

A Truthful $(1 - \epsilon)$ -Optimal Mechanism for On-demand Cloud Resource Provisioning

Xiaoxi Zhang¹, *Student Member, IEEE, ACM*, Chuan Wu¹, *Member, IEEE, ACM*,
Zongpeng Li², *Member, IEEE, ACM*, and Francis C.M. Lau¹, *Senior Member, IEEE*

¹Department of Computer Science, The University of Hong Kong, Hong Kong

²Department of Computer Science, University of Calgary, Calgary, Canada

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.

Index Terms—Cloud Computing; Auction; Resource Allocation; Pricing; Truthful Mechanisms

I. INTRODUCTION

Cloud computing services have been proliferating in today's Internet for the past decade. They create a shift in resource provisioning from on-premise hardware to shared resource pools accessible over the Internet. To be flexible at meeting users' resource demands, leading cloud platforms such as Amazon EC2 [1], Microsoft Azure [2] and GoGrid [3] exploit advanced virtualization technologies to pack resources (CPU, RAM, and Disk Storage) 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 9 categories and 39 types [4]. However, the increased variety on the provider's side still often falls short of addressing user needs precisely, which could lead to a waste of resources and an unjustifiably inflated payment by the users. For example, suppose a user needs to run a computationally intensive job (e.g., a MapReduce job) by acquiring 16 vCPU units and 16 GB memory [5] in EC2's Singapore data center, to process 160 GB usage data. The best offer Amazon EC2 can make is a `c3.4xlarge` instance, which unfortunately is far from a perfect match, leading to 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 [6]). What is not ready, however, is an effective pricing mechanism to charge for those customized VMs on the spot. The current representative pricing 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 ever-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 instances market [7] represents the first attempt at a more market-driven pricing system, which, however, comes without any guarantee of truthfulness or SLA [8][9]. Some recent work further studied auction mechanism design for cloud resource provisioning from different perspectives [9][10][11]. However, most of them model VMs as type-oblivious commodities, and therefore fail to provision dynamically assembled VMs properly.

Aside from pricing, the challenge of packing available resources 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 [12], 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 [13]. To achieve truthfulness with an approximation algorithm, researchers have exploited the concept of critical bids [14], or resorted to some LP decomposition techniques [15][16][17]. The approximation ratios of these auctions with respect to social welfare optimality depend on the efficiency of the approximation algorithm employed, which is typically much larger than 1 [16][17].

This work aims to leverage the state-of-the-art techniques

in smoothed analysis [18][19] and randomized reduction, to design a highly efficient randomized auction mechanism for the provisioning and pricing of 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 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 achieving $(1-\epsilon)$ -optimal 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. Detailed steps are as followed.

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 and the packing constraints 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 within 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.

Next, 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.

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

Our mechanism design yields some interesting results: (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 [20]. (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 strong properties above guaranteed by our VM auction are made possible by unleashing the power of randomization,

through the art of calculated random perturbation (for computational efficiency) and associated perturbation (for truthfulness). While there exists 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 of its kind that applies the same carefully prepared randomization scheme twice in subtly different forms and in a coordinated fashion to achieve polynomial algorithm complexity and truthful mechanism design in the same auction framework. We believe that this new technique can be generalized to be 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.* [21] aim at minimizing the energy consumption in computing task scheduling. Alicherry *et al.* [22] study VM allocation in distributed cloud systems, taking into consideration the communication cost. Joe-Wong *et al.* [23] 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.* [10] design a truthful auction based on an approximation algorithm for resource allocation, but without proving the performance of the resource allocation algorithm. Zaman *et al.* [11] also presents an auction-based VM allocation but focuses on static resource provisioning and only guarantees a large approximation ratio. Wang *et al.* [8] 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 an approximation ratio which depends on the number of VMs. Zhang *et al.* [24] and Wang *et al.* [9] 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.* [16] and Shi *et al.* [17] 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. Mashayekhy *et al.* [25] also consider heterogeneous resources in a cloud auction. Their proposed allocation rule is proved to be a PTAS, which finds a partial allocation solution first and then allocates through dynamic programming the remaining resources to unprovisioned users. Unfortunately, the running time of their PTAS mechanism can be exponential in $1/\epsilon$, where ϵ denotes the approximation error. Shi *et al.* [26] design online VM auctions via a pricing-curve-based method. However, the resources which have been

allocated and paid by the users could be cancelled by the cloud provider in their problem model, which is not practical and user-friendly. One of the latest VM auctions proposed by Zhang *et al.* [27] investigates social welfare and profit maximization in online VM auctions, where the future demand of each type of resource could be reserved in a customized amount and duration. In their work, the performance of the mechanisms relies on an assumption that each user's demand for each resource is extremely small compared to the corresponding capacity. Unlike theirs, we study the auction in an offline version where the small bid assumption is much milder than theirs. Additionally, this work departs from the existing literature by applying smoothed analysis and randomized reduction techniques to design a randomized auction, which achieves 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 [18]. 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 [18]. Smoothed analysis has been applied recently in areas such as combinatorial programming [28], computational geometry [29], game analysis [30]. Dughmi *et al.* [31] 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 Auction

We consider an IaaS cloud system providing a pool of geo-distributed resources (*e.g.*, CPU, RAM and Disk storage) through servers deployed in multiple data centers. The different types of resources are packed into heterogeneous VMs for lease to cloud users. Suppose there are D data centers in the cloud and M types of VMs in total can be assembled to offer (which can be potentially a very large number)¹ composed of K types of resources. Let $[X]$ denote the set of integers $\{1, 2, \dots, X\}$, d index the data centers in the cloud system and m index the types of VM. Each VM of type $m \in [M]$ consumes an r_m^k amount of type- k resource, for all $k \in [K]$.

¹Note that we allow flexible VM assembly on demand and the numbering of VM types is purely for the ease of presentation.

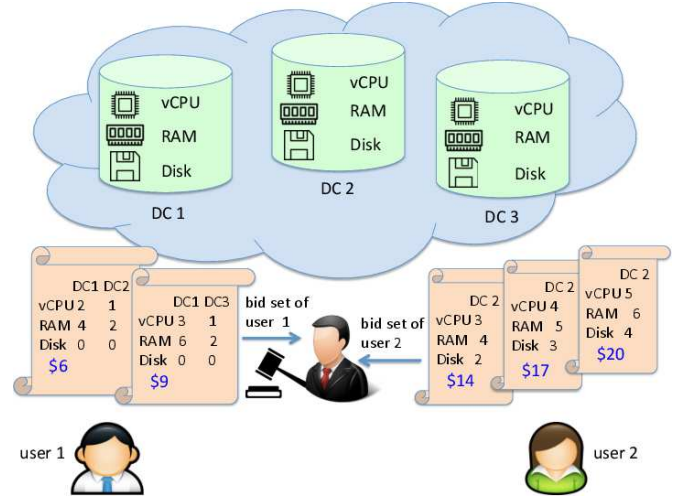


Fig. 1: An illustration of the VM auction.

Each data center $d \in [D]$ has a capacity c_{dk} for each resource of type $k \in [K]$.

The cloud provider acts as an auctioneer and sells custom-made VMs to W cloud users through auctions. The cloud users request resources in the form of VMs through bidding in the auction. Let N be the total number of bids submitted by all the cloud users. We use $i \in [N]$ to index the bids. Each cloud user $w \in [W]$ is allowed to 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}_w denote the set of bids submitted by user w . Each bid B_i contains a list of requested VMs of different types and location preferences, *i.e.*, q_{md}^i VMs of type- m in data center d , $\forall m \in [M], d \in [D]$, and a bidding price b_i , *i.e.*, the reported valuation of the resource combination required in B_i . More specifically, each bid $i \in [N]$ can be formulated as follows:

$$B_i = \{b_i, \{q_{md}^i\}_{m \in [M], d \in [D]}\}.$$

For ease of problem formulation, we use $R_i^{kd} = \sum_{m=1}^M q_{md}^i r_m^k$ to denote the overall amount of type- k resource required by bid i in data center d .

