An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing

Weijie Shi, *Student Member, IEEE*, Linquan Zhang, *Student Member, IEEE*, Chuan Wu, *Member, IEEE*, Zongpeng Li, *Senior Member, IEEE*, and Francis C. M. Lau, *Senior Member, IEEE*

***Abstract*—Auction mechanisms have recently attracted substantial attention as an efficient approach to pricing and allocating resources in cloud computing. This work, to the authors' knowledge, represents the first online combinatorial auction designed for the cloud computing paradigm, which is general and expressive enough to both: 1) optimize system efficiency across the temporal domain instead of at an isolated time point; and 2) model dynamic provisioning of heterogeneous virtual machine (VM) types in practice. The final result is an online auction framework that is truthful, computationally efficient, and guarantees a competitive ratio ≈ 3.30 in social welfare in typical scenarios. The framework consists of three main steps: 1) a tailored primal-dual algorithm that decomposes the long-term optimization into a series of independent one-shot optimization problems, with a small additive loss in competitive ratio; 2) a randomized subframework that applies primal-dual optimization for translating a centralized cooperative social welfare approximation algorithm into an auction mechanism, retaining the competitive ratio while adding truthfulness; and 3) a primal-dual algorithm for approximating the one-shot optimization with a ratio close to e. We also propose two extensions: 1) a binary search algorithm that improves the average-case performance; 2) an improvement to the online auction framework when a minimum budget spending fraction is guaranteed, which produces a better competitive ratio. The efficacy of the online auction framework is validated through theoretical analysis and trace-driven simulation studies. We are also in the hope that the framework can be instructive in auction design for other related problems.**

***Index Terms*—Cloud computing, combinatorial auction, resource allocation, pricing, online algorithms, truthful mechanisms.**

I. Introduction

CLOUD computing has recently emerged as a new computing paradigm that enables prompt and on-demand access to computing resources. As exemplified in Amazon EC2 [1] and Microsoft Azure [2], cloud providers invest substantially into their datacenter infrastructure, providing a virtually unlimited "sea" of CPU, RAM, and storage resources to cloud users, often assisted by virtualization technologies. The elastic and on-demand nature of cloud computing assists cloud users to meet their dynamic and fluctuating demands with minimal management overhead, while the cloud ecosystem as a whole achieves *economies of scale* through cost amortization. Currently, most cloud providers adopt a fixed-price policy and charge users a fixed amount per-virtual-machine (VM) usage. For example, Table I shows the available VM types at Amazon EC2 and their hourly prices at different datacenters. Despite their apparent simplicity, fixed-price policies inherently lack market agility and efficiency, failing to rapidly adapt to real-time demand–supply relation changes. Consequently, overpricing and underpricing routinely occur, which either dispel or undercharge the users, jeopardizing overall system social welfare as well as the providers' revenue.

TABLE I
Amazon EC2 VM Instances

VM Type	CPU*	RAM	Disk	Virginia	Ireland	Tokyo
m1.medium	2	3.75GB	410GB	\$0.120	\$0.130	\$0.175
m1.large	4	7.5GB	840GB	\$0.240	\$0.260	\$0.350
m1.xlarge	8	15GB	1.68TB	\$0.480	\$0.520	\$0.700
c1.medium	5	1.7GB	350GB	\$0.145	\$0.165	\$0.185
c1.xlarge	20	7GB	1.68TB	\$0.580	\$0.660	\$0.740
m2.2xlarge	13	34.2GB	850GB	\$0.820	\$0.920	\$1.101

* EC2 compute units

Toward effectively discovering the market value of VMs, auction mechanisms have been at the focal point of recent literature on cloud resource allocation and pricing [3], [4]. Spot Instance [5] is a first-step attempt to apply the auction mechanism on Amazon EC2, which was enhanced by subsequent work [4], [6]. A series of recent work further study auction mechanism design in cloud markets from different perspectives [7]–[9]. Unfortunately, all existing designs either consider one-round auctions only or model VMs as type-oblivious commodities and fail to account for the providers' ability to dynamically assemble VMs.

Cloud Auctions Should Be Online: Real-world cloud resource transactions happen either when customer demands arrive or cloud resources become available, and hence are modeled more naturally by an online auction that incorporates the time dimension. Most cloud computing customers, enterprise or individual, are on a preallocated budget for a given time period (e.g., a year or a month like that in an ad-auction [10]). Thus, a customer's purchase desire drastically declines over time, which needs to be considered for a practical auction mechanism. However, most existing cloud auctions focus on a single-round auction only and ignore such temporal correlation in decision making [8].

Manuscript received June 25, 2014; revised December 01, 2014; February 17, 2015; and May 12, 2015; accepted May 28, 2015; approved by IEEE/ACM Transactions on Networking Editor U. Ayesta. Date of publication July 01, 2015; date of current version August 16, 2016. This work was supported in part by a grant from Hong Kong RGC under the contract HKU 718513 and NSERC Grant 10006298.

W. Shi, C. Wu, and F. C. M. Lau are with the Department of Computer Science, The University of Hong Kong, Hong Kong (e-mail: wjshi@cs.hku.hk; cwu@cs.hku.hk; fcmlau@cs.hku.hk).

L. Zhang and Z. Li are with the Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: linqzhan@ucalgary.ca; zongpeng@ucalgary.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2444657

Cloud Auctions Should Be Combinatorial: A cloud computing job in practice often demands a bundle of heterogeneous VM instances for its successful execution, and hence a cloud auction is naturally a combinatorial auction. For example, a social game application that consists of a front-end Web server layer, a load-balancing layer, and a back-end data storage layer is best served by a combination of VMs that are intended for communication-intensive, computation-intensive, and storage-intensive tasks, respectively. Such combinatorial VM auctions represent a dramatic departure of most existing VM auction designs that assume VMs are type-oblivious commodities, in that all VMs are essentially of the same type, and hence are substitutable or substitutable up to a simple multiplicative factor. Embracing heterogeneous VM types in the model further brings about the opportunity of considering *dynamic resource provisioning*: Decisions on VM assembling, which organizes the CPU, RAM, and Disk resource pools into typed VM instances, are no longer made randomly *a priori* [3], but made dynamically upon receiving user bids. Dynamic resource provisioning enables higher efficiency in cloud resource utilization, higher seller revenue for the provider, and higher social welfare for the entire cloud system.

This work generalizes and subsumes existing literature on cloud auctions by designing the first online combinatorial auction in which VMs of heterogeneous types are allocated in multiple consecutive time-slots. The final result is an online auction framework that simultaneously guarantees the following properties.

1) Truthfulness, the holy grail of auction mechanism design. It ensures that economically-motivated selfish buyers are automatically elicited to reveal their true valuations of the VMs they demand, in the bids submitted. This simplifies analysis of the resulting auction in theory and increases the predicability of auction outcomes in practice.
2) Combinatorial auctions, supporting heterogeneous VM types located at different datacenters. Besides heterogeneity in their types, another dimension of VM diversity may arise due to their geographical locations (assuming multiple datacenters). A combinatorial auction is hence necessary.
3) Dynamic resource provisioning. The number of instances of each VM type is not predefined, but dynamically adjusted as part of the auction mechanism, tailored to real-time user demand.
4) Online auction: In commercial cloud platforms, auctions are executed repeatedly and the prices change termly. Each user is subject to a practical budget limitation for a given time period. Our online auction models a long-time auction over multiple rounds that are coupled together by customer budgets. A competitive ratio of $e + \frac{1}{1-e} \simeq 3.30$ is guaranteed in typical scenarios, i.e., our online auction achieves a long-term social welfare that is at least a 1/3.30 fraction of the offline optimum.

Our proposed online auction framework consists of three main modules: A) translating online optimization into a series of one-round optimization problems; B) translating an approximation algorithm for one-round optimization into a truthful auction; and C) designing an effective approximation algorithm for one-round optimization.

First, we formulate a linear integer program that characterizes the long-term social welfare optimization problem in the cloud market for VMs and formulate the dual LP of its LP relaxation (without the integer constraints). A tailored primal-dual algorithm iteratively adjusts a dual variable corresponding to each customer's budget, acting as a shadow price that signals how "tight" the latter is. A series of one-round combinatorial VM auctions are then executed under a fixed shadow price vector. Such primal-dual decoupling of the auction rounds admits a rather intuitive interpretation: The algorithm strikes to avoid prematurely depleting a user's budget, and gives higher priority to cloud customers with low budget pressures during each auction round. As a result, we prove that the decomposition introduces an additive loss to the competitive ratio bounded by $\frac{1}{e-1}$.

Second, for each one-round combinatorial auction problem, we employ a randomized auction subframework, which exploits the underlying packing property of one-round social welfare maximization and translates any centralized cooperative approximation algorithm into an auction, inheriting the same approximation ratio while adding truthfulness. At the core of this translation is a primal-dual optimization-based decomposition technique that decomposes an optimal fractional solution to one-round social welfare maximization into a convex combination of integral solutions, recently developed in the literature of theoretical computer science [11] and successfully applied in the literature of computer networking [12]. We also propose a new binary search-based technique to find the minimum feasible scale-down ratio in the fractional solution decomposition and thereby improve the average-case performance of the online auction algorithm.

Third, we design a specific approximation algorithm for one-round VM allocation by applying iterative primal-dual solution updates followed by dual fitting. The resulting algorithm is polynomial-time computable and guarantees an approximation ratio λ that approaches e in practical scenarios. Combining all three modules together, the overall competitive ratio of the resulting online auction framework is bounded by $e + \frac{1}{e-1} \simeq 3.30$ in typical scenarios. We hope that the online auction framework proposed in this work, as well as its three components, may shed light to the design of auction mechanisms in related problem settings.

What's more, we further propose an improvement to the online auction framework in the case that a minimum budget spending fraction is guaranteed. The improved online auction algorithm uses a new method to adjust the dual variables corresponding to users' budgets over time and achieves a better competitive ratio.

The rest of the paper is organized as follows: We discuss related work in Section II, define the problem model in Section III, present the online algorithm framework and the auction algorithm in Sections IV and V, discuss an improvement of the online auction framework in Section VI, present simulations in Section VII, and conclude the paper in Section VIII.

II. Related Work

Auctions have been extensively studied for resource allocation in computing and network systems, which simultaneously target bidding truthfulness and economic efficiency. Classic applications of auctions are found in a wide range of research areas, such as network bandwidth allocation [13], wireless spectrum allocation [12], and wireless crowdsourcing [14].

The celebrated VCG mechanism [15] is a well-known type of auction. It is essentially the only type of auction that simultaneously guarantees both truthfulness and absolute economic efficiency (social welfare maximization), through calculating the optimal allocation and a carefully designed pricing rule. However, when the underlying allocation problem is NP-hard, which is common for combinatorial auctions [16], VCG becomes computationally infeasible. When polynomial-time approximation algorithms are applied to solving the underlying allocation problem, VCG loses its truthfulness property [17]. One usually needs to custom design a payment rule to work in concert with the approximation algorithm at hand to achieve truthfulness; for example, this can be done by exploiting the concept of *critical bids* [18]. Another relatively new alternative is to resort to the LP decomposition technique [11], as done in this work, which is universally applicable to problems with a packing or covering structure.

Recently, a series of auction mechanisms are designed for VM allocation in cloud computing. Wang *et al.* [6] apply the critical value method, and derive a mechanism that is collusion-resistant, an important property in practice. Yet their work, like many others, considers only one-round auctions; and their algorithm has a competitive ratio $O(\sqrt{k})$, where k is the number of VM instances. Mashayekhy *et al.* [19] propose an online mechanism for resource allocation in clouds, also based on the critical value method, but omit a theoretical performance analysis. Shanmuganathan *et al.* [20] introduce the concept of *bundles* in VM allocation. Zhang *et al.* [8] are among the first to study dynamic VM provisioning and design a truthful single-round auction using the LP decomposition method. However, our work is more advanced than theirs in two aspects: 1) We consider the problem over a period of time, instead of just one round, and serve cloud users in an online manner. Our mechanism more closely resembles a real-world cloud market in practice. 2) We not only apply but also propose improvements to the decomposing method, which can improve the performance of the online auction in practice.

