ACM SIGACT News*, vol. 12, no. 3, pp. 52–65, 1980.

[18] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[19] M. Alicherry and T. V. Lakshman, "Network Aware Resource Allocation in Distributed Clouds," in *Proc. of IEEE INFOCOM*, 2012.

[20] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multi-Resource Allocation: Fairness-Efficiency Tradeoffs in a Unifying Framework," in *Proc. of IEEE INFOCOM*, 2012.

[21] H. Röglin and S.-H. Teng, "Smoothed Analysis of Multiobjective Optimization," in *Proc. of IEEE FOCS*, 2009.

[22] D. Arthur and S. Vassilvitskii, "Worst-case and Smoothed Analysis of the ICP Algorithm, with an Application to the k-means Method," in *Proc. of IEEE FOCS*, 2006.

[23] X. Chen, X. Deng, and S.-H. Teng, "Computing Nash Equilibria: Approximation and Smoothed Complexity," in *Proc. of IEEE FOCS*, 2006.

[24] S. Dughmi and T. Roughgarden, "Black-Box Randomized Reductions in Algorithmic Mechanism Design," in *Proc. of IEEE FOCS*, 2010.

[25] "A Truthful Near-Optimal Mechanism for On-demand Cloud Resource Provisioning," Tech. Rep., `https://www.dropbox.com/s/58ngejldyblqeu6/TechReport.pdf` 2014.

[26] S. Dobzinski and S. Dughmi, "On the Power of Randomization in Algorithmic Mechanism Design," in *Proc. of IEEE FOCS*, 2009.

[27] G. L. Nemhauser and Z. Ullmann, "Discrete Dynamic Pprogramming and Capital Allocation," *Management Science*, vol. 15, no. 9, pp. 494–505, 1969.

[28] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2012.03.20. Posted at URL http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.