Upon receiving user bids, the cloud provider computes the outcome of the auction, including (i) the resource allocation scheme, $\vec{x} = \{x_1, \dots, x_N\}$, 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 . An illustration of the system is given in Fig. 1.

We summarize important notations in Table I for ease of reference.

TABLE I: Notation

W	# of users	N	# of bids
M	# of VM types	K	# of resource types
D	# of data centers	B_i	the i th bid
v_i	true valuation of bid i	u_i	utility of bid i
P	perturbation matrix	\mathcal{B}_w	bid set of user w
b_i	bidding price of bid i	\hat{b}_i	perturbed b_i
ϵ	parameter in $(0, 1)$		
r_m^k	amount of type- k resource in a type- m VM		
q_{md}^i	# of type- m VMs in DC d requested in bid i		
R_i^{kd}	demand of type- k resource in DC d in bid i		
\hat{R}_i^{kd}	perturbed R_i^{kd}		
c_{kd}	capacity of type- k resource in DC d		
$s(\vec{x})$	social welfare under allocation solution \vec{x}		
$C_{kd}(\vec{x})$	demand for type- k resource in DC d under \vec{x}		
θ_i^j	parameter in $[0, \epsilon/N]$		
$\Omega(\vec{x})$	distribution based on \vec{x} , ϵ and θ		
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 (4)		
\vec{x}^f	fractional solution perturbed from \vec{x}^p		
\vec{y}^ϵ	auction's final allocation solution		
$p_i(\vec{y}^\epsilon)$	payment of bid i under \vec{y}^ϵ		

B. Goals of Mechanism Design

We pursue the following 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, $\sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} p_i x_i$, and the aggregate users' utility $\sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} (v_i - p_i) x_i$. 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 $\sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} v_i x_i$, which equals $\sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} b_i x_i$ 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 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.

$$\text{maximize } \sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} b_i x_i \quad (1)$$

subject to:

$$\sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} x_i R_i^{kd} \leq c_{kd}, \quad \forall k \in [K], \forall d \in [D], \quad (1a)$$

$$\sum_{i \in \mathcal{B}_w} x_i \leq 1, \quad \forall w \in [W], \quad (1b)$$

$$x_i \in \{0, 1\}, \quad \forall i \in \mathcal{B}_w, \forall w \in [W]. \quad (1c)$$

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 Appendix A.

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 [32]. An MIDR algorithm is 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, e.g., the set of distributions over all the feasible solutions.² If we can design an MIDR allocation rule, then we can combine a randomized VCG payment scheme following a similar distribution to obtain an auction mechanism that is truthful in expectation [32]. To achieve the other two goals of our auction design, the allocation algorithm should be $(1 - \epsilon)$ -optimal in social welfare and have polynomial running time in expectation.

We next establish the randomized allocation algorithm in two steps. First, we design an exact algorithm based on dynamic programming for solving the social welfare maximization problem in Sec. IV-A. Next, we design the randomized allocation algorithm based on a perturbation framework, by running the exact algorithm on a randomized perturbed version of the original maximization problem and sampling the final allocation solution from a distribution. The randomized sampling compensates the perturbation on the problem, done

²To achieve truthfulness of VCG-based mechanisms, there are no additional restrictions on the distributional range or the distributions within the range (e.g., the size or specific form) in such MIDR allocation algorithms. In fact, a well-designed range of distributions, as what we will propose, is the key for a better approximation ratio.

before running the exact algorithm, by transforming the optimal solution of the perturbed problem into an near-optimal-in-expectation solution to the original problem. As the core of the randomized allocation algorithm, the perturbation rule is carefully designed, in order to lead to a $(1 - \epsilon)$ approximation ratio, as well as polynomial running time of the algorithm, both in expectation.

We further describe the payment scheme in Sec. IV-C.

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 $\vec{x} = \{x_i, \forall i \in \mathcal{B}_w, w \in [W]\}$ 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 [33].

Let $s(\vec{x}) = \sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} b_i x_i$ denote the social welfare under allocation solution \vec{x} , and $C_{kd}(\vec{x}) = \sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} x_i R_i^{kd}$ 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 $s(\vec{x}') \geq s(\vec{x})$ and $C_{kd}(\vec{x}') \leq C_{kd}(\vec{x}), \forall k \in [K], \forall d \in [D]$, with at least one inequality being strict among the above, as well as $\sum_{i \in \mathcal{B}_w} x'_i \leq 1, \forall w \in [W]$.

We identify 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 $[N]$ (the bids in $[N]$ 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 $\mathcal{P}(N)$ which is the set of Pareto optimal solutions of ILP (1).

We show the following property of the Pareto optimal solution sets, with proof given in Appendix B.

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 $\mathcal{P}(i-1)$, $\forall i = 2, \dots, N$.*

Let $\mathcal{P}(i-1) + 1$ denote the set of i -dimensional solutions obtained by simply adding 1 as the i th 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 i th 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

Algorithm 1: The Exact Algorithm for ILP (1)

```

1 Input:  $\vec{b}, \vec{R}, \vec{c}$ 
2 Output: exact optimal solution  $\vec{x}$ 
3 if  $C_{kd}(\{1\}) \leq c_{kd}, \forall k \in [K], \forall d \in [D]$  then
4    $\mathcal{P}(1) = \{0, 1\}$ ;
5 else
6    $\mathcal{P}(1) = \{0\}$ ;
7 for  $i = 2, \dots, N$  do
8   for all  $\vec{x}^{(i-1)} \in \mathcal{P}(i-1)$  do
9      $\vec{x}^{(i)} = \{\vec{x}^{(i-1)}, 1\}$ ;
10    if  $\vec{x}^{(i)}$  satisfies Constraints (1a) and (1b) then
11       $\mathcal{P}(i-1) + 1$ ;
12    Merge  $\mathcal{P}(i-1) + 0$  and  $\mathcal{P}(i-1) + 1$  into  $\mathcal{P}(i)'$ ;
13    Prune the solutions dominated by others in  $\mathcal{P}(i)'$  to
    obtain  $\mathcal{P}(i) = \{\vec{x}^{(i)} \in \mathcal{P}(i)' | \nexists \vec{x}^{(i)'} \in \mathcal{P}(i)' : \vec{x}^{(i)'} \text{ dominates } \vec{x}^{(i)}\}$ ;
14 return  $\vec{x} = \operatorname{argmax}_{\vec{y} \in \mathcal{P}(N)} s(\vec{y})$ 

```

capacity limits, and contains only one Pareto optimal solution 0, otherwise. Then we construct $\mathcal{P}(i), i = 2, \dots, N$, 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 $\mathcal{P}(N)$ that achieves the maximum social welfare.

The computation complexity of the exact algorithm in Alg. 1 is polynomial in the number of Pareto optimal solutions in $\mathcal{P}(N)$, 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., $|\mathcal{P}(1)| \leq \dots \leq |\mathcal{P}(N)|$.*

The proof is given in Appendix C.

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

The proof is given in Appendix E.

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, i.e., running in polynomial time in expectation, and can be used as a building block in a perturbation framework for producing a randomized allocation algorithm.

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

We next design the randomized algorithm to solve the social welfare maximization problem in (1) in polynomial time in expectation. The basic idea 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 solution from solving the random perturbation of the social welfare maximization problem,

based on smoothed analysis techniques [18][28]. The random perturbation on the original problem is carefully designed, in close connection with the distribution to sample feasible solutions, to achieve $(1 - \epsilon)$ -optimal social welfare of (1) in expectation. Especially, the most salient feature of algorithm design in this work, as the first in the literature, is to apply a pair of associated random perturbation schemes for smoothed polynomial time algorithm design and for randomized auction design, respectively.

Algorithm design. Given an arbitrary parameter $\epsilon \in (0, 1)$ and KDN random variables $\{\theta_1^j, \theta_2^j, \dots, \theta_N^j\}_{j \in \{0, \dots, KD-1\}}$ that are independently and identically chosen from a uniform distribution on the interval of $[0, \frac{\epsilon}{N}]$. Let $\vec{\theta}^j = \{\theta_1^j, \dots, \theta_N^j\}, \forall j \in \{0, \dots, KD-1\}$. Suppose the packing constraints in (1a) are ordered. Let $j \in [KD]$ index the sorted constraints; thus we can use R_i^j to replace R_i^{kd} in (1a), $\forall j \in \{1, \dots, KD\}$, and will refer to j as a resource, which represents the corresponding resource (k) in the respective data center (d).