Extending single-round truthful auctions into online auctions in a straightforward way usually breaks the truthfulness property [21]. The lack of future information brings a key challenge in pursuing truthfulness. For example, the VCG auction does not directly work in the online setting since the optimal allocation for the future cannot be calculated, even given unlimited computational resources. A known technique for achieving truthfulness in online auctions is based on the concept of a *supply curve* [22], as applied by Zhang *et al.* [4] in their design of an online cloud auction algorithm. The bidding language and the user characteristic proposed in their work are novel and capture the heterogeneous demands in cloud market. However, they only consider a single type of VM, significantly simplifying the underlying social welfare maximization. In absence of multiple VM types, their model naturally ignores the dynamic provisioning problem. Wang *et al.* propose an online auction for cloud markets [9], and their model also focuses on one type of VM only.

Many resource allocation algorithms have been proposed in cloud computing scenarios with different focuses. Alicherry *et al.* [23] consider network load when allocating VMs in a distributed cloud system. Maguluri *et al.* [24] tackle the randomness of arriving workloads and solve optimization problems for load balancing and VM scheduling. Xu *et al.* [25] summarize the recent attempts in managing performance overhead in clouds. Lin *et al.* [26], [27] study the energy efficiency in VM provisioning. VM migration overhead [28] and energy consumption [29] are also important practical issues in VM provisioning. In addition, data traffic commonly exists between VMs. Guo *et al.* [30] focus on traffic variation issues in the underlying datacenter networks and solve the VM placement problem with traffic awareness. These aspects are not fully covered in the current paper, and will be investigated in our future work.

III. Problem Model

A. Cloud System

We consider a cloud spanning Q geographically distributed datacenters, each with a pool of R types of resources including CPU, RAM, and disk storage that can be dynamically assembled into M different types of VMs for lease to cloud users. Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$. Each VM of type $m \in [M]$ is constituted by $\alpha_{m,r}$ units of type-r resource, for all $r \in [R]$. There are N users of the cloud system, which request VMs of different types to execute their jobs. The cloud provider acts as the auctioneer and leases VMs to the users through auctions. The system runs in a time-slotted fashion within a span of $1, 2, \ldots, T$, where T is a potentially large number. We suppose the available amounts of resources at each datacenter are time-varying, i.e., there are $A_{q,r}^{(t)}$ units of type-r resource in datacenter q at time $t \in [T]$, whose value may change from one time-slot to another.[1] In each time-slot, one round of the auctions is carried out, where the cloud provider decides the VM allocation for the current time-slot based on user bids. The terms *time-slot* and *round* are used interchangeably.

The N cloud users are bidders in the auctions, each submitting a bid containing K optional bundles in each round. A bundle consists of a list of desired quantities of VMs of different types, as well as the bidder's valuation for the bundle. Specifically, let $d_{n,k,m,q}^{(t)}$ denote the number of type-m VMs in datacenter q that user n specifies in its kth bundle in time-slot t, and $b_{n,k}^{(t)}$ be its valuation for this kth bundle in t. The kth bundle in the bid of user n in the auction at t is described by a $(MQ+1)$-tuple of elements $d_{n,k,m,q}^{(t)}, \forall m, q$, and $b_{n,k}^{(t)}$. The cases where a user may join and leave (intermittent bidding) or may bid a smaller number of bundles than K are all subsumed by our bid model by allowing empty bundles in the bids.

In each round, upon receiving users' bids, the cloud provider computes its resource allocation and produces the auction results, $y_{n,k}^{(t)} \in \{0, 1\}$, where $y_{n,k}^{(t)} = 1$ if user n wins bundle k and 0 otherwise, as well as its payment $\Pi_n^{(t)}$ for acquiring VMs in its winning bundle. We assume that a user can win at most one bundle among its K optional bundles in each round of the auctions (given that any need for combining two or more bundles can be expressed as a separate bundle already). In addition, the VM demands in each bundle cannot be supplied partially,

[1]The varying amounts of resources may be caused by removal or addition of servers, due to failure and recovery, or potential reservation or release of resources for special purposes, e.g., as part of a hybrid cloud of an enterprise.

TABLE II
NOTATION

N	# of users	[X]	integer set $\{1, 2, \ldots, X\}$
T	# of time slots	R	# of resource types
M	# of VM types	Q	# of datacenters

K	# of optional bundles in each bid
α_m^r	amount of resource r in each type-m VM
$A_{q,r}^{(t)}$	available resource r at datacenter q at time t
$b_{n,k}^{(t)}$	user n's valuation for its k-th bundle at t
$d_{n,k,m,q}^{(t)}$	# of type-m VM at dc q in n's k-th bundle at t
$c_{n,k,r,q}^{(t)}$	amount of resource r at dc q in n's k-th bundle at t
B_n	user n's total budget
$y_{n,k}^{(t)}$	user n wins its k-th bundle at time t or not
$y_{n,k}^{(t)F}$	optimal fractional solution
$y_{n,k}^{(t)l}$	an integer solution to (3)
$\Pi_n^{(t)}$	user n's payment at time t
$\Pi_n^{(t)F}$	user n's payment at time t under fractional VCG
$\Pi_n^{(t)l}$	user n's payment at time t under allocation $\mathbf{y}^{(t)l}$
β_l	probability to choose integer solution $\mathbf{y}^{(t)l}$
$u_n^{(t)}$	user n's utility at time t
ν	the competitive ratio of $\mathcal{A}_{round}$
λ	the approximation ratio of Alg. 2
$w_{n,k}^{(t)}$	the reduced valuation of n's k-th bundle at time t
B_{max}	max ratio: a single bundle bid / a user's budget
γ	$(1 + B_{max})^{1/B_{max}}$
g_n	user n's minimum budget spending fraction
g	$\min_{n\in[N]} g_n$

i.e., the cloud provider either provides all the required VMs in a bundle to the bidder or rejects the bundle.

Let $u_n^{(t)}$ denote the utility function of user n in time-slot t, which is decided by its valuations of the bundles and its payment at t. We will present the concrete form of the utility function in Section V. We assume user n has a total budget B_n, which is a bound of its overall payment in the auctions throughout the system span $[T]$ under consideration, e.g., a preallocated budget for VM rental over a month or a year, which is assumed to be public information. A user's valuations in its bids are independent from its current budget level, while its current budget level will be taken into consideration at the cloud provider when allocating resources.

We list important notation in this paper in Table II.

B. Online Auction Problem

We aim to design an online auction mechanism to be carried out by the cloud provider, which guides resource allocation in the cloud system in a round-by-round fashion through multiple consecutive rounds. The auction design targets the following properties.

1) *Truthfulness* (Definition 1): Bidding true valuations is a dominant strategy at the users, and consequently, both bidding strategies and auction design are simplified.
2) *Individual rationality*: Each bidder obtains a nonnegative utility by participating in the auction in any time-slot, i.e., $u_n^{(t)} \geq 0, \forall n, t$.
3) *Social welfare maximization*: The social welfare in our system is the sum of the cloud provider's revenue, $\sum_{t\in[T]}\sum_{n\in[N]}\Pi_n^{(t)}$, and all the users' utility gain, $\sum_{t\in[T]}\sum_{n\in[N]}(\sum_{k\in[K]} b_{n,k}^{(t)}y_{n,k}^{(t)} - \Pi_n^{(t)})$, which equals aggregated user valuation of the winning bundles (under truthful bidding), $\sum_{t\in[T]}\sum_{n\in[N]}\sum_{k\in[K]} b_{n,k}^{(t)}y_{n,k}^{(t)}$. Users' payment and the provider's revenue cancel out each other.

Definition 1 (Truthfulness): The auction mechanism is truthful if for any user n at any time t, declaring a bid that truthfully reveals its requirements of VM quantities, $d_{n,k,m,q}^{(t)}, \forall m, q, k$, and its valuations of bundles $b_{n,k}^{(t)}, \forall k$, always maximizes its expected utility, regardless of other users' bids.

We first formulate an offline social welfare optimization problem that provides the "ideal" optimal allocation strategies for the cloud provider to address users' VM demands in the entire system lifespan T, assuming bids are truthful. Let $c_{n,k,r,q}^{(t)} = \sum_m d_{n,k,m,q}^{(t)}\alpha_{m,r}$ be the amount of type-r resource at datacenter q required in user n's kth bundle

$$\text{maximize} \sum_{t\in[T]}\sum_{n\in[N]}\sum_{k\in[K]} b_{n,k}^{(t)}y_{n,k}^{(t)} \tag{1}$$

$$\text{s.t.} \sum\nolimits_{k\in[K]} y_{n,k}^{(t)} \leq 1 \quad \forall\, n \in [N], t \in [T] \tag{1a}$$

$$\sum\nolimits_{k\in[K]}\sum\nolimits_{t\in[T]} b_{n,k}^{(t)}y_{n,k}^{(t)} \leq B_n \quad \forall n \in [N] \tag{1b}$$

$$\sum_{n\in[N]}\sum_{k\in[K]} c_{n,k,r,q}^{(t)}y_{n,k}^{(t)} \leq A_{q,r}^{(t)} \quad \forall q\in[Q], r\in[R], t\in[T] \tag{1c}$$

$$y_{n,k}^{(t)} \in \{0,1\} \quad \forall n\in[N], k\in[K], t\in[T]. \tag{1d}$$

Constraint (1a) specifies that each user can win at most one bundle each round. Constraint (1b) is the budget constraint at each user. Constraint (1c) limits the overall demand for each type of resource in the winning bundles by the amount available.

Introducing dual variable vectors s, x, and z to constraints (1a), (1b), and (1c) respectively, and ignoring the binary variable constraint (1d) temporarily, we can formulate the dual of the resulting linear program to be used in the primal-dual algorithm design in Section IV

$$\min \sum_{n\in[N]} B_n x_n + \sum_{n\in[N]}\sum_{t\in[T]} s_n^{(t)} + \sum_{q\in[Q]}\sum_{r\in[R]}\sum_{t\in[T]} A_{q,r}(t)z_{q,r}^{(t)} \tag{2}$$

$$\text{s.t. } b_{n,k}^{(t)}x_n + s_n^{(t)} + \sum_{r\in[R]}\sum_{q\in[Q]} c_{n,k,r,q}^{(t)}z_{q,r}^{(t)} \geq b_{n,k}^{(t)} \quad \forall n \in [N], k \in [K], t \in [T] \tag{2a}$$

$$x_n, s_n^{(t)}, z_{q,r}^{(t)} \geq 0 \quad \forall n \in [N], q \in [Q], r \in [R], t \in [T]. \tag{2b}$$

To derive an optimal solution to (1), complete knowledge about the system over its entire lifespan is needed, which is apparently not practical. In a dynamic cloud system, the provider should allocate resources on the fly, based on the current amount of available resources, $A_{q,r}^{(t)}$'s, and users' bidding bundles including resource demands $d_{n,k,m,q}^{(t)}$'s and valuations $b_{n,k}^{(t)}$'s, which are not known *a priori*. We seek to design an online auction mechanism for real-time resource allocation, which also guarantees truthful bidding. We achieve the goals in two steps. First, in Section IV, we assume that a truthful auction mechanism to be carried out *in each time-slot* is known and guarantees an approximation ratio ν, and we propose an online algorithm framework that produces a competitive ratio of $(1 + B_{\max})(\nu + \frac{1}{\gamma-1})$ as compared to the offline optimum.

Algorithm 1 The Online Algorithm Framework $\mathcal{A}_{\text{online}}$

```
$x_n^{(0)} \leftarrow 0, \forall n \in [N]$
for all $1 \leq t \leq T$ do
    $w_{n,k}^{(t)} = \begin{cases} 0, & \text{if } x_n^{(t-1)} \geq 1 \\ b_{n,k}^{(t)}\left(1 - x_n^{(t-1)}\right), & \text{otherwise} \end{cases} \quad \forall n, k.$
    Run $\mathcal{A}_{\text{round}}$. Let $\mathcal{N}$ be the set of winning users, and $k_n$ be the index of their corresponding winning bundle, for $n \in \mathcal{N}$. Define $b_{n,k_n}^{(t)} = 0$ for $n \notin \mathcal{N}$.
    $x_n^{(t)} \leftarrow x_n^{(t-1)}\left(1 + \frac{b_{n,k_n}^{(t)}}{B_n}\right) + \frac{b_{n,k_n}^{(t)}}{B_n(\gamma - 1)} \quad \forall n \in [N]$
end for
```

Second, in Section V, we design a single-round randomized auction, which achieves the approximation ratio of ν as well as individual rationality and truthfulness.

IV. Online Algorithm Framework

We design an online algorithm framework $\mathcal{A}_{\text{online}}$ as shown in Algorithm 1, which solves the offline optimization problem (1) and its dual (2), using a subroutine $\mathcal{A}_{\text{round}}$ running at each time-slot. We next discuss the one-round resource allocation problem to be solved by $\mathcal{A}_{\text{round}}$, as well as the design rationale of the online algorithm framework.

A. One-Round Resource Allocation

Assuming truthful bids are known, the one-round social welfare maximization problem at time t is as follows, which includes the constraints from the offline optimization problem (1) related to the current time-slot, and excludes the user budget constraints (dealt with in the online algorithm framework instead). $w_{n,k}^{(t)}$, a reduced valuation of user n for bundle k from the actual valuation $b_{n,k}^{(t)}$ adjusted according to the level of its remaining budget, is used in the objective function. The rationale will be detailed in Section IV-B. Given $w_{n,k}^{(t)}$, the cloud provider's current resource supplies $A_{q,r}^{(t)}$'s, and users' resource demands $c_{n,k,r,q}^{(t)}$'s, the one-round optimization problem decides the optimal resource allocation $y_{n,k}^{(t)}$ at t

$$\text{maximize} \sum\nolimits_{n\in[N]} \sum\nolimits_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)} \tag{3}$$

$$\text{s.t.} \sum\nolimits_{k\in[K]} y_{n,k}^{(t)} \leq 1 \quad \forall n \in [N] \tag{3a}$$

$$\sum_{n\in[N]} \sum_{k\in[K]} c_{n,k,r,q}^{(t)} y_{n,k}^{(t)} \leq A_{q,r}^{(t)} \quad \forall q \in [Q], r \in [R] \tag{3b}$$

$$y_{n,k}^{(t)} \in \{0,1\} \quad \forall n \in [N], k \in [K]. \tag{3c}$$

Adopting the same dual variables as in the dual of (1) and omitting constraint (3c), we formulate the dual of LP (3)

$$\text{minimize} \sum\nolimits_{n\in[N]} s_n^{(t)} + \sum\nolimits_{q\in[Q]} \sum\nolimits_{r\in[R]} A_{q,r}^{(t)} z_{q,r}^{(t)} \tag{4}$$

$$\text{s.t. } s_n^{(t)} + \sum_{q\in[Q]} \sum_{r\in[R]} c_{n,k,r,q}^{(t)} z_{q,r}^{(t)} \geq w_{n,k}^{(t)} \quad \forall n \in [N], k \in [K] \tag{4a}$$

$$s_n^{(t)}, z_{q,r}^{(t)} \geq 0 \quad \forall n \in [N], q \in [Q], r \in [R]. \tag{4b}$$

The primal problem (3) is a special case of the multidimensional multiple-choice 0–1 knapsack problem [31], which is both NP-hard and, more strongly, has no fully polynomial-time approximation schemes unless P = NP [32]. What we will pursue in $\mathcal{A}_{\text{round}}$ is an auction mechanism, which not only guarantees individual rationality and truthfulness, but also employs a primal-dual approximation algorithm that solves problems (3) and (4) to decide resource allocation in polynomial time with a small approximation ratio. We delay the discussion of the auction mechanism to Section V, but first utilize its properties when analyzing our online algorithm framework. We will show that given a competitive ratio ν achieved by the one-round auction mechanism, our online algorithm framework achieves a good competitive ratio.

B. Online Algorithm

When a good approximation algorithm for one-round resource allocation is in place, the difficulty of designing an online algorithm lies in achieving a good competitive ratio, defined as the maximum ratio between the offline optimal social welfare derived by solving (1) exactly and the social welfare produced by the online algorithm. The budget limits the bundles a user can acquire over the T rounds of auctions, leading to different amounts of overall social welfare when the budget is spent in different rounds. The intuition we follow in designing the online algorithm is that inefficiency in social welfare may appear when a user's budget runs out at an early stage since its future bids become invalid after its budget depletion, narrowing down possible future allocation decisions at the provider, prohibiting larger social welfare. The ideal scenario is that each user's budget can last for all the T rounds of auctions, making it possible for the cloud provider to explore the best resource allocation strategies over the entire span to approach the best overall social welfare.

Under this intuition, we should be cautious when winning a bundle suddenly exhausts a user's remaining budget. Our main idea in the online algorithm in Algorithm 1 is to associate the resource allocation in each round with the users' remaining budgets. We introduce an auxiliary variable $x_n^{(t)}$ for each user $n \in [N]$, whose value starts at 0, increases with the decrease of the remaining budget of the user, and reaches 1 when the budget is exhausted. Instead of the actual valuation $b_{n,k}^{(t)}$ of each bundle, $w_{n,k}^{(t)} = b_{n,k}^{(t)}(1 - x_n^{(t-1)})$ is used in the one-round resource allocation A_{round} as in (3), such that the bid from a user with a smaller remaining budget will be evaluated less at the cloud provider, leading to a lower chance of acquiring a bundle. A user's budget lasts for a longer period of time as a result. $x_n^{(t)}$ is updated after each round of resource allocation in Algorithm 1, where $\gamma = (1 + B_{\max})^{\frac{1}{B_{\max}}}$. $B_{\max} = \max_{n\in[N],t\in[T],k\in[K]}\{b_{n,k}^{(t)}/B_n\}$, which is the maximum ratio between the valuation of any bundle and the corresponding user's budget. We consider $B_{\max} \ll 1$, given that users typically do not put a large proportion of their total budget on one bundle in one round. The increment of $x_n^{(t)}$ is carefully computed (see proof of Theorem 1), such that the budget constraint (1b) is guaranteed over the T rounds of online auctions. We set dual variable x_n in the offline dual problem (2), associated with constraint (1b), to the value of the auxiliary variable $x_n^{(t)}$ after T rounds $x_n^{(T)}$. In this way, the adjustment of $x_n^{(t)}$ in each round can be understood as the adjustment of the dual variable x_n toward an optimal solution to the offline dual problem (2).

The performance of our online algorithm in Algorithm 1 is stated in Theorem 1, with a detailed proof in Appendix A.

Theorem 1: If we have an auction mechanism in $\mathcal{A}_{\text{round}}$ that carries out resource allocation in each round to produce feasible solutions for (3) and (4), and guarantees $\nu p \geq d$ (hence the competitive ratio of the auction algorithm is also ν), $\mathcal{A}_{\text{online}}$ is $(1+B_{\max})(\nu + \frac{1}{\gamma-1})$-competitive for optimization (1). Here, $p = \sum_{n\in[N]}\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)}$ is the objective value of the one-round resource allocation problem in (3), and $d = \sum_n s_n^{(t)} + \sum_{q,r} A_{q,r}^{(t)} z_{q,r}^{(t)}$ is the dual objective value in (4).

We note that when $B_{\max} \to 0$, the competitive ratio approaches $\nu + \frac{1}{e-1}$, i.e., the long-term online optimization framework incurs only an additive loss of $\frac{1}{e-1}$ in competitive ratio, as compared to the one-round allocation algorithm.

V. Randomized Auction Mechanism

We now present a randomized auction mechanism $\mathcal{A}_{\text{round}}$ that efficiently allocates resources according to users' bids in each time-slot and guarantees individual rationality and truthfulness. The auction mechanism in each round allocates resources according to the one-round resource allocation problem in (3) and decides the payments from the winning bidders. The classic Vickrey–Clarke–Groves (VCG) mechanism [15] is a potential candidate for our auction design, which assigns items (VM bundles in our case) to bidders in a socially optimal manner by solving a corresponding resource allocation problem, charges each winner the externality it exerts on other bidders, and ensures that the optimal strategy for a bidder is to bid its true valuations. However, our allocation problem in (3) is NP-hard, and hence a VCG mechanism becomes computationally infeasible. We therefore resort to a fractional version of the VCG auction for achieving both computational efficiency (polynomial-time complexity) and economic efficiency [social welfare maximization in (3)], by applying the VCG mechanism to the LP relaxation of the integer program (3). The fractional VCG mechanism produces fractional bundle allocation results, which are not practically applicable. We further employ a primal-dual optimization-based decomposition technique that decomposes such an optimal fractional solution into a convex combination of integral solutions, and then design a randomized auction that randomly picks one from the integral solutions as the bundle allocation result in each round and retains the nice properties of a fractional VCG auction. We detail the fractional VCG auction, the decomposition technique, and the randomized auction design in Sections V-A–V-C.

A. Fractional VCG Auction

In the fractional VCG auction, the auctioneer solves the LP relaxation of (3) by relaxing constraint (3c) to $0 \leq y_{n,k}^{(t)} \leq 1, \forall n,k$, to decide the bundle allocation in t. Let $\mathbf{y}^{(t)\boldsymbol{F}} = (y_{n,k}^{(t)\boldsymbol{F}})_{\forall n,k}$ denote the resulting optimal fraction allocation, where $y_{n,k}^{(t)\boldsymbol{F}} \in [0,1]$. To compute the VCG payment from a winner, the auctioneer solves the LP relaxation again with the winner excluded from the allocation. Let $\widetilde{V}_{-n}^{(t)}$ denote the social welfare achieved when winner n is excluded. The payment of winner n, $\Pi_n^{(t)\boldsymbol{F}}$, is: $\Pi_n^{(t)F} = \widetilde{V}_{-n}^{(t)} - \sum_{n'\neq n}\sum_{k\in[K]} y_{n',k}^{(t)F} w_{n',k}^{(t)}$.

The utility function $u_n^{(t)}$ of bidder n in a VCG auction is typically defined as the difference between its valuation and its payment. In our online auction framework, a user's utility in each round should be related not only to its valuation and payment, but also to its remaining budget: Intuitively, smaller utility gain is appreciated if a user won a bundle when its remaining budget is small, and larger otherwise. We characterize this property using a utility function

$$u_n^{(t)} = \sum_{k\in[K]} y_{n,k}^{(t)F} w_{n,k}^{(t)} - \Pi_n^{(t)F}. \tag{5}$$

This utility function is consistent with the social welfare calculation in the one-round allocation problem (3). Thus, a user's budget can potentially last longer, enabling later acquirement of better bundles with the same consumption of budget (i.e., the same payment), contributing to social welfare efficiency over all T rounds of auctions.

We show in Theorem 2 that under this utility function, bidding true valuations is the best strategy for each user in the fractional VCG auction. A nonnegative utility is guaranteed for each bidder, based on VCG auction theory [15].

Theorem 2: The fractional VCG auction that produces fractional allocation $y_{n,k}^{(t)F}, \forall n \in [N], k \in [K]$, and payments $\Pi_n^{(t)F}, \forall n \in [N]$, is truthful and individual rational.

The detailed proof can be found in Appendix B.