We perturb the bidding price b_i in the objective function and the demand R_{kd}^i of the first $K \times D - 1$ packing constraints of ILP (1) independently to:

$$\hat{b}_i = (1 - \epsilon/2)b_i + \frac{\theta_i^0 \sum_{i'=1}^N b_{i'}}{N}, \forall i \in [N] ; \quad (2)$$

$$\hat{R}_i^j = R_i^j + \frac{\theta_i^j \sum_{i'=1}^N R_{i'}^j}{N}, \forall i \in [N], j \in \{1, \dots, KD-1\} \quad (3)$$

Here, $\theta_i^0, \forall i \in [N]$, are the random variables associated with b_i 's, the coefficients in the objective function, and $\theta_i^j, \forall i \in [N]$, are associated with R_i^j 's in the j th constraint in (1a), $\forall j \in \{1, \dots, KD-1\}$. Note that the last constraint in (1a) is not perturbed. We define $\hat{R}_i^{KD} = R_i^{KD}, \forall i \in [N]$, for this unperturbed last constraint in (1a).³ The perturbed social welfare maximization problem is:

$$\text{maximize} \quad \sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} \hat{b}_i x_i \quad (4)$$

subject to:

$$\sum_{w \in [W]} \sum_{i \in \mathcal{B}_w} x_i \hat{R}_i^j \leq c_j, \forall j \in \{1, \dots, KD\}, \quad (4a)$$

(1b) and (1c)

According to the perturbation in (2), let

$$P = (1 - \epsilon/2)I + \frac{\vec{\theta}^0 \vec{1}^T}{N} \quad (5)$$

be the perturbation matrix of the objective function, where I is the $N \times N$ identity matrix. Then we can express the perturbation of bidding price vector as $\vec{\hat{b}} = P\vec{b}$. We solve the perturbed social welfare maximization problem using the exact algorithm (Alg. 1), and derive the optimal solution \vec{x}^p and

³In line with the latest smoothed analysis techniques, we adopt the semi-random model [18]. In this context, for a binary maximization problem with one objective function and multiple packing constraints, the objective function and all the packing constraints except the last one are perturbed due to the reasons explained in Appendix F.

optimum value of the perturbed objective function $POPT = \vec{\hat{b}}^T \vec{x}^p$. 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. The following lemma shows that the optimal objective value of the perturbed problem is at least $(1 - \epsilon)$ -fraction of the optimal social welfare of the original problem, which is very close as long as the perturbation, decided by ϵ , is small enough, under a small bid assumption. The proof is given in Appendix D.

Lemma 3. $POPT \geq (1 - \epsilon)OPT$, if $\max_{i \in [N], k \in [K], d \in [D]} \frac{R_{kd}^i}{c_{kd}} \leq \frac{1}{2KD(2 + \frac{1}{\epsilon})}$.

The small bid assumption stated in Lemma 3 essentially requires that the demand in each bid for each type of resource in each desired datacenter is small as compared to the corresponding resource capacity, which is easy to justify in real systems. Moreover, if a smaller ϵ is chosen, the assumption becomes stronger, *i.e.*, the ratio of the largest demand among the bids for each type of resource over the resource capacity is required to be smaller.

Lemma 3 gives that $POPT = (P\vec{b})^T \vec{x}^p = \vec{b}^T (P^T \vec{x}^p) \geq (1 - \epsilon)OPT$. We can 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 i th 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 $\vec{l}_i, \forall i \in [N]$, are feasible solutions to (1). Note that \vec{x}^p is a feasible solution to (1) as well, since all the perturbed coefficients of the packing constraints in ILP (4) are no smaller than those of ILP (1). The set of feasible solutions to sample from hence is $\{\vec{x}^p, \vec{l}_1, \dots, \vec{l}_N, \vec{0}\}$, where $\vec{0}$ is a N -dimensional all-zero vector. The final allocation solution of (1), denoted by \vec{y}^ϵ , is randomly produced following the distribution $\Omega(\vec{x}^p)$ below:

$$\Omega(\vec{x}^p) = \begin{cases} Pr[\vec{y}^\epsilon = \vec{x}^p] = 1 - \epsilon/2, \\ Pr[\vec{y}^\epsilon = \vec{l}_i] = \frac{\sum_{j=1}^N \theta_j^0 x_j^p}{N}, \forall i \in \{1, \dots, N\}, \\ Pr[\vec{y}^\epsilon = \vec{0}] = 1 - Pr[\vec{y}^\epsilon = \vec{x}^p] - \sum_{i=1}^N Pr[\vec{y}^\epsilon = \vec{l}_i]. \end{cases} \quad (6)$$

We can verify that the probabilities of all candidate solutions are positive and sum up to exactly 1. We then have that the expectation of \vec{y}^ϵ is

$$E[\vec{y}^\epsilon] = (1 - \epsilon/2)\vec{x}^p + \left(\frac{\sum_{j=1}^N \theta_j^0 x_j^p}{N}\right) \left(\sum_{i=1}^N \vec{l}_i\right) = P^T \vec{x}^p = \vec{x}^f. \quad (7)$$

Given the above, the design of all the candidate solutions in $\Omega(\vec{p})$ and probability assignment of each of them were aiming 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 (5). 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. Here, \vec{y}^ϵ is equal to x^p with a high probability of $1 - \epsilon/2$, which is in accordance with the $1 - \epsilon/2$ part in (5). 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 (4) can be guaranteed, which will be proved in Theorem 3.

Algorithm steps. 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}^ϵ
 - 3 Choose $\{\theta_1^j, \dots, \theta_N^j\}_{j \in \{0, \dots, KD-1\}}$ independently and identically in the interval $[0, \frac{\epsilon}{N}]$;
 - 4 Construct each perturbed parameter \hat{b}_i and \hat{R}_i^{kd} according to (2) and (3), respectively;
 - 5 Compute $\vec{x}^p = \text{Algorithm 1}(\hat{\vec{b}}, \hat{\vec{R}}, \vec{c})$;
 - 6 Produce distribution $\Omega(\vec{x}^p)$ according to (6);
 - 7 **return** A sample \vec{y}^ϵ according to $\Omega(\vec{x}^p)$ in (6).
-

We further illustrate our algorithm using a simple example in Fig. 2. We consider a toy example where only two bids are submitted to the auctioneer. Suppose $(0,0)$, $(0,1)$, $(1,1)$ and $(1,0)$ are the feasible solutions of the original problem in (1). Since the coefficients of the packing constraints \hat{R}_i^{kd} are randomly enlarged to be \hat{R}_i^{kd} in the perturbed problem (4), the set of feasible solutions of the perturbed problem could be shrunk. An extreme case could happen where both bids are accepted in the optimal solution of (1), i.e., $\vec{x}^* = (1, 1)$, while at most one of the two bids could be accepted in the optimal solution of (4) (the green area denotes the feasible region of the perturbed problem). By running the exact algorithm Alg. 1 to solve (4), the optimal solution $\vec{x}^p = (1, 0)$ is obtained. The near-optimality of Alg. 2 is basically achieved by two facts. One is that due to the small perturbation of \hat{R}_i^{kd} 's, there is an extremely small probability⁴ that the optimal solution of the original problem is infeasible to the perturbed problem. The other is that the perturbation of the vector \vec{b}^T could be compensated by the perturbation of the feasible solutions of the perturbed problem. As Fig. 2 (a) shows, Alg. 2 first perturbs \vec{b} to $P\vec{b}$ by rotating \vec{b} by a small angle α , and exactly solves for the maximal $(P\vec{b})^T \vec{x}$ from the three feasible

⁴For rigorousness, this probability could be quite high when there are a large number of bids submitted. Fortunately, according to the proof of Lemma 3, with the small bid assumption, even if the optimal solution of the original problem is infeasible to the perturbed problem, the objective value of the perturbed problem under \vec{x}^p will not deviate much to that under \vec{x}^* .