B. Decomposing the Fractional Solution

Since fractional VM bundles are impractical in real-world cloud systems, we next decompose the fractional solution into a combination of integer solutions, which will be used by our randomized auction mechanism. We apply the LP duality-based decomposition technique [11]. The goal of the decomposition is to find $\beta_l \in [0,1]$ and a set of integer solutions $\mathbf{y}^{(t)l}, \forall l \in \boldsymbol{L}$ ($\boldsymbol{L}$ is an index set), to the one-round resource allocation problem (3), such that $\sum_{l\in\boldsymbol{L}} \beta_l \mathbf{y}^{(t)l} = \mathbf{y}^{(t)\boldsymbol{F}}$, and $\sum_{l\in\boldsymbol{L}} \beta_l = 1$. The randomized auction in each round can choose the integer solution $\mathbf{y}^{(t)l}$ with probability β_l, achieving a good competitive ratio in social welfare in expectation, as compared to that achieved by the optimal integer solution to (3). However, there in fact does not exist a convex combination of integer solutions, $\sum_{l\in\boldsymbol{L}} \beta_l \mathbf{y}^{(t)l}$, that equals the fraction solution $\mathbf{y}^{(t)\boldsymbol{F}}$, because otherwise the expected social welfare achieved by these integer solutions equals that achieved by the fractional solution, which apparently contradicts with the fact that the fractional solution achieves a higher social welfare than any integer solution. Therefore, to achieve a feasible decomposition, we need to scale down the optimal fractional solution by a certain factor. According to [11], if there exists an approximation algorithm that solves the one-round allocation problem (3) with an approximation ratio of λ and guarantees $\lambda p \geq d$ (where p and d are the objective function values of the primal problem (3) and dual (4), respectively), we can use λ as

Algorithm 2 A Primal-Dual Algorithm to Solve One-round Allocation Problem (3)

```
$\mathcal{N} \leftarrow \emptyset, z_{\text{base}} \leftarrow QR \cdot exp((C_{\min}^{(t)} - 1))$
$y_{n,k}^{(t)} \leftarrow 0, s_n^{(t)} \leftarrow 0, z_{q,r}^{(t)} \leftarrow 1/A_{q,r}^{(t)}, \forall n, k, r, q$
while $\sum_{r\in[R]} \sum_{q\in[Q]} A_{q,r}^{(t)} z_{q,r}^{(t)} < z_{\text{base}}$ AND $|\mathcal{N}| \neq N$ do
  for all $n \notin \mathcal{N}$ do
    $k(n) = \arg\max_{k\in[K]}\{w_{n,k}^{(t)}\}$
  end for
  $n^* = \arg\max_{n\in[N]}\{\frac{w_{n,k(n)}^{(t)}}{\sum_{r\in[R]}\sum_{q\in[Q]} c_{n,k(n),r,q}^{(t)} z_{q,r}^{(t)}}\}$
  $y_{n^*,k(n^*)}^{(t)} \leftarrow 1, s_{n^*}^{(t)} \leftarrow w_{n^*,k(n^*)}^{(t)}, \mathcal{N} \leftarrow \mathcal{N} \cup \{n^*\}$
  for all $r \in [R], q \in [Q]$ do
    $z_{q,r}^{(t)} \leftarrow z_{q,r}^{(t)} \cdot {z_{\text{base}}}^{c_{n^*,k(n^*),q,r}^{(t)}/(A_{q,r}^{(t)} - C_{q,r}^{(t)})}$
  end for
end while
```

the scaling factor, and rest assured that a feasible solution to the following decomposition problem exists:

$$\text{minimize } \sum_{l\in\boldsymbol{L}} \beta_l \tag{6}$$

$$\text{s.t. } \sum_{l\in\boldsymbol{L}} \beta_l y_{n,k}^{(t)l} = y_{n,k}^{(t)F}/\lambda \quad \forall n \in [N], k \in [K] \tag{6a}$$

$$\sum_{l\in\boldsymbol{L}} \beta_l \geq 1 \tag{6b}$$

$$\beta_l \geq 0 \quad \forall l \in \boldsymbol{L}. \tag{6c}$$

We give a simple numerical example to illustrate the decomposition. Suppose there are two users and two types of VMs. The fractional solution is (0.3, 0.7, 1, 0.7), representing that the first (second) user receives 0.3 (1) unit of VM 1 and 0.7 (0.7) unit of VM 2, respectively. It can be decomposed into two integer solutions: (1, 0, 1, 0) and (0, 1, 1, 1), with corresponding probabilities 0.3 and 0.7. However, in fact we cannot find any combination of integer solutions satisfying the one-round constraints in (3) (i.e., the integer solutions are not feasible) since the optimal fractional solution should be better than any feasible integer solution. Therefore, we have to divide the fractional solution by a factor λ before decomposing it.

We next present a primal-dual algorithm that solves (3) with an approximation ratio λ, and then discuss how to solve the decomposition problem (6) to obtain β_l and $\mathbf{y}^{(t)l}$, $\forall l \in \boldsymbol{L}$.

1) Primal-Dual Algorithm for One-Round Resource Allocation: Algorithm 2 is our primal-dual approximation algorithm to the NP-hard allocation problem (3). $C_{q,r}^{(t)} = \max_{n,k}\{c_{n,k,r,q}^{(t)}\}$ is the maximum amount of type-r resource at datacenter q required by any bundle in t. $C_{\min}^{(t)} = \min_{r\in[R],q\in[Q]}\{A_{q,r}^{(t)}/C_{q,r}^{(t)}\}$ is the minimum ratio between the total amount of available resource of a type and the amount of the resource required by one bundle. In practice, the resource pool is substantially larger than a single user's demand, and hence $C_{\min}^{(t)} \gg 1$. The main idea of the algorithm is to introduce an auxiliary variable $z_{q,r}^{(t)}$ for each type of resource [which is the dual variable associated with constraint (3b)], acting as the unit price in allocation decision. The unit price $z_{q,r}^{(t)}$ is updated according to the remaining amount of this type of resource. The algorithm evaluates each bundle according to the unit prices and the amount of required resources and always chooses the user with a higher bid on a lower valued bundle as the winner.

Theorem 3: Algorithm 2 computes feasible primal and dual solutions for (3) and (4) and guarantees $\lambda p \geq d$ (p and d defined in Theorem 1), $\lambda = 1 + \epsilon^{(t)}(e(QR)^{1/(C_{\min}^{(t)}-1)} - 1)(1 + 1/(c_{\min}^{(t)} - 1))$ with $\epsilon^{(t)} = \max_{k_1,k_2\in[K],r\in[R]}\{c_{n,k_1,r,q}^{(t)}/c_{n,k_2,r,q}^{(t)}\}$. The approximation ratio of Algorithm 2 is also λ.

The proof of the theorem is given in Appendix C. Here, $\epsilon^{(t)}$ is the maximum ratio between the overall demands for any type of resource in any two bundles in a user's bid in t. When Q, R are small constants and the provider's resource pool is relatively large compared to users' resource demands in the bundles, λ tends to $1 + \epsilon^{(t)}(e - 1)$. If further $\epsilon^{(t)} \to 1$, or each user bids a single bundle, λ tends to e.

2) Decomposition With LP Duality-Based Technique: To solve the decomposition problem (6), we can first find all the possible integer solutions $\mathbf{y}^{(t)l}$ to (3) using some exhaustive search method, and then directly solve (6) to derive the decomposition coefficients β_l's. However, this method has an exponential-time complexity since there are an exponential number of possible integer solutions $\mathbf{y}^{(t)l}$, and hence an exponential number of variables in LP (6). We therefore resort to its dual, formulated in (7), where dual variables $v_{n,k}^{(t)}$ and τ associate with primal constraints (6a) and (6b), respectively

$$\text{maximize } \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau \tag{7}$$

$$\text{s.t. } \sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} + \tau \leq 1 \quad \forall l \in \boldsymbol{L} \tag{7a}$$

$$\tau \geq 0. \tag{7b}$$

Even though the dual (7) has an exponential number of constraints, the ellipsoid method can be applied to solve it in polynomial time. The ellipsoid method obtains an optimal dual solution using a polynomial number of separating hyperplanes. Algorithm 2 acts as a key component of a separation oracle for generating these separating hyperplanes, and a feasible integer solution to (3) can be derived each time a separating hyperplane is generated [11]. Hence, a polynomial number of candidate integer solutions $\mathbf{y}^{(t)l}$'s are produced through the process of the ellipsoid method, and the primal problem (6) can be reduced to a linear program with a polynomial number of variables (β_l's) corresponding to these integer solutions. Then, we can solve the reduced primal problem in polynomial time. The correctness of the above decomposition method is given in Lemma 1, with detailed proof and the construction of the separation oracle in Appendix D.

Lemma 1: The decomposition method correctly obtains a polynomial number of integer solutions $\{\mathbf{y}^{(t)l}\}_{l\in\boldsymbol{L}}$ to the one-round allocation problem (3) and the probabilities $\beta_l, \forall l \in \boldsymbol{L}$, which solve (6), in polynomial time.

C. Randomized Auction

Algorithm 3 gives our randomized auction to be carried out in each round of the online algorithm in Algorithm 1. It selects an integer solution $\mathbf{y}^{(t)l}$ produced by the decomposition method with probability β_l. The payment from a winning bidder n satisfies two conditions: 1) The expectation of the payment should be equal to the scale-down fractional payment, $\sum_{l\in\boldsymbol{L}} \Pi_n^{(t)l}\beta_l = \Pi_n^{(t)F}/\lambda$, in order to remain truthfulness. 2) The payment $\Pi_n^{(t)l}$

Algorithm 3 One-Round Randomized Auction $\mathcal{A}_{\text{round}}$ in t

```
1: Solve LP relaxation of (3), with $w_{n,k}^{(t)} = \max\{0, (1 - x_n^{(t-1)})b_{n,k}^{(t)}\}$. Denote the fractional solution by $y_{n,k}^{(t)F}, \forall n, k$.
2: Calculate the fractional payment $\Pi_n^{(t)F}$ by VCG payment rule.
3: Solve the pair of primal-dual decomposition LPs in (6) and (7) using the ellipsoid method, using Algorithm 2 as a separation oracle, and derive a polynomial number of integer solutions to (3), $\mathbf{y}^{(t)l}$, $\forall l \in \boldsymbol{L}$, and the corresponding decomposition coefficients $\beta_l$.
4: Choose $\mathbf{y}^{(t)l}$ with probability $\beta_l$, $\forall l \in \boldsymbol{L}$
5: $\forall n \in [N], \Pi_n^{(t)l} = \Pi_n^{(t)F} \cdot \frac{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)l}}{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)F}}$
```

should be no larger than n's valuation of its winning bundle $\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)l}$ in order to guarantee individual rationality.

The following theorem provides the properties achieved by the randomized auction, with proof in Appendix E.

Theorem 4: $\mathcal{A}_{\text{round}}$ runs in polynomial time and is truthful, individual rational, and $\lambda(1 + B_{\max})$-competitive.

Our online auction results when we plug in the one-round randomized auction $\mathcal{A}_{\text{round}}$ into the online algorithm framework $\mathcal{A}_{\text{online}}$ in Algorithm 1. The competitive ratio of the online auction can be derived readily from Theorem 1 using $\nu = \lambda(1 + B_{\max})$, the competitive ratio of the one-round randomized auction given in Theorem 4.

Theorem 5: $\mathcal{A}_{\text{online}}$ in Algorithm 1 combining with $\mathcal{A}_{\text{round}}$ in Algorithm 3 constitutes a truthful, individual rational, $(1 + B_{\max})(\lambda(1 + B_{\max}) + \frac{1}{\gamma-1})$- competitive online auction.

The complete proof of Theorem 5 can be found in Appendix F. We note that when $B_{\max} \to 0$, the competitive ratio tends to $\lambda + \frac{1}{e-1}$. Following the discussions on Theorem 3 in Section V-B.1, when λ tends to e, the competitive ratio of the online auction tends to $e + \frac{1}{e-1} \simeq 3.30$.

D. Improving the Scale-Down Factor

We have decomposed the fractional solution in Section V-B after scaling it down by the approximation ratio λ of the one-round allocation Algorithm 2, such that a feasible solution to the decomposition problem (6) is guaranteed. According to Theorem 5, the scale-down factor λ is closely related to the competitive ratio of our online auction, such that a smaller scale-down factor leads to a better competitive ratio. However, a scale-down factor smaller than λ may not guarantee a feasible decomposition. We therefore design a binary search algorithm in Algorithm 4 to compute the smallest scale-down factor that enables feasible decomposition.

The algorithm is designed based on a property of the scale-down factor, as given in Theorem 6 (proof in Appendix G). With its monotonicity, we can find the smallest, feasible scale-down factor using binary search (with arbitrary small error). We should note that this trial-and-error method may improve the performance of our online auction algorithm on average in practice, but does not change the theoretical competitive ratio

Algorithm 4 Binary searching smallest scale-down factor

```
Require: allowable error $\delta$
  Decide the scale-down ratio $\lambda$ in Algorithm 3 with the following steps: $\lambda_l \leftarrow 1$, $\lambda_r \leftarrow \lambda$
  while $\lambda_r - \lambda_l > \delta$ do
    $\lambda_m \leftarrow (\lambda_l + \lambda_r)/2$
    Solve (7) with scale-down factor $\lambda_m$.
    If Decomposing success then $\lambda_r \leftarrow \lambda_m$ Else $\lambda_l \leftarrow \lambda_m$
  end while
  Solve (7) with scale-down factor $\lambda_r$.
```

Algorithm 5 The Improved Online Algorithm Framework based on Minimum Budget Spending Guarantee $\mathcal{A}'_{\text{online}}$

```
$x_n^{(0)} \leftarrow 0, x_n^{\boldsymbol{s}} \leftarrow g_n/(\gamma^{1-g_n} - (1 - g_n)), \forall n \in [N]$
for all $1 \le t \le T$ do
  $w_{n,k}^{(t)} = \begin{cases} 0, & \text{if } x_n^{(t-1)} \ge 1 \\ b_{n,k}^{(t)}(1-\max\{x_n^{(t-1)}, x_n^{\boldsymbol{s}}\}), & \text{otherwise} \end{cases} \quad \forall n, k$
  Run $\mathcal{A}_{\text{round}}$. Let $\mathcal{N}$ be the set of winning users, and $k_n$ be the index of their corresponding winning bundle, for $n \in \mathcal{N}$. Define $b_{n,k_n}^{(t)} = 0$ for all $n \notin \mathcal{N}$.
  $x_n^{(t)} \leftarrow x_n^{(t-1)} + \Big(\max\{x_n^{(t-1)}, x_n^{\boldsymbol{s}}\} + \frac{x_n^{\boldsymbol{s}}(1-g_n)}{g_n}\Big)\frac{b_{n,k_n}^{(t)}}{B_n} \ \forall n \in [N]$
end for
```

in Theorem 5 in the worst case. We will investigate the effectiveness of the improved scale-down factor in our trace-driven simulations.

Theorem 6: If the fractional allocation $y_{n,k}^{(t)F}$ can be decomposed under scale-down factor λ_1, then it can also be decomposed under any factor $\lambda_2 > \lambda_1$.

VI. Improvement of the Online Algorithm Framework With Minimum Budget Spending

In this section, we propose an improvement of the online algorithm framework $\mathcal{A}_{\text{online}}$ when additional information on users' budget spending is available [10]. Suppose that each user's total spending over T rounds is at least a fraction of its budget. Specifically, user n is going to use at least an amount $g_n B_n$ of its overall budget, where $0 \le g_n \le 1$. When this minimum budget spending guarantee is in place, we design an improved online algorithm framework $\mathcal{A}'_{\text{online}}$, as given in Algorithm 5. We show that this improved online algorithm can achieve a better worst-case competitive ratio $(1 + B_{\max})(\lambda(1 + B_{\max}) + \frac{1-g}{\gamma^{1-g}-(1-g)})$, where $g = \min_{n\in[N]}\{g_n\}$ is the minimum budget spending fraction among all users, as compared to the competitive ratio of the original online auction $\mathcal{A}_{\text{online}}$, $(1+B_{\max})(\lambda(1+B_{\max})+\frac{1}{\gamma-1})$ given in Theorem 5. For example, in the case that $B_{\max} \to 0$, the additive loss in the competitive ratio brought by $\mathcal{A}_{\text{online}}$, as compared to the one-round auction $\mathcal{A}_{\text{round}}$, is $\frac{1}{\gamma-1} \approx 0.582$; if we know that all users are going to spend at least $g_n = 90\%$ of their respective budget, the additive loss brought by the improved online auction $\mathcal{A}'_{\text{online}}$ is approximately 0.099.

The main idea of improving $\mathcal{A}_{\text{online}}$ to $\mathcal{A}'_{\text{online}}$ is as follows. If user n spends a large fraction of its overall budget over time, the value of x_n will be close to 1 upon termination of the online algorithm. In the original online algorithm in Algorithm 1, the value of x_n is exponentially increased from 0 to 1 with the increase of budget consumption since we can prove that no matter what the final value of x_n is, updating x_n exponentially guarantees a good performance of the online algorithm. Now we know that x_n will at least reach a value $x_n^{\boldsymbol{s}} = g_n/(\gamma^{1-g_n} - (1-g_n))$ (computed according to the minimum budget expenditure $g_n B_n$ based on the update formula of x_n in Algorithm 1), and we can increase the value of x_n linearly with the increase of budget consumption before it reaches $x_n^{\boldsymbol{s}}$, and then increase x_n in the original way (as in Algorithm 1) afterwards. In this way, we can prove that a better competitive ratio can be achieved.

The properties achieved by $\mathcal{A}'_{\text{online}}$ are given in Theorem 7, with complete proof in Appendix H.

Theorem 7: $\mathcal{A}'_{\text{online}}$ in Algorithm 5 combining with $\mathcal{A}_{\text{round}}$ in Algorithm 3 constitutes a truthful, individual rational, $(1+B_{\max})(\lambda(1+B_{\max})+\frac{1-g}{\gamma^{1-g}-(1-g)})$- competitive online auction.

VII. Performance Evaluation

We evaluate our online auction design using trace-driven simulations. We investigate 23 types of VMs distributed in Q (default value 9) datacenters, assembled from three types of resources (CPU, RAM, and Disk capacity, $R = 3$). Users' resource demands are extracted from Google cluster-usage data [33], which record jobs submitted to the Google cluster with information on their resource demands (CPU, RAM, Disk). We translate each job request in the Google data into a bidding bundle as follows: We generate a set of VMs in Table I randomly that altogether can make up the resource demands in the job request; the valuation in the bidding bundle is calculated as the product of the total cost to acquire these VMs according to the VM charges in Table I and a random coefficient in the range of [0.5, 2]. In this way, we obtain a pool of bidding bundles from the Google data. In each round of the online auctions, each user randomly picks K (default 3) bundles in the pool, tags each VM in each bundle with a datacenter randomly selected from the Q datacenters, and bids the bundles. A user n's total budget B_n is decided by multiplying the sum of valuations in all the bundles the user may bid in the T rounds of auctions by a random coefficient in the range of [0.5, 1]. We also compute the total amount of resource of each type r needed by all the bid bundles of N users in each round, scale it down using a random factor in [0, 1], and distribute the overall amount of type-r resource to Q datacenters evenly, to obtain the amount of available resource, $A_{q,r}^{(t)}$, for each type of resource in each datacenter at each time. Note that we run random bundle selection for each user over T rounds first to estimate users' budgets B_n's and available resources in the datacenters, $A_{q,r}^{(t)}$'s, before running the experiments to evaluate our online auction with the obtained B_n's and $A_{q,r}^{(t)}$'s. We suppose the maximum ratio between the overall demands for any type of resource in any two bundles in a user's bid in each time-slot t, i.e., $\epsilon^{(t)}$, is no larger than 2.5, by picking up bundles with similar resource demands for each user in the auctions, which we believe to reflect the reality better. We also suppose all users spend at least 70% of their respective budget.

A. Performance of Our Online Algorithms

We compare the performance of four algorithms:

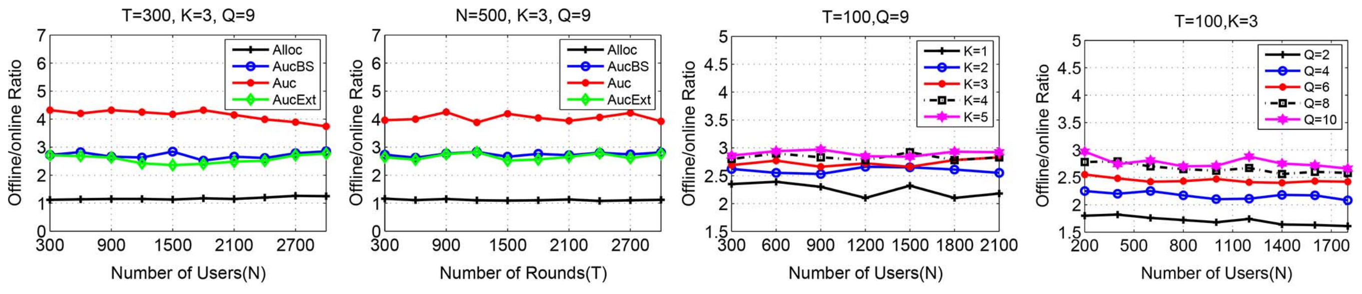
- *Alloc*, a pure online resource allocation algorithm, with the one-round resource allocation algorithm Algorithm 2 serving in the place of A_{round} in $\mathcal{A}_{\text{online}}$ in Algorithm 1;
- *Auc*, our online auction algorithm presented in Section V-C, i.e., $\mathcal{A}_{\text{online}}$ combined with $\mathcal{A}_{\text{round}}$ in Algorithm 3;
- *AucBS*, the online auction algorithm with the improved scale-down factor, i.e., adding the binary search in Algorithm 4 to the auction algorithm $\mathcal{A}_{\text{round}}$ in $\mathcal{A}_{\text{online}}$;
- *AucExt*, the improved online auction algorithm with the improved scale-down factor, i.e., adding the binary search in Algorithm 4 to $\mathcal{A}_{\text{round}}$ in the improved online auction algorithm $\mathcal{A}'_{\text{online}}$ in Algorithm 5.

We compare these algorithms in different settings, based on the ratio between the offline optimal social welfare derived by solving (1) exactly and the overall social welfare produced by each online algorithm over T rounds, which we refer to as the *offline/online ratio*. In each scenario, we repeat each experiment for 10 times to derive the average ratios.

We first compare the algorithms through varying the number of cloud users N from 300 to 3000, while fixing the number of rounds $T = 300$, as illustrated in Fig. 1(a). The offline/online ratio of *Auc* declines when N is large ($N > 2000$), which is consistent with our theoretical analysis in Theorem 3: The larger the scale of the cloud system, the larger the value of $C_{\min}^{(t)}$, and consequently the better offline/online ratio results. When users' truthful resource demands and valuations are assumed available for free, the pure online resource allocation algorithm, *Alloc*, achieves an offline/online ratio close to 1, which shows that our online algorithm framework together with the one-round resource allocation algorithm performs closely to the offline optimum in social welfare, if all the cloud users are cooperative. *AucBS* achieves a better offline/online ratio as compared to *Auc*, revealing the usefulness of our improved scale-down factor based on the binary-search Algorithm 4 in decomposing the fractional solution into better integer solutions in practical scenarios. *AucExt* only slightly outperforms *AucBs* in this average-case ratio since its improvement mainly lies in the theoretical worst-case competitive ratio.

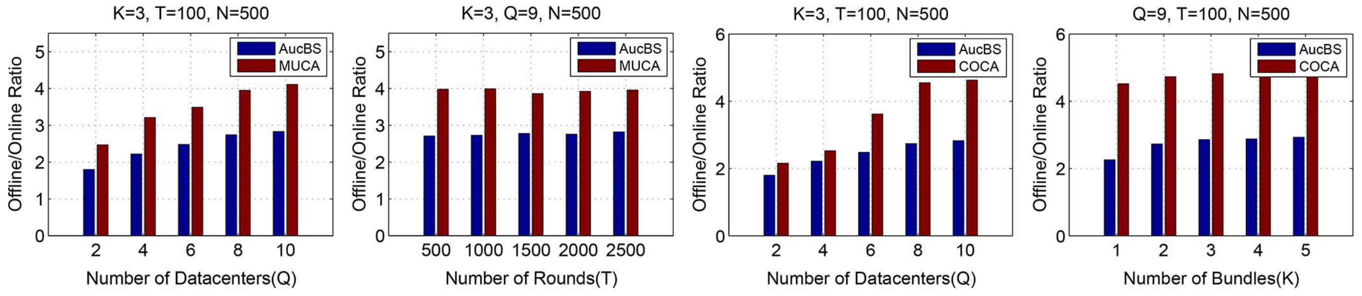
We next vary the total number of rounds T our system is running for while fixing the number of users to 500. Suppose each round is 1 h. We observe in Fig. 1(b) that the offline/online ratio of each algorithm always remains at similar levels, demonstrating the stable performance of our online algorithms regardless of the total number of rounds they are applied into.