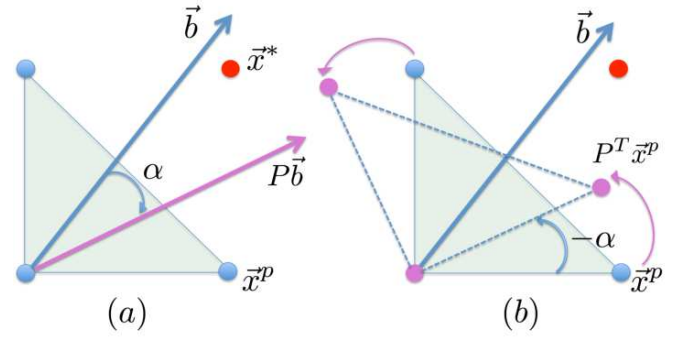


Fig. 2: An example to show the mechanism design

solutions (blue points in Fig. 2(a)). It is equivalent to rotating each feasible solution of the perturbed problem by an angle of $-\alpha$ and solving for the maximal $\vec{b}^T (P^T \vec{x})$ from the rotated solutions $P^T \vec{x}$'s (pink points in Fig. 2(b)). Directed by this insight, we randomly choose \vec{y}^ϵ from a set of feasible solutions of the original problem with the expectation of $P^T \vec{x}^p$. Because $\vec{b}^T (P^T \vec{x}^p) = (P\vec{b})^T \vec{x}^p$, the expected social welfare gained by Alg. 2, which is $\vec{b}^T \vec{y}^\epsilon$, preserves the approximation ratio of $(P\vec{b})^T \vec{x}^p$ to $\vec{b}^T \vec{x}^*$.

Analysis. 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 in 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 techniques [18][28]. The reason is that the expected number of the Pareto optimal solutions of the perturbed social welfare maximization problem in (4) 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 (5). According to smoothed analysis, Alg. 1 is said to run in *smoothed polynomial time*.

Theorem 3. *The expectation of the random variable $|\mathcal{P}(N)|^2$ of the perturbed social welfare maximization problem (4) is upper bounded by $O(N^{8KD}/\epsilon^{2KD})$, where the perturbed parameters are produced according to (2) and (3) with $\{\theta_1^j, \theta_2^j, \dots, \theta_N^j\}_{j \in \{0, \dots, KD-1\}}$ independently and identically chosen from a uniform distribution on the interval of $[0, \frac{\epsilon}{N}]$.*

The proof is given in Appendix F.

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

The proof is given in Appendix G.

(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), under the small bid assumption $\max_{i \in [N], k \in [K], d \in [D]} \frac{R_i^{kd}}{c_{kd}} \leq \frac{1}{2KD(2 + \frac{1}{\epsilon})}$.*

Proof. We have shown $E[\vec{y}^\epsilon] = \vec{x}^f$. Hence $E[\vec{b}^T \vec{y}^\epsilon] =$

$\vec{b}^T \vec{x}^f = \vec{b}^T (P^T \vec{x}^p) = (P\vec{b})^T \vec{x}^p = POPT \geq (1 - \epsilon)OPT$, based on Lemma 3. \square

C. The Truthful-in-Expectation VM Auction

Recall the MIDR mechanism design [32] that we introduced at the beginning of this section. By now we have designed the randomized allocation algorithm which chooses an allocation solution following the distribution $\Omega(\vec{x}^p)$ in (6). 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 [32].

Let \mathcal{T} and $\hat{\mathcal{T}}$ denote the set of all feasible solutions of ILP (1) and of the perturbed ILP (4), respectively. Recall the perturbation rule for each demand parameter in (3). We randomly enlarge the coefficient of the packing constraints from R_i^{kd} to \hat{R}_i^{kd} , therefore, $\hat{\mathcal{T}}$ is a subset of \mathcal{T} . For each feasible solution $\vec{x} \in \hat{\mathcal{T}}$, we can obtain a distribution $\Omega(\vec{x})$ in the same way as the distribution in (6) 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 $\{\theta_1^j, \dots, \theta_N^j\}_{j \in \{0, \dots, KD-1\}}$, the distribution $\Omega(\vec{x})$ is dependent on feasible solution \vec{x} , but independent of the users' bids. $\mathcal{R} = \{\Omega(\vec{x}), \forall \vec{x} \in \hat{\mathcal{T}}\}$ is a compact set including all the distributions indexed by feasible solution \vec{x} in the set $\hat{\mathcal{T}}$. Due to $\hat{\mathcal{T}} \subset \mathcal{T}$, \mathcal{R} is also a compact set of distributions over the feasible solutions in \mathcal{T} . Using \mathcal{R} as the distributional range, we have

$$\begin{aligned} E_{\vec{y}^\epsilon \sim \Omega(\vec{x}^p)}[\vec{b}^T \vec{y}^\epsilon] &= (P\vec{b})^T \vec{x}^p = \max_{\vec{x} \in \hat{\mathcal{T}}} (P\vec{b})^T \vec{x} \\ &= \max_{\vec{x} \in \hat{\mathcal{T}}} \vec{b}^T (P^T \vec{x}) \\ &= \max_{\Omega(\vec{x}) \in \mathcal{R}} E_{\vec{y} \sim \Omega(\vec{x})}[\vec{b}^T \vec{y}]. \end{aligned} \quad (8)$$

The first equation is due to (7). The second equation is because \vec{x}^p is the optimal solution of the perturbed ILP (4). The last equation is due to $E_{\vec{y} \sim \Omega(\vec{x})}[\vec{y}] = P^T \vec{x}$, which can be readily obtained according to (7). Hence the solution \vec{y}^ϵ 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. \square

Algorithm 3: The Randomized Auction Mechanism

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

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

$$p_i(\vec{y}^\epsilon) = \vec{b}_{-i}^T \vec{y}_{-i}^\epsilon - (\vec{b}^T \vec{y}^\epsilon - b_i y_i^\epsilon), \forall i \in [N]. \quad (9)$$

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

We summarize our complete randomized auction mechanism in Alg. 3. The following theorem shows that the auction design fulfils our design objectives.

Theorem 7. *The auction mechanism in Alg. 3, which combines the randomized allocation algorithm in Alg. 2 and the randomized VCG payment in (9), runs in polynomial time in expectation, is truthful in expectation, and achieves $(1 - \epsilon)$ -optimal social welfare in expectation, under the small bid assumption $\max_{i \in [N], k \in [K], d \in [D]} \frac{R_i^{kd}}{c_{kd}} \leq \frac{1}{2KD(2 + \frac{1}{\epsilon})}$.*

Proof. According to the principles of MIDR algorithms [32], to render a truthful-in-expectation mechanism, we should combine an MIDR allocation rule with a VCG-like payment as follows:

$$p'_i = E[\vec{b}_{-i}^T \vec{y}_{-i}^\epsilon - (\vec{b}^T \vec{y}^\epsilon - b_i y_i^\epsilon)], \quad (10)$$

where the expectation is computed as follows in more details: $E_{\vec{y}_{-i}^\epsilon \sim \Omega(\vec{x}_{-i}^p)}[\vec{b}_{-i}^T \vec{y}_{-i}^\epsilon] - E_{\vec{y}^\epsilon \sim \Omega(\vec{x}^p)}[\vec{b}^T \vec{y}^\epsilon - b_i y_i^\epsilon]$. Here \vec{x}_{-i}^p is the optimal solution of the perturbed ILP (4), 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 (10) efficiently, it has been proved [31] that instead of using (10), 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 (10).

The expectation of our random payment in (9) is exactly

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

Hence the random payment in (9) renders truthfulness in expectation and can be computed in polynomial time in expectation. Combining Theorem 4 and Theorem 5, this theorem is proved. \square

V. PERFORMANCE EVALUATION

We evaluate our randomized auction using trace-driven simulations, exploiting Google cluster-usage data [34] 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 VM bundle bid, and the resource demand of each task in the job into a VM in the bundle. There are around 1000 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_{md}^i , 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 $|\mathcal{B}_w|$ bundles from the pool to bid. We compute the capacity of type- k resource, c_{dk} , 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 $[0, 0.5W/N]$, 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 $W = 500$, the upper-bound on the number of bids a user can submit is $|\mathcal{B}_w| = 4$, the number of data centers is $D = 8$, and $\epsilon = 0.05$. We repeat each experiment for 50 times to obtain the average results. Note that some of the resource demands (R_i^{kd}) violate the small bid assumption that is required in the analysis of the approximation ratio of our allocation algorithm in Lemma 3, Theorem 5 and Theorem 7. In this sense, our algorithm could work well even when the bid demands are not as small as we require in the theoretical analysis.

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. Let RPAA represent our Randomized Perturbation-based Approximation Algorithm in Alg. 2 in the figures. Given $\epsilon = 0.05$, the theoretical expected approximation ratio is 0.95. Figs 3-5 show 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 Lemma 3 and (7) 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 W is large can also be explained by (13) in the proof of Lemma 3, that the inequality tends to equality when W (and hence N) is large. Fig. 3 and Fig. 4 also demonstrate that the average ratio decreases slightly with the increase of the number of data centers or the number of resource types, respectively. The reasons are the same:

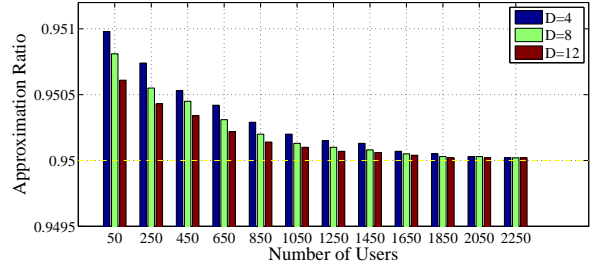


Fig. 3: Approx. Ratio of RPAA

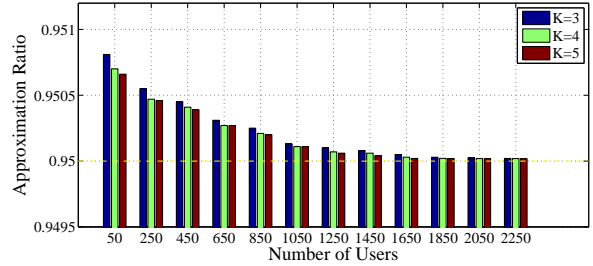


Fig. 4: Approx. Ratio of RPAA

either a larger D or a large K leads to more constraints in (1a) being perturbed, which causes a larger deviation between the original problem and the perturbed problem.

Fig. 5 further shows that the average approximation ratio is worse if a user submit more bids. The reason is clear: when the number of users W is fixed, the more bids a user can submit, the larger the total number of bids N is; hence due to the same reason as analyzed above, the ratio is closer to the theoretical one.

Fig. 6 compares the average approximation ratio obtained in our experiments and the respective theoretical approximation ratios at different values of ϵ , by plotting the relative approximation ratio $\frac{\text{average approx. ratio}}{1 - \epsilon}$. The average approximation ratio of our algorithm outperforms the theoretical ratio more when ϵ is larger, since $\theta_i^0, i = 1, \dots, N$ are larger (they are selected in the interval $[0, \epsilon/N]$) and the gap between the empirical approximation ratio and the theoretical one is larger according to (13) in the proof of Lemma 3. The better performance at a smaller N can be explained similarly as that for Figs 3-5.

B. Social Welfare Comparison with PDAA

We now compare the social welfare achieved by our Alg. 2 with the primal-dual approximation algorithm in [16] (which is essentially the algorithm used in [17] as well), denoted by PDAA. The algorithm in [16] does not consider the

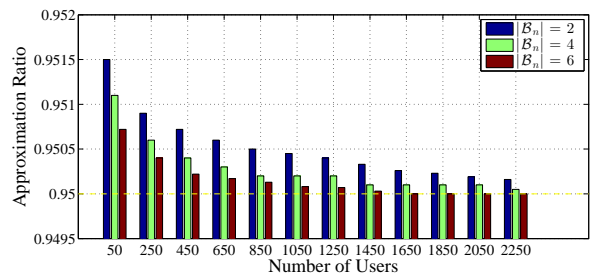


Fig. 5: Approx. Ratio of RPAA

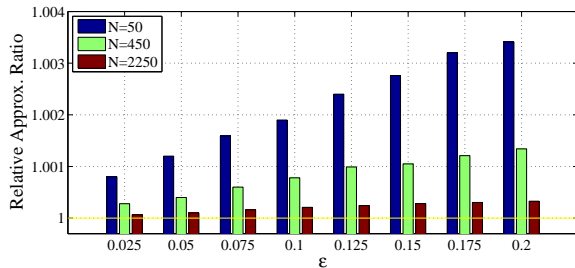


Fig. 6: Relative Approx. Ratio of RPAA

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.

Fig. 7 and Fig. 8 show that our algorithm consistently outperforms the algorithm in [16] 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 [16].

Fig. 7 also indicates that the social welfare of both algorithms increases with the increase of W and $|\mathcal{B}_w|$. The resource capacity in our experiments is set to be roughly linear in the total resource demand of the users, and when W 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. 8 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. If any resource demand in any data center was not satisfied, a bid would be rejected. Moreover, as explained for Fig. 3 and Fig. 4, more data centers lead to more constraints being perturbed (demands enlarged) in Alg. 2, which makes \bar{x}^p deviate more from the optimum of the original problem. Note that there is a high probability for the final allocation solution \bar{y}^c to be equal to \bar{x}^p . Based on all the above, the chance for each bundle to be accepted decreases, and the social welfare decreases slightly with the increase of D .

C. User Satisfaction Comparison with PDAA

We next evaluate user satisfaction achieved by both RPAA and PDAA, which is the percentage of users accepted as winners in the respective auctions. Fig. 9 and Fig. 10 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. 10 decreases slightly as more data centers are included, suffering from the same cause as explained for Fig. 8.

Finally, we remark on the running time incurred by the two algorithms. Figs. 7–10 illustrate that our algorithm outperforms the algorithm in [16] 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 $O(N^6 \log N)$, where N is the number of bids, while our algorithm has an expected time complexity of $O(KDN^{8KD+1}/\epsilon^{2KD})$. (details in the proof of Theorem 4 in Appendix G). 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 geodistributed 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 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 Elastic Compute Cloud,” <http://aws.amazon.com/ec2/>.
- [2] “Windows Azure,” <http://www.windowsazure.com/>.
- [3] “GoGrid,” <http://www.gogrid.com>.
- [4] “Amazon EC2 Instance Types,” <http://aws.amazon.com/ec2/instance-types/>.
- [5] “Recommended Memory Configurations for the MapReduce Service,” <https://ambari.apache.org/1.2.0/installing-hadoop-using-ambari/content/ambari-chap3-7-9a.html>.
- [6] 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.
- [7] “Amazon EC2 Spot Instances,” <http://aws.amazon.com/ec2/spot-instances/>.
- [8] Q. Wang, K. Ren, and X. Meng, “When Cloud Meets eBay: Towards Effective Pricing for Cloud Computing,” in *Proc. of IEEE INFOCOM*, 2012.
- [9] W. Wang, B. Liang, and B. Li, “Revenue Maximization with Dynamic Auctions in IaaS Cloud Markets,” in *Proc. of IEEE/ACM IWQoS*, 2013.
- [10] S. Zaman and D. Grosu, “Combinatorial Auction-Based Dynamic VM Provisioning and Allocation in Clouds,” in *Proc. of IEEE CloudCom*, 2011.