We further evaluate the performance of *AucBS* when the number of bundles each user bids for in each round, K, and the number of datacenters, Q, vary. Fig. 1(c) shows that in general the performance of the improved online auction is better when the number of bundles is smaller. This can be explained as follows: The competitive ratio of our online auction (given in Theorem 5) is related to λ, the approximation ratio of the one-round resource allocation algorithm in Algorithm 2, which is further closely related to $\epsilon^{(t)}$. When K is smaller, $\epsilon^{(t)}$ is potentially smaller, and thus λ is smaller, leading to a lower competitive ratio of the online auction. In a practical cloud system, the K bundles that a user bids are typically different representations of the user's same resource demands in a



(a) (b) (c) (d)

Fig. 1. Performance of *Alloc*, *Auc*, *AucBS*, and *AucExt* (a) under different number of users, (b) under different number of rounds, (c) with different number of bundles, and (d) with different number of datacenters.



(a) (b) (c) (d)

Fig. 2. Performance comparison between *AucBS*, *MUCA*, and *COCA* (a) under different number of datacenters, (b) under different number of rounds, (c) under different number of datacenters, and (d) under different number of bundles.

time-slot, e.g., different bundles may specify different numbers of different types of VMs requested from different datacenters, which add up to a similar amount of each type of resource across different bundles, to serve the user's need in t. Therefore, we do not expect a large value of $\epsilon^{(t)}$ at any time. When the value of $\epsilon^{(t)}$ is capped (e.g., to 2.5 in our simulation settings), the competitive ratio is bounded even when K takes larger values, as shown by the similar offline/online ratios obtained when $K = 3, 4$, or 5, respectively, in Fig. 1(c).

Fig. 1(d) shows that when the total number of datacenters in the cloud system increases, the performance of our online auction degrades slightly because the approximation ratio λ of Algorithm 2 is larger when Q is larger. Nevertheless, we do not expect more than a few tens of datacenters in a real-world cloud system, and the offline/online ratio is still acceptable around 2.80 when Q is 10.

We note that the performance ratios obtained in our simulations are much smaller than the theoretical ratios computed based on Theorem 5, in the range of 6–7 under the same settings as used in our simulations. This promises the good performance of our online auction algorithm in practice.

B. Comparison to Existing Auction Mechanisms

Now we compare our online auction algorithm *AucBS* to another auction algorithm *MUCA* [6]. The settings in *MUCA* are the most similar in the existing literature, which however is a one-round multi-unit combinatorial auction, where each user only demands one resource bundle ($K = 1$). The main idea of *MUCA* is to calculate a virtual price for each bundle based on the amount of resource consumed by a bundle and a randomly assigned unit resource price, and to choose the winning bundle as the one with the highest cost efficiency, i.e., the highest valuation with the lowest virtual price. The competitive ratio of *MUCA* is $O(\sqrt{D})$, where D is the total number of available VM instances. In order to do a fair comparison between *MUCA* and our online algorithm, we run *MUCA* as a subroutine instead of A_{round} in the same online algorithm framework $\mathcal{A}_{\text{online}}$ in Algorithm 1; for multiple bundles submitted by one user, the *MUCA* subroutine greedily chooses the most cost-efficient available bundle.

Fig. 2(a) compares *AucBS* and *MUCA* to different numbers of datacenters, Q. The performance of *AucBS* is always better than *MUCA*. With more datacenters, the offline/online ratio of *AucBS* increases slower than *MUCA*, exhibiting that our algorithm handles multiple datacenters better. The reason is that the competitive ratio of our algorithm mainly depends on λ, which increases with Q very slowly $(O(Q^{1/C_{\min}^{(t)}}))$, while the competitive ratio of *MUCA* is close to $O(\sqrt{Q})$, which increases significantly with Q.

Fig. 2(b) compares the performance of both algorithms when they run for different overall time. We observe that the performance of *AucBS* is again better and the ratios of both algorithms are quite stable with different time lengths.

Finally, we compare our algorithm *AucBS* to another online auction algorithm, *COCA* [4]. The main idea of *COCA* is to calculate an estimated payment for user's requested resources based on a pricing curve and accept users with positive utility as winners. The original COCA model focuses on one type of resource; for fair comparison, we extend this method to multiple types of resources and multiple resource bundles, create a pricing curve for each type of resource with a coefficient assigned following the unit price of the respective resource in real-world cloud services [34], and always choose the bundle with the largest utility for each user. Note that we simply follow the underlying pricing curve method of *COCA*, and assign coefficients in an intuitive fashion. In this way, the extended *COCA* algorithm retains similar characteristics (such as truthfulness), but its theoretical performance in social welfare is no longer guaranteed. Before presenting the simulation comparison, we compare the extended *COCA* algorithm and our *AucBS* from a theoretical perspective: the theoretical performance of *COCA* depends on the ratio between the highest and lowest unit valuations, while *AucBS* is not influenced by the distribution of user valuations. However, the performance of *AucBS* is negatively affected when T is small (e.g., when $T < 10$) since $B_{\max}$ becomes larger, while the performance of *COCA* is the same for any values of T.

We compare the offline/online ratio achieved by the two algorithms by taking the average over 10 times of run of each experiment. Fig. 2(c) shows that the advantage of *AucBS* is clearer at larger values of Q. This is because more datacenters result in more types of resources (CPU/RAM/disk at different datacenters are treated as different types of resources), and *COCA* handles multiple types of resources in a simple intuitive fashion, which becomes less efficient when the number of types becomes larger. Fig. 2(d) reveals that the performance of *AucBS* is again consistently better under different numbers of bundles, K.

VIII. Concluding Discussions

This work presents the first online combinatorial auction for the VM market in cloud computing. It advances the state-of-the-art of cloud auction design in that all previous VM auction mechanisms are either one-round only or simplify VMs into type-oblivious good (and hence circumvent the challenge imposed by combinatorial auctions). Our online auction comprises three components. First, we design an intuition-driven primal-dual algorithm for translating the online social welfare optimization problem into a series of one-round optimizations, incurring only a small additive penalty in competitive ratio. Second, we apply a randomized auction subframework that can translate a cooperative approximation algorithm to the one-round optimization into an auction. Third, we apply a greedy primal-dual algorithm that approximates the one-round social welfare optimization. We also propose two extensions to the basic online auction framework to further improve the competitive ratio. Our overall online VM auction guarantees a theoretical competitive ratio close to 3.30 in typical scenarios, and its design may shed light on similar auction problems in related settings.

Finally, we briefly discuss practical implementation concerns of dynamic VM provisioning in real-world cloud systems. Latencies for resource partitioning are incurred for VM provisioning, which are a common issue in IaaS cloud provisioning. To realistically reduce such latencies before a VM can be ready to run, a practical solution is to maintain pools of preassembled VMs with typical resource configurations (e.g., summarized according to historical user demands) and do hotplug of CPU/RAM/disk [35] upon user's requests to produce customized VMs. Hotplug of CPU and RAM is highly efficient, while the preparation for the hard disk requires longer time. Nevertheless, users' demands for disk capacity are typically more uniform than those for CPU and RAM, which allows the cloud provider to prepare some "standard" sizes of virtual disks in advance (e.g., 256 GB, 512 GB, etc.). In this way, the latency for dynamic VM provisioning can be reduced to as low as a few seconds, which is ignorable compared to the hours of VM rental periods. In addition, our work does not deal with server provisioning, according to the practical knowledge that decisions on switching servers on or off are typically made at much larger time intervals than VM provisioning, e.g., once per month or so in Amazon EC2 cloud according to discussions with Amazon employees. We plan to pursue auction mechanism design for two-time-scale decision making and resource allocation in our future work.

Appendix A
Proof of Theorem 1

We prove the correctness and the competitiveness of $\mathcal{A}_{\text{online}}$ by proving three claims.

1) At the end of the algorithm, it produces feasible solution for dual (2).
2) Let $P^{(t)}$ be the value of the objective function in (1) after tth iteration, $\Delta P^{(t)} = P^{(t)} - P^{(t-1)}$, the same for $\Delta D^{(t)}$ in dual (2). Then, $\mathcal{A}_{\text{online}}$ satisfies $\Delta D^{(t)} \leq (\nu + \frac{1}{\gamma-1})\Delta P^{(t)}$, at any round.
3) The algorithm produces an almost feasible solution for primal (1). Specifically, its outputs $y_{n,k}^{(t)}$ satisfy (1a), (1c), and (1d). For constraint (1b), we achieve a slightly weaker property : For all user $n \in [N]$, $\sum_{t\in[T]}\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)} \leq B_n(1+B_{\max})$

Proof of (1): Since $w_{n,k}^{(t)} \geq b_{n,k}^{(t)}(1-x_n^{(t-1)})$ no matter whether $x_n^{(t-1)} < 1$ or $x_n^{(t-1)} \geq 1$, constraint (4a) guarantees $s_n^{(t)} + \sum_{r\in[R]}\sum_{q\in[Q]} c_{n,k,r}^{(t)} z_{q,r}^{(t)} \geq b_{n,k}^{(t)}(1-x_n^{(t-1)})$ Also notice $x_n^{(t)}$ is nondecreasing with t, so (2a) holds.

Proof of (2): At time t, $\Delta P^{(t)} = \sum_{n\in\mathcal{N}} b_{n,k_n}^{(t)}$. Substitute $\sum_{n\in\mathcal{N}} b_{n,k_n}^{(t)}(1-x_n^{(t-1)})$ for p, and we get claim (2).

Proof of (3): Constraints (1a), (1c), (1d) are guaranteed by the constraints in (3). In order to analyze the property about constraint (1b), we prove the following:

$$x_n^{(t')} \geq \frac{1}{\gamma-1}\left(\gamma^{\frac{\sum_{t=1}^{t'}\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)}}{B_n}} - 1\right), \quad 0 \leq t'' \leq T. \tag{8}$$

We prove (8) by induction. Equation (8) holds for $t' = 0$ apparently. Suppose it holds for $t'-1$, then for t': If $n \notin \mathcal{N}$, the inequality holds since both sides are still the same value at time $t'-1$. If $n \in \mathcal{N}$: $x_n^{(t)} = \frac{1}{\gamma-1}\left(\gamma^{\frac{\sum_{t=1}^{t'-1}\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)}}{B_n}}(1+\frac{b_{n,k_n}^{(t)}}{B_n}) - 1\right)$ Comparing this to our target (8), obviously we only need to show: $1 + \frac{b_{n,k_n}^{(t)}}{B_n} \geq \gamma^{\frac{b_{n,k_n}^{(t)}}{B_n}}$. We utilize the inequality: $\frac{\ln(1+x)}{x} \geq \frac{\ln(1+y)}{y}, \forall 0 \leq x \leq y \leq 1$ Since $\frac{b_{n,k_n}^{(t)}}{B_n} \leq B_{\max}$, we prove (8). Now we utilize the inequality (8) to prove claim (3). For some user n, suppose t' is the first time $\sum_{t=1}^{t'}\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)} \geq B_n$. Then by (8), $x_n^{(t')} \geq 1$. The algorithm never gives user n any new bundles once $x_n \geq 1$, since the weight $w_{n,k}^{(t)}$ in $\mathcal{A}_{\text{round}}$ will be set to 0. Hence, $\sum_{t=1}^{T}\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)} = \sum_{t=1}^{t'}\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)}$. We know t' is the first time user n's total winning bids exceeding its budget B_n and finishes the proof of claim (3).

Since the increment of valuation is the minimum between user's valuation and his remaining budget, total social welfare should be at least $P^{(T)}/(1+B_{\max})$. By claim 2: $\Delta D^{(t)} \leq (\nu$

$+\frac{1}{\gamma-1})\Delta P^{(t)}$, and recall $P^{(0)} = D^{(0)} = 0$, we have $D^{(T)} \leq (\nu + \frac{1}{\gamma-1})P^{(T)}$. Thus, the social welfare is at least $D^{(T)}/[(1+B_{\max})(\nu+\frac{1}{\gamma-1})]$. By duality, the approximation ratio of $\mathcal{A}_{\text{online}}$ is $(1+B_{\max})(\nu+\frac{1}{\gamma-1})$.