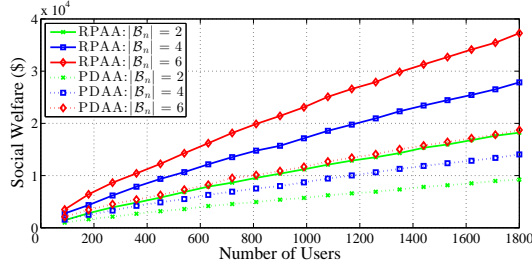


Fig. 7: Social Welfare of RPAA and PDAA

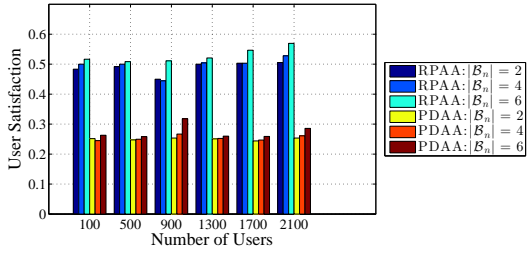


Fig. 9: User Satisfaction of RPAA and PDAA

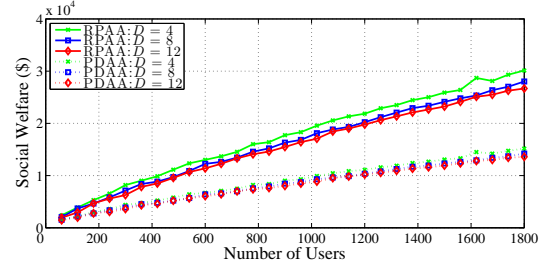


Fig. 8: Social Welfare of RPAA and PDAA

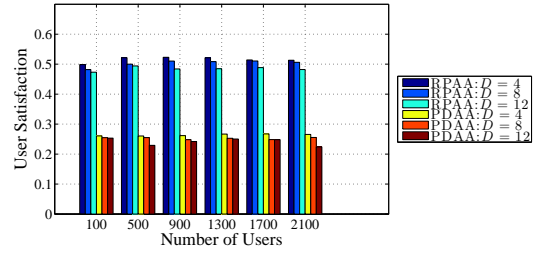


Fig. 10: User Satisfaction of RPAA and PDAA

- [11] —, “Combinatorial Auction-based Allocation of Virtual Machine Instances in Clouds,” *Journal of Parallel and Distributed Computing*, vol. 73, no. 4, pp. 495–508, 2013.
- [12] W. Vickrey, “Counterspeculation Auctions and Competitive Sealed Tenders,” *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.
- [13] 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.
- [14] D. Lehmann, L. I. O’callaghan, and Y. Shoham, “Truth revelation in approximately efficient combinatorial auctions,” *Journal of the ACM*, vol. 49, no. 5, pp. 577–602, 2002.
- [15] R. Lavi and C. Swamy, “Truthful and near-optimal mechanism design via linear programming,” in *Proc. of IEEE FOCS*, 2005, pp. 595–604.
- [16] L. Zhang, Z. Li, and C. Wu, “Dynamic Resource Provisioning in Cloud Computing: A Randomized Auction Approach,” in *Proc. of IEEE INFOCOM*, 2014.
- [17] 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.
- [18] D. Spielman and S.-H. Teng, “Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time,” in *Proc. of ACM STOC*, 2001.
- [19] T. Brunsch and H. Röglin, “Improved Smoothed Analysis of Multiobjective Optimization,” *Journal of the ACM*, vol. 62, no. 1, pp. 4:1–4:58, 2015.
- [20] 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.
- [21] 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.
- [22] M. Alicherry and T. V. Lakshman, “Network Aware Resource Allocation in Distributed Clouds,” in *Proc. of IEEE INFOCOM*, 2012.
- [23] 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.
- [24] 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.
- [25] L. Mashayekhy, M. M. Nejad, and D. Grosu, “A PTAS Mechanism for Provisioning and Allocation of Heterogeneous Cloud Resources,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2386–2399, 2015.
- [26] W. Shi, C. Wu, and Z. Li, “RSMOA: A Revenue and Social Welfare Maximizing Online Auction for Dynamic Cloud Resource Provisioning,” in *Proc. of IWQoS*, 2014.
- [27] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. Lau, “Online Auctions in IaaS Clouds: Welfare and Profit Maximization with Server Costs,” in *Proc. of ACM SIGMETRICS*, 2015.
- [28] H. Röglin and S.-H. Teng, “Smoothed Analysis of Multiobjective Optimization,” in *Proc. of IEEE FOCS*, 2009.
- [29] 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.
- [30] X. Chen, X. Deng, and S.-H. Teng, “Computing Nash Equilibria: Approximation and Smoothed Complexity,” in *Proc. of IEEE FOCS*, 2006.
- [31] S. Dughmi and T. Roughgarden, “Black-Box Randomized Reductions in Algorithmic Mechanism Design,” in *Proc. of IEEE FOCS*, 2010.
- [32] S. Dobzinski and S. Dughmi, “On the Power of Randomization in Algorithmic Mechanism Design,” in *Proc. of IEEE FOCS*, 2009.
- [33] G. L. Nemhauser and Z. Ullmann, “Discrete Dynamic Programming and Capital Allocation,” *Management Science*, vol. 15, no. 9, pp. 494–505, 1969.
- [34] 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>.
- [35] J. Puchinger, G. R. Raidl, and U. Pferschy, “The Multidimensional Knapsack Problem: Structure and Algorithms,” *INFORMS Journal on Computing*, vol. 22, no. 2, pp. 250–265, 2010.
- [36] A. Kulik and H. Shachnai, “There is No EPTAS for Two-dimensional Knapsack,” *Information Processing Letters*, vol. 110, no. 16, pp. 707–710, 2010.
- [37] “Moments of a random variable,” [https://en.wikipedia.org/wiki/Moment_\(mathematics\)](https://en.wikipedia.org/wiki/Moment_(mathematics)).

APPENDIX A PROOF OF THEOREM 1

Proof. Let

$$R_j^i = \begin{cases} R_{kd}^i & j = 1, \dots, KD \\ 1 & \text{if } i \in \mathcal{B}_{j-KD} \\ 0 & \text{if } i \notin \mathcal{B}_{j-KD} \end{cases} \quad j = KD + 1, \dots, KD + W$$

and

$$c_j = \begin{cases} c_{kd} & j = 1, \dots, KD \\ 1 & j = KD + 1, \dots, KD + W \end{cases}$$

The social welfare maximization problem (1) can be converted to a Multi-dimensional Knapsack Problem in polynomial time:

$$\text{maximize } \sum_{i=1}^N b_i x_i \quad (11)$$

subject to:

$$\sum_{i=1}^N x_i R_j^i \leq c_j, \quad j = 1, \dots, KD + W,$$

$$x_i \in \{0, 1\}, \quad \forall i \in [N]$$

The Multi-dimensional Knapsack Problem is a classic strongly NP-hard combinatorial optimization problem [35]. Moreover, it has no FPTAS unless P=NP [20][36]. \square

APPENDIX B PROOF OF LEMMA 1

Proof. If $\vec{x}^{(i)} \in \mathcal{P}(i)$, then $\vec{x}^{(i)}$ is not dominated by any solutions in $\mathcal{P}(i)$. If $\vec{x}^{(i)} - x_i^{(i)}$ (i.e., the vector obtained by removing the last element $x_i^{(i)}$ from $\vec{x}^{(i)}$) is dominated by some solution in $\mathcal{P}(i-1)$, e.g., $\vec{x}_d^{(i-1)}$, then $\vec{x}_d^{(i-1)} + x_i^{(i)}$ (i.e., the vector obtained by appending $x_i^{(i)}$ to the end of $\vec{x}_d^{(i-1)}$) dominates $\vec{x}^{(i)}$, which is a contradiction. Thus we can conclude that if $\vec{x}^{(i)} \in \mathcal{P}(i)$, then $\vec{x}^{(i)} - x_i^{(i)} \in \mathcal{P}(i-1)$. \square

APPENDIX C PROOF OF LEMMA 2

Proof. According to the construction from $\mathcal{P}(i)'$ to $\mathcal{P}(i)$ in Algorithm 1, some solutions of $\mathcal{P}(i)'$ will be pruned. When we merge $\mathcal{P}(i-1) + 0$ and $\mathcal{P}(i-1) + 1$ to $\mathcal{P}(i)'$, (1) if all the solutions in $\mathcal{P}(i-1) + 0$ are kept in $\mathcal{P}(i)'$, then $|\mathcal{P}(i)'| \geq |\mathcal{P}(i-1) + 0|$; (2) Otherwise, if $\exists \vec{x}_a^{(i)} \in \mathcal{P}(i-1) + 0$ which is pruned, then $\exists \vec{x}_b^{(i)} \in \mathcal{P}(i-1) + 1$ that dominates $\vec{x}_a^{(i)}$, which means there always exists a solution which substitutes the removed one. Hence $|\mathcal{P}(i)| \geq |\mathcal{P}(i-1)|$. \square

APPENDIX D PROOF OF LEMMA 3

Proof. We define solution \vec{x}^{*-} to be a feasible solution to the perturbed problem which is no larger than \vec{x}^* component-wisely. Due to $\vec{b} = P\vec{b}$, we have

$$POPT = (P\vec{b})^T \vec{x}^p \geq (P\vec{b})^T \vec{x}^{*-}, \quad (12)$$

since \vec{x}^{*-} is a feasible solution to the perturbed problem. In the following, we will show that the perturbed objective value under \vec{x}^{*-} is at least a $(1-\Delta)$ -fraction of the perturbed objective value under \vec{x}^* , which is further at least a $(1-\Delta)(1-\epsilon/2)$ -fraction of OPT , the original objective value:

$$(P\vec{b})^T \vec{x}^{*-} \geq (1-\Delta)(P\vec{b})^T \vec{x}^*$$

$$\geq (1-\Delta) \sum_{i \in [N]} \left((1-\epsilon/2)b_i + \frac{\epsilon \sum_{i' \in [N]} b_{i'}}{N} \right) x_i^* \quad (13)$$

$$\geq (1-\Delta)(1-\epsilon/2)\vec{b}^T \vec{x}^* \quad (14)$$

The rest of the proof is to show that we could upper-bound $\Delta \leq \epsilon/2$ by assuming that each bid only requests for a small amount of demand for each type of resource. Let \mathcal{S}^* denote the set of the accepted bids of \vec{x}^* . To find a \vec{x}^{*-} , a feasible solution to the perturbed problem, our method is to construct a feasible solution to the perturbed problem by dropping some

bids from the accepted bids of \vec{x}^* , until all the constraints are feasible again. This idea is directed by the fact that the demand parameters are only slightly perturbed, thus dropping a small amount of demand from the original optimal solution may only lose little social welfare. Note that such \vec{x}^{*-} may not be unique, but we only need to find a possible one and lower bound the derived perturbed objective value under it.

TABLE II: Dropping-based-on-Sorting Algorithm DoS

- 1: Identify \mathcal{Q}^+ , the set of the violated constraints;
- 2: For each $j \in \mathcal{Q}^+$:
- 3: Sort all the bids of the original optimal set \mathcal{S}^* in the increasing order of $\frac{b_j}{R_j^j}$;
- 4: Drop the sorted bids one by one, until the constraint j is feasible.

To prove that the perturbed objective value under \vec{x}^{*-} is at least $1 - \epsilon/2$ of that under \vec{x}^* is equivalent to show that the total bidding price of dropped bids is at most a $\epsilon/2$ fraction of the total bidding price aggregated by the accepted bids of \vec{x}^* . Before that, we define \mathcal{Q} to be the set of j (recall all the tuples of (k, d) are sorted and re-indexed by j). Let \mathcal{Q}^+ be the subset of \mathcal{Q} where at each element $j \in \mathcal{Q}^+$, the total perturbed demand of dimension j exceeds the respective capacity. Recall that in the algorithm DoS , we drop bids in $|\mathcal{Q}^+|$ rounds to make the violated constraints feasible. A critical concept \mathcal{S}_j^- is defined to be the set of dropped bids in the round of j . Three facts are shown in (15) and (16).

$$\sum_{i \in \mathcal{S}_j^-} \hat{R}_i^j \leq (1+2\epsilon)R_{max}^j, \quad \forall j \in \mathcal{Q}^+ \quad (15)$$

$$\sum_{i \in \mathcal{S}^*} \hat{R}_i^j > c_j, \forall j \in \mathcal{Q}^+; \quad \sum_{i \in \mathcal{S}^*} \hat{R}_i^j \leq c_j, \quad \forall j \in \mathcal{Q} \setminus \mathcal{Q}^+ \quad (16)$$

Here, fact in (16) follows the definitions of \mathcal{Q}^+ and \mathcal{Q} . The fact in (15) is derived as follows. Recall that the perturbation of demand parameter is $\hat{R}_i^j = R_i^j + \frac{\theta_i^j \sum_{i' \in [N]} R_{i'}^j}{N}$ where $\theta_i^j \sim U(0, \frac{\epsilon}{N})$. We define the maximal demand of k -type resource in data center d to be $R_{max}^j = \max_{i \in [N]} R_i^j$. Then we have

$$\sum_{i \in \mathcal{S}^*} \hat{R}_i^j \leq \sum_{i \in \mathcal{S}^*} R_i^j + \sum_{i \in \mathcal{S}^*} \frac{\epsilon}{N} R_{max}^j \leq \sum_{i \in \mathcal{S}^*} R_i^j + \epsilon R_{max}^j. \quad (17)$$

Moreover, we have the total demand under the optimal solution of the original problem is at most the capacity of each type of resource at each data center, i.e., $\sum_{i \in \mathcal{S}^*} R_i^j \leq c_j$ for each $j \in \mathcal{Q}$. Thus, for each $j \in \mathcal{Q}^+$, we can upper-bound the total perturbed demand under \vec{x}^* , to be $\sum_{i \in \mathcal{S}^*} \hat{R}_i^j \leq c_j + \epsilon R_{max}^j$. It means that if the dropped amount is at least ϵR_{max}^j of each $j \in \mathcal{Q}^+$, the packing constraints will be feasible again. Now the question is, in each round, how much demand will be actually dropped at most? Consider at some time in round j when the dropping is still not finished yet. We must have that the dropped j th demand is less than ϵR_{max}^j ; otherwise the remaining j th demand does not exceed c_j , the capacity of j th demand, which means this round of dropping should

have been stopped. Then consider the moment in this round that the total j th demand that have been dropped is less than ϵR_{max}^j , but the bid to be dropped next will finish this round of dropping. We call this moment *critical moment*. Note that this critical moment always exists in each dropping round. Since the very next dropped bid in the critical moment is at most $\hat{R}_{max}^j \leq (1 + \epsilon)R_{max}^j$. Plus ϵR_{max}^j , the dropped amount before the critical moment, we prove the fact in (15).

The above three facts essentially mean that we find a \bar{x}^{*-} , a feasible solution of the perturbed problem, such that the allocated resource amount is not much less than that under \bar{x}^* , the optimal solution to the original problem. Recall that to lower-bound Δ , the remaining is to prove that the total perturbed bidding price generated by remaining bids is not much less than that of \bar{x}^* . How to associate the remaining demand to the the remaining social welfare? In fact, the sorting operation gives the answer. Recall in each round $j \in \mathcal{Q}^+$, we sort and drop the bids of \mathcal{S}^* in the increasing order of $\frac{\hat{b}_i}{\hat{R}_i^j}$, the bidding price per unit of j th demand. It means that the dropped bids have lower bidding price per unit of demand j than the average $\frac{\hat{b}_i}{\hat{R}_i^j}$ over all the bids in \mathcal{S}^* . The basic idea to upper-bound the dropped social welfare is as follows. In each round j , the dropped j th demand could be upper-bounded according to the fact in (15). Then we could upper-bound the fraction of the loss of perturbed social welfare due to the dropped bids in the j th round over the total perturbed social welfare. Finally, taking over all the rounds, we upper-bound the fraction of the total loss of perturbed social welfare over that under \bar{x}^* , which is shown as follows.

$$\sum_{j \in \mathcal{Q}^+} \frac{\sum_{i \in \mathcal{S}_j^-} \hat{b}_i}{\sum_{i \in \mathcal{S}^*} \hat{b}_i} \leq \sum_{j \in \mathcal{Q}^+} \frac{\sum_{i \in \mathcal{S}_j^-} \hat{R}_i^j}{\sum_{i \in \mathcal{S}^*} \hat{R}_i^j} \quad (\text{due to sorting}) \quad (18)$$

$$\leq \sum_{j \in \mathcal{Q}^+} \frac{(1 + 2\epsilon)R_{max}^j}{c_j} \quad (\text{due to: (15) and (16)}) \quad (19)$$

$$\leq KD(1 + 2\epsilon) \times \frac{1}{2KD(2 + \frac{1}{\epsilon})} = \frac{\epsilon}{2} \quad (20)$$

The first inequality in (20) is due to the small bid assumption stated in Lemma 3, $\frac{R_{max}^j}{c_j} \leq \frac{1}{2KD(2 + \frac{1}{\epsilon})}$. The assumption essentially means that each bid has a small demand of each type of resource in the desired data center, compared to the corresponding capacity. Thus, $1 - \Delta$ in (14), the fraction of remaining perturbed social welfare over that under \bar{x}^* is lower-bounded to be $1 - \frac{\epsilon}{2}$.