Appendix B
Proof of Theorem 2

Suppose user n's bid is $b_{n,k}^{(t)}$, then we can calculate the value of $w_{n,k}^{(t)}$ by definition. We omit this calculation process in our proof and directly assume that user n submits bid $w_{n,k}^{(t)}$ and other users $n' \neq n$ submit bids $w_{n',k}^{(t)}$. Then, according to the payment rule $\Pi_n^{(t)}$, user n's utility can be calculated as: $u_n^{(t)} = \sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} w_{n,k}^{(t)} - \widetilde{V}_{-n}^{(t)}$. $y_{n,k}^{(t)F}$ is calculated by maximizing $\sum_{n,k} y_{n,k}^{(t)} w_{n,k}^{(t)}$, which is the total social welfare. Hence, $\sum_{n,k} y_{n,k}^{(t)F} w_{n,k}^{(t)}$ is greater than $\widetilde{V}_{-n}^{(t)}$ because the latter is the maximum social welfare with the same amount of resources and one less user. Thus, $u_n^{(t)} \geq 0$. Next we compare the utility under the truthful bid and a false bid. Suppose user n submits a false bid $\widetilde{w}_{n,k}^{(t)}$. Then, the fractional allocation decision becomes $\widetilde{y}_{n,k}^{(t)F}$. His utility under false bid is calculated similarly:

$$\widetilde{u}_n^{(t)} = \sum_{k\in[K]} \widetilde{y}_{n,k}^{(t)F} w_{n,k}^{(t)} - \widetilde{V}_{-n}^{(t)} + \sum_{n'\neq n}\sum_{k\in[K]} \widetilde{y}_{n',k}^{(t)F} w_{n',k}^{(t)}.$$

The difference of these two utilities is

$$u_n^{(t)} - \widetilde{u}_n^{(t)} = \sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} w_{n,k}^{(t)} - \sum_{n\in[N]}\sum_{k\in[K]} \widetilde{y}_{n,k}^{(t)F} w_{n,k}^{(t)}.$$

Again, $y_{n,k}^{(t)F}$ maximizes social welfare, so $\sum_{n,k} y_{n,k}^{(t)F} w_{n,k}^{(t)} \geq \sum_{n,k} \widetilde{y}_{n,k}^{(t)F} w_{n,k}^{(t)}$, and $u_n^{(t)} - \widetilde{u}_n^{(t)} \geq 0$.

Appendix C
Proof of Theorem 3

Constraints (3a) and (3c) are obviously never violated. We study the constraint (3b). Suppose in iteration τ, $w_{n^*,k(n^*)}$ is the first bid that violates constraint (3b) when added to the allocation. This means $\exists q \in [Q], r \in [R]$ such that $\sum_{n\in\mathcal{N}} c_{n,k(n),r,q} \leq A_{q,r} < c_{n^*,k(n^*),r,q} + \sum_{n\in\mathcal{N}} c_{n,k(n),r,q}$. We know $C_{q,r} \geq c_{n^*,k(n^*),r,q}$, so $\sum_{n\in\mathcal{N}} c_{n,k(n),r,q} \geq A_{q,r} - C_{q,r}$. Hence, $\frac{\sum_{n\in\mathcal{N}} c_{n,k(n),r,q}}{A_{q,r}-C_{q,r}} \geq 1$. $A_{q,r} z_{q,r}^{\tau-1} = z_{\text{base}}^{\frac{\sum_{n\in\mathcal{N}} c_{n,k(n),r,q}}{A_{q,r}-C_{q,r}}} \geq z_{\text{base}}$, where $z_{q,r}^{\tau-1}$ represents the value of $z_{q,r}$ before iteration τ. Thus, constraint (3b) is never violated. The feasibility of the dual (4) is more complicated because the solution is not always feasible during the iterations. However, the dual variables are feasible after scaling down, which is described in the following lemma.

Lemma 2: If $(s_n^{\tau-1}, z_{q,r}^{\tau-1})$ is the dual solution before iteration τ, then $(s_n^{\tau-1}, \epsilon f(z,n^*,k(n^*)) z_{q,r}^{\tau-1})$ is a feasible solution to the dual (4), where $f(z,n,k)$ is defined as $f(z,n,k) = w_{n,k}/\sum_{q\in[Q]}\sum_{r\in[R]} c_{n,k,r,q} z_{q,r}$, and z is the set of $z_{q,r}^{\tau-1}$, n^* is the selected user in iteration τ.

Proof: We have $\forall n \in [N], k \in [K]$, $w_{n,k(n)} \geq w_{n,k}$. Note that we choose the n^* by choosing the maximum ratio of $w_{n,k}$ and a summation. Therefore, $f(z,n^*,k(n^*)) \geq \frac{w_{n,k(n)}}{\sum_{q\in[Q]}\sum_{r\in[R]} c_{n,k(n),r,q} z_{q,r}^{\tau-1}}$, and $f(z,n^*,k(n^*))\sum_{q\in[Q]}\sum_{r\in[R]} c_{n,k(n),r,q} z_{q,r}^{\tau-1} \geq w_{n,k(n)}, \forall n \notin \mathcal{N}$. So: $\forall n \notin \mathcal{N}, k \in [K]$, $\epsilon f(z,n^*,k(n^*))\sum_{q\in[Q]}\sum_{r\in[R]} c_{n,k(n),r,q} z_{q,r}^{\tau-1} \geq w_{n,k}$. Thus, $(s_n^{\tau-1}, \epsilon f(z,n^*,k(n^*)) z_{q,r}^{\tau-1})$ is feasible to dual (4). ■

Now we are ready to prove the final conclusion of Theorem 3. Let $d_1^\tau = \sum_{n\in[N]} s_n$, $d_2^\tau = \sum_{q\in[Q]}\sum_{r\in[R]} A_{q,r} z_{q,r}^\tau$. Let d^* be the optimal solution to the dual (4). Let (n^τ, k^τ) be the bundle selected in iteration τ. Totally ω iterations are executed.

Case 1: $\mathcal{A}_{\text{round}}$ stops at iteration ω where $\mathcal{N} = [N]$ and $\sum_{q\in[Q]}\sum_{r\in[R]} A_{q,r} z_{q,r} < z_{\text{base}}$. The solution is optimal.

Case 2: $\mathcal{A}_{\text{round}}$ stops at iteration ω where $d_2^\omega \geq z_{\text{base}}$, and $\exists \tau \leq \omega$, such that $\lambda \geq d^*/d_1^{\tau-1}$. Then, it is a λ-approximation solution since $d_1^{\tau-1} = p^{\tau-1}$ and d_1^τ is nondecreasing of τ.

Case 3: $\mathcal{A}_{\text{round}}$ stops at iteration ω where $d_2^\omega \geq z_{\text{base}}$, and $\lambda < d^*/d_1^{\tau-1}, \forall \tau \leq \omega$. For any iteration τ, $d_2^\tau = \sum_{q\in[Q]}\sum_{r\in[R]} A_{q,r} z_{q,r}^\tau$. Define $\delta = (\frac{A_{q,r}}{C_{q,r}} - 1)(z_{\text{base}}^{1/(\frac{A_{q,r}}{C_{q,r}}-1)} - 1$. Then, $d_2^\tau = \sum_{q\in[Q]}\sum_{r\in[R]} A_{q,r} z_{q,r}^{\tau-1}(1 + \frac{\delta}{\frac{A_{q,r}}{C_{q,r}}-1})^{(\frac{c_{n^\tau,k^\tau,q,r}}{A_{q,r}-C_{q,r}})}$. We utilize the inequality: $(1+a)^x \leq 1 + ax, \forall x \in [0,1]$, $d_2^\tau \leq \sum_{q\in[Q]}\sum_{r\in[R]} A_{q,r} z_{q,r}^{\tau-1}(1 + \frac{\delta}{\frac{A_{q,r}}{C_{q,r}}-1} \cdots \frac{c_{n^\tau,k^\tau,q,r}}{(A_{q,r}-C_{q,r})})$. Let $\Delta = \max_{q\in[Q],r\in[R]} \frac{\delta A_{q,r}}{A_{q,r}-C_{q,r}}$, then: $d_2^\tau \leq d_2^{\tau-1} + \Delta\sum_{q\in[Q]}\sum_{r\in[R]}(c_{n^\tau,k^\tau,q,r} z_{q,r}^{\tau-1})$. Note that $\frac{\delta A_{q,r}}{A_{q,r}-C_{q,r}}$ is nonincreasing of $\frac{A_{q,r}}{z_{q,r}}$. Therefore, $\frac{\delta A_{q,r}}{A_{q,r}-C_{q,r}}$ reaches its maximum when $\frac{A_{q,r}}{z_{q,r}} = C_{\min}$. Recall the definition of $f(z,n^\tau,k^\tau)$, we have $\sum_{q\in[Q]}\sum_{r\in[R]}(c_{n^\tau,k^\tau,q,r} z_{q,r}^{\tau-1}) = \frac{w_{n^\tau,k^\tau}}{f(z,n^\tau,k^\tau)}$. Here is a corollary of Lemma 2: $f(z,n^\tau,k^\tau) \geq \frac{d^*-d_1^{\tau-1}}{\epsilon d_2^{\tau-1}}$. Hence, $\frac{1}{f(z,n^\tau,k^\tau)} \leq \frac{\epsilon\lambda d_2^{\tau-1}}{(\lambda-1)d^*}$. We utilize the inequality: $1 + x \leq e^x, \forall x \geq 0$ here, and repeat this process: $d_2^\omega \leq d_2^0 e^{(\epsilon p \frac{\lambda\Delta}{(\lambda-1)d^*})}$. Note that $d_2^0 = QR$ and $d_2^\omega \geq z_{\text{base}}$, so $z_{\text{base}} \leq QRe^{(\epsilon p \frac{\lambda\Delta}{(\lambda-1)d^*})}$. Hence, $d^*/p \leq \frac{\epsilon\lambda\Delta}{(\lambda-1)(C_{\min}-1)}$. Also note that $\frac{\epsilon\lambda\Delta}{(\lambda-1)(C_{\min}-1)} = \lambda$. According to the weak duality, this finishes the proof of Theorem 3.

Appendix D
Separation Oracle and Proof of Lemma 1

First, we describe the separation oracle. In each iteration of the ellipsoid method, a possible solution of (7) $\{v_{n,k}^{(t)}\}, \tau$ is generated, and is given as the input of the seperation oracle. The separation oracle we present in Algorithm 6 judges whether this solution is feasible. If it is not feasible, the oracle returns a conflict constraint as the separation plane, i.e., a set of $y_{n,k}^{(t)l}$. We use

$$\frac{1}{\lambda}\sum\nolimits_{n\in[N]}\sum\nolimits_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau \geq 1$$

as the separation plane. If the objective value is larger than 1, we need to find a set of $y_{n,k}^{(t)l}$ that satisfies $\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} \geq 1 - \tau$. Remember the input $v_{n,k}^{(t)}$ and τ makes the objective value larger than 1: $\frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)} \geq 1 - \tau$. Hence, we find a conflict constraint if we can find $y_{n,k}^{(t)l}$ satisfies $\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} \geq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)}$.