According to (14), we have that the social welfare of the perturbed problem under \bar{x}^- over that under \bar{x}^* is at least $(1 - \Delta)(1 - \epsilon/2) \geq (1 - \epsilon/2)(1 - \epsilon/2) \geq 1 - \epsilon$, which serves as a critical step to lower-bound the approximation ratio of our allocation algorithm. Note that our analysis method is valid for milder conditions that require a higher upper-bound of the fraction of bid demand over capacity. Nevertheless, to obtain the $(1 - \epsilon)$ -approximation ratio, we will consider such a small bid assumption which is stated in Lemma 3, Theorem 5 and Theorem 7.

APPENDIX E PROOF OF THEOREM 2

Proof. There are two main steps in each i th round:

1. Given $\mathcal{P}(i - 1)$, we construct $\mathcal{P}(i)'$ by using two copies of all the solutions $\bar{x}^{(i-1)} \in \mathcal{P}(i - 1)$ and adding the i th bid to each solution of one copy with the value of 0 and 1, respectively. The time this step takes is linear in $|\mathcal{P}(i - 1)|$, increasing the total bidding price and resource demands of all the solutions in $\mathcal{P}(i - 1)$.

2. For each solution $\bar{x}^{(i)} \in \mathcal{P}(i)'$, we compare $(b(\bar{x}^{(i)}), C_{kd}(\bar{x}^{(i)}), \forall k \in [K], \forall d \in [D])$ to all the other solutions in $\mathcal{P}(i)'$ and delete all the dominated solutions. Thus the time this step takes is linear in $KD|\mathcal{P}(i)'|^2$.

According to the analysis in step 1 and step 2, the running time is bounded by:

$$O(KD \sum_{i=1}^{N-1} |\mathcal{P}(i)|^2).$$

$$\text{Due to Lemma 2, } O(KD \sum_{i=1}^{N-1} |\mathcal{P}(i)|^2) \leq O(KDN|\mathcal{P}(N)|^2). \quad \square$$

APPENDIX F PROOF OF THEOREM 3

Proof. This theorem desires to upper-bound the number of Pareto optimal solutions in expectation when parameters in (4) are independently and randomly perturbed. We adopt classical smoothed analysis in which the expected number of Pareto optimal solutions usually relies on the input size and the maximal perturbation density of the perturbed input parameters. Let ϕ denote the upper-bound of the perturbation density of each \hat{b}_i and \hat{R}_i^{kd} . Now we calculate ϕ of our perturbed problem. Since θ_i^j is drawn from $[0, \frac{\epsilon}{N}]$, the value of \hat{b}_i lies in the interval $[(1 - \epsilon)b_i, (1 - \epsilon)b_i + \frac{\epsilon \sum_{j=1}^N b_j}{N^2}]$, with the interval length of $\frac{\epsilon \sum_{j=1}^N b_j}{N^2}$. Let $b_{max} = \max\{b_1, \dots, b_N\}$. We have $\sum_{i'=1}^N b_{i'} \geq b_{max}$. Thus the interval length is no smaller than $\frac{\epsilon b_{max}}{N^2}$ and equivalently, the density of \hat{b}_i is upper-bounded everywhere in the interval $[(1 - \epsilon)b_i, (1 - \epsilon)b_i + \frac{\epsilon \sum_{j=1}^N b_j}{N^2}]$ by $\frac{N^2}{\epsilon b_{max}}$. Similarly, the interval length of the perturbation of \hat{R}_i^j is no smaller than $\frac{\epsilon R_{max}^j}{N^2}$, and the density of \hat{R}_i^j is upper-bounded everywhere in the interval $[R_i^j, R_i^j + \frac{\sum_{i'=1}^N R_{i'}^j}{N^2}]$ by $\frac{N^2}{\epsilon R_{max}^j}$. Moreover, without loss of generality, we can normalize all the bidding prices to be in $[0, 1]$ by dividing each b_i by b_{max} . We also normalize all the demands to be in $[0, 1]$ by diving each R_i^j by R_{max}^j and diving each c_j by R_{max}^j . The perturbation density will be upper-bounded by $\frac{N^2}{\epsilon}$.

According to the latest result of the smoothed number of Pareto optimal solutions for a multi-objective integer programming [19], we obtain the result in the Proposition 1. Before showing the proposition, we show that the result of a multi-objective optimization problem works for our problem.

According to the definition of a Pareto optimal solution to a multi-objective integer programming [19], a solution \bar{x} in the feasible region is Pareto optimal if and only if there is

□

no solution which is at least as good as \vec{x} and better than \vec{x} in at least one criteria. That is, a Pareto optimal solution is a feasible solution that is not dominated by any other solution in the feasible region. In a multi-objective optimization problem, there are multiple objective functions to be optimized in a feasible region. In our problem, according to the definition of Pareto optimal solution (Pareto Optimal Allocation), the defined objectives (criteria to compare two solutions, not the objective function) are $C_{kd}(\vec{x})$'s, the total demand for each k -type of resource in data center d , plus $s(\vec{x})$, the social welfare under \vec{x} . It means we have in total $KD + 1$ objects of criteria to define a Pareto optimal solution. In our problem, a solution \vec{x} is a Pareto optimal solution if and only if there does not exist a feasible solution \vec{x}' that dominates \vec{x} in all objects of criteria, which falls in the family of the Pareto optimal solution in [19]. Besides the fact that the definition of a Pareto optimal solution for a multi-objective optimization problem is essentially the same as ours, the perturbation policy of our work fits that of [19] where: Each coefficient (b_i and \hat{R}_i^j , $\forall j \in [KD - 1]$) is chosen independently according to its own quasi-concave probability density function where the density at each point of the perturbation interval is upper-bounded by ϕ . Here, quasi-concave density [19] requires that the probability density is non-decreasing within the left half perturbation interval while is non-increasing in the right half. Our perturbation of each parameter follows the uniform distribution which has a quasi-concave density function.

Proposition 1. *For any constant K , D and $\alpha \in \mathbb{N}$, the α th moment of the smoothed number of Pareto optimal solutions of a multi-objective binary programming is $O((N^{2KD} \phi^{KD})^\alpha)$ for quasi-concave perturbation density functions with density everywhere upper-bounded by ϕ .*

Here, according to definition of α th moment of a random variable [37], $\alpha = 2$ is the case to calculate the expectation of $|\mathcal{P}(N)|^2$. Thus putting $\phi = \frac{N^2}{\epsilon}$ and $\alpha = 2$ into Proposition 1, we have that $E[|\mathcal{P}(N)|^2] \leq O(N^{8KD} / \epsilon^{2KD})$, which the theorem follows. \square

APPENDIX G PROOF OF THEOREM 4

Proof. Combining Theorem 2 and Theorem 3, we derive the expected running time of our exact algorithm on the randomly perturbed ILP (4) (line 5 in Alg. 2) as

$$O(KDNE[|\mathcal{P}(N)|^2]) = O(KDN^{8KD+1} / \epsilon^{2KD}).$$

Note that K and D are fixed constants. The perturbation matrix P can be obtained in polynomial time (lines 3–4 in Alg. 2). The set of feasible solutions to ILP (1) and the distribution Ω can be constructed in polynomial time as well (lines 6 in Alg. 2). Hence the theorem is proven. \square