Algorithm 6 Separation Oracle

```
Require: input $v_{n,k}^{(t)}, \tau$
1: If $\frac{1}{\lambda}\sum_n \sum_k y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau = 1$, Then return "YES"
2: If $\frac{1}{\lambda}\sum_n \sum_k y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau < 1$, Then return "NO" with separation plane $\frac{1}{\lambda}\sum_n \sum_k y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau \geq 1$
3: If $\frac{1}{\lambda}\sum_n \sum_k y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau > 1$, Then
4:     $\widetilde{v}_{n,k}^{(t)} = \max\{0, v_{n,k}^{(t)}\}$
5:     Run $\mathcal{A}_{\text{round}}$ with input $\widetilde{v}_{n,k}^{(t)}, c_{n,k,r,q}^{(t)}, A_{q,r}^{(t)}$, get output $\widetilde{y}_{n,k}^{(t)l}$. Set $y_{n,k}^{(t)l} = \widetilde{y}_{n,k}^{(t)l}$ if $v_{n,k}^{(t)} \geq 0$, and 0 otherwise.
6:     Return "NO" and $\sum_n \sum_k y_{n,k}^{(t)l} v_{n,k}^{(t)} + \tau \leq 1$
7: EndIf
```

The following is the detailed proof of Lemma 1. We first show that for any $\{v_{n,k}^{(t)}\}$, we can find in polynomial time a feasible integer allocation $y_{n,k}^{(t)l}$ such that $\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} \geq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)}$. Note the $v_{n,k}^{(t)}$ here can be negative value, so we cannot invoke $\mathcal{A}_{\text{round}}$ directly. So we define $\widetilde{v}_{n,k}^{(t)}$. Then, we use $\mathcal{A}_{\text{round}}$, with input vector $\widetilde{v}_{n,k}^{(t)}$ to get $\widetilde{y}_{n,k}^{(t)l}$ satisfying $\sum_{n\in[N]}\sum_{k\in[K]} \widetilde{y}_{n,k}^{(t)l}\widetilde{v}_{n,k}^{(t)} \leq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)}$. Also note that $y_{n,k}^{(t)l}$ satisfies $\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} \geq \sum_{n\in[N]}\sum_{k\in[K]} \widetilde{y}_{n,k}^{(t)l}\widetilde{v}_{n,k}^{(t)}$. Thus, we find an integer solution $y_{n,k}^{(t)l}$. Next we show the optimal value of (7), and hence of (6) is exactly 1. Here is a feasible solution with value 1: $\tau = 1, v_{n,k}^{(t)} = 0, \forall n \in [N], k \in [K]$, so the optimal value is at least 1. Then, we claim that it is at most 1: suppose $\frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau > 1$. Then, we can find an integer solution using the previous method such that $\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} + \tau \geq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)} + \tau \geq 1$, which contradicts constraint (7a). Finally we show the function of our designed oracle is correct. Two cases are obvious: objective value equals 1 and smaller than 1. When larger than 1, the oracle generates $\widetilde{v}_{n,k}^{(t)}$ and calls $\mathcal{A}_{\text{round}}$. The correctness of this method has been discussed. Therefore, this oracle solves (7) as we expect.

Appendix E
Proof of Theorem 4

We calculate the expectation of his utility: $U_n^{(t)} = \sum_{l\in\boldsymbol{L}}(\beta_l \sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)l} - \Pi_n^{(t)l}) = (\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)F} - \Pi_n^{(t)F})/\lambda$. His utility cannot be increased by false bid. Hence, the scaled-down auction is also truthful. The social welfare under the fractional VCG auction is $\sum_{n\in[N]}\min\{\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)F}, B_n\}$. Now we calculate the expected social welfare under randomized auction: $\sum_{l\in\boldsymbol{L}}(\beta_l \sum_{n\in[N]}\min\{\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)l}, B_n\}) \geq \frac{\sum_{n\in[N]}\min\{\sum_{k\in[K]} b_{n,k}^{(t)} y_{n,k}^{(t)F}, B_n\}}{\lambda(1+B_{\max})}$. This proves the competitive ratio since the social welfare under optimal fractional solution is larger than under integer solution. Next we show the individual rationality. First note that the definition $\Pi_n^{(t)l} = \Pi_n^{(t)F} \cdot \frac{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)l}}{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)F}}$ guarantees $\sum_{l\in\boldsymbol{L}}\beta_l \Pi_n^{(t)l} = \Pi_n^{(t)F}/\lambda$. Then, we calculate user n's utility under random choice $lu_n^{(t)} = \sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)l} - \Pi_n^{(t)F} \cdot \frac{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)l}}{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)F}}$. In order to prove $u_n^{(t)} \geq 0$, we only need to prove $\Pi_n^{(t)F} \leq \sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)F}$.

Appendix F
Proof of Theorem 5

The only difference between Theorems 5 and 1 is we introduce randomness here. Recall the proof of Theorem 1, the only claim affected by randomness is claim (2). We analyze the expectation of the increment on the primal and dual $E[\Delta P]$ and $E[\Delta D]$. At time t, $E[\Delta P^{(t)}] = \sum_{n\in[N]}\sum_{k\in[K]} E[y_{n,k}^{(t)} b_{n,k}^{(t)}]$. $E[\Delta D^{(t)}] \leq (\lambda(1+B_{\max}) + \frac{1}{\gamma-1})E[\Delta P^{(t)}]$, which finishes our proof.

Appendix G
Proof of Theorem 6

Suppose with scale-down ratio λ_1, $y_{n,k}^{(t)F}$ is decomposed into a set of integer allocations $y_{n,k}^{(t)l}$. Then, $\sum_{l\in\boldsymbol{L}}\beta_l y_{n,k}^{(t)l} = y_{n,k}^{(t)F}/\lambda_1, \forall n \in [N], k \in [K]$. We obtain ratio λ_2 by adding a "zero" allocation into the decomposition set $\boldsymbol{L}: y_{n,k}^{(t)*} = 0$. The possibilities are adjusted: $\beta_l' = \beta_l \cdot \frac{\lambda_1}{\lambda_2}$. The possibility of the "zero" allocation is $1 - \sum_{l\in\boldsymbol{L}}\beta_l'$. We can verify that the new decomposition set, with new possibilities β_l' is a decomposition for ratio λ_2, $\sum_{l\in\boldsymbol{L}}\beta_l' y_{n,k}^{(t)l} = y_{n,k}^{(t)F}/\lambda_2, \forall n \in [N], k \in [K]$.

Appendix H
Proof of Theorem 7

First we prove $x_n^{(T)}$ to be at least $x_n^{\boldsymbol{s}} = g_n/(\gamma^{1-g_n} - (1-g_n))$. Whenever $x_n < x_n^{\boldsymbol{s}}$, we update x_n by: $x_n^{(t)} = x_n^{(t-1)} + (x_n^{\boldsymbol{s}} + \frac{1-g_n}{\gamma^{1-g_n}-(1-g_n)})\frac{b_{n,k_n}^{(t)}}{B_n}$. Its total valuation is $\sum_{t\in[T]} b_{n,k_n}^{(t)} = g_n B_n$, and we have $x_n^{(T)} = x_n^{\boldsymbol{s}}$. Then, we prove the three claims just similar to the proof in Appendix A. Proofs of the first two claims are trivially similar. Now we prove the third claim by inductively prove the following inequality relation: Suppose user n has spent g_n' fraction of budget. If $g_n' \leq g_n$, $x_n \geq g_n'(x_n^{\boldsymbol{s}} + \frac{1-g_n}{\gamma^{1-g_n}-(1-g_n)})$. If $g_n' > g_n$, $x_n \geq x_n^{\boldsymbol{s}} c^{g_n'-g_n} + \frac{(1-g_n)(c^{g_n'-g_n}-1)}{\gamma^{1-g_n}-(1-g_n)}$. The first case is proved by monotonicity. The second case can be proved by induction, similar to Appendix A.

References

[1] "Amazon Elastic Compute Cloud," [Online]. Available: http://aws.amazon.com/ec2/

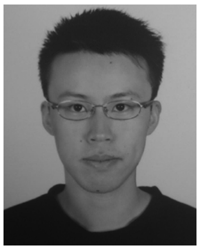
[2] "Windows Azure," [Online]. Available: http://www.windowsazure.com/

[3] H. Fu, Z. Li, and C. Wu, "Core-selecting auction design for dynamically allocating heterogeneous VMs in cloud computing," in *Proc. IEEE CLOUD*, 2014, pp. 152–159.

[4] H. Zhang *et al.*, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," in *Proc. IEEE INFOCOM*, 2013, pp. 1510–1518.

[5] "Amazon EC2 spot instances," [Online]. Available: http://aws.amazon.com/ec2/spot-instances/

[6] Q. Wang, K. Ren, and X. Meng, “When cloud meets eBay: Towards effective pricing for cloud computing,” in *Proc. IEEE INFOCOM*, 2012, pp. 936–944.
[7] S. Zaman and D. Grosu, “Combinatorial auction-based mechanisms for VM provisioning and allocation in clouds,” in *Proc. IEEE/ACM CCGrid*, 2012, pp. 729–734.
[8] L. Zhang, Z. Li, and C. Wu, “Dynamic resource provisioning in cloud computing: A randomized auction approach,” in *Proc. IEEE INFOCOM*, 2014, pp. 433–441.
[9] W. Wang, B. Liang, and B. Li, “Revenue maximization with dynamic auctions in IaaS cloud markets,” in *Proc. IEEE ICDCS*, 2013, pp. 1–6.
[10] N. Buchbinder, K. Jain, and J. S. Naor, “Online primal-dual algorithms for maximizing ad-auctions revenue,” in *Proc. Annu. Eur. Symp.*, 2007, pp. 253–264.
[11] R. Lavi and C. Swamy, “Truthful and near-optimal mechanism design via linear programming,” in *Proc. IEEE FOCS*, 2005, pp. 595–604.
[12] Y. Zhu, B. Li, and Z. Li, “Truthful spectrum auction design for secondary networks,” in *Proc. IEEE INFOCOM*, 2012, pp. 873–881.
[13] C. Wu, Z. Li, X. Qiu, and F. C. M. Lau, “Auction-based P2P VoD streaming: Incentives and optimal scheduling,” *Trans. Multimedia Comput., Commun., Appl.*, vol. 8, no. 15, 2012, Art. no 14.
[14] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos, “TRAC: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing,” in *Proc. IEEE INFOCOM*, 2014, pp. 1231–1239.
[15] W. Vickrey, “Counterspeculation, auctions, and competitive sealed tenders,” *J. Finance*, vol. 16, no. 1, 1961, DOI: 10.1111/j.1540-6261.1961.tb02789.x.
[16] M. H. Rothkopf, A. Pekeč, and R. M. Harstad, “Computationally manageable combinational auctions,” *Manage. Sci.*, vol. 44, no. 8, pp. 1131–1147, 1998.
[17] A. Mu'Alem and N. Nisan, “Truthful approximation mechanisms for restricted combinatorial auctions,” *Games Econ. Behav.*, vol. 64, no. 2, pp. 612–631, 2008.
[18] D. Lehmann, L. I. Oćallaghan, and Y. Shoham, “Truth revelation in approximately efficient combinatorial auctions,” *J. ACM*, vol. 49, no. 5, pp. 577–602, 2002.
[19] L. Mashayekhy, M. M. Nejad, D. Grosu, and A. V. Vasilakos, “Incentive-compatible online mechanism for resource provisioning and allocation in cloud,” in *Proc. IEEE CLOUD*, 2014, pp. 312–319.
[20] G. Shanmuganathan, A. Gulati, and P. Varman, “Defragmenting the cloud using demand-based resource allocation,” in *Proc. ACM SIGMETRICS*, 2013, pp. 67–80.
[21] N. Nisan, *Algorithmic Game Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2007.
[22] R. Lavi and N. Nisan, “Competitive analysis of incentive compatible on-line auctions,” in *Proc. ACM EC*, 2000, pp. 233–241.
[23] M. Alicherry and T. Lakshman, “Network aware resource allocation in distributed clouds,” in *Proc. IEEE INFOCOM*, 2012, pp. 963–971.
[24] S. T. Maguluri, R. Srikant, and L. Ying, “Stochastic models of load balancing and scheduling in cloud computing clusters,” in *Proc. IEEE INFOCOM*, 2012, pp. 702–710.
[25] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, “Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions,” *Proc. IEEE*, vol. 102, no. 1, pp. 11–31, Jan. 2014.
[26] L. Wang *et al.*, “GreenDCN: A general framework for achieving energy efficiency in data center networks,” *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4–15, Jan. 2014.
[27] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, “Joint virtual machine assignment and traffic engineering for green data center networks,” *Perform. Eval. Rev.*, vol. 41, no. 3, pp. 107–112, 2014.
[28] F. Xu, F. Liu, L. Liu, H. Jin, and B. Li, “iAware: Making live migration of virtual machines interference-aware in the cloud,” *IEEE Trans. Comput.*, vol. 63, no. 12, pp. 3012–3025, Dec. 2013.
[29] Z. Zhou *et al.*, “On arbitrating the power-performance tradeoff in SaaS clouds,” in *Proc. IEEE INFOCOM*, 2013, pp. 872–880.
[30] J. Guo, F. Liu, D. Zeng, J. C. Lui, and H. Jin, “A cooperative game based allocation for sharing data center networks,” in *Proc. IEEE INFOCOM*, 2013, pp. 2139–2147.
[31] N. Cherfi and M. Hifi, “A column generation method for the multiple-choice multi-dimensional Knapsack problem,” *Comput. Optimiz. Appl.*, vol. 46, no. 1, pp. 51–73, 2010.
[32] 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.
[33] “Google cluster data,” [Online]. Available: https://code.google.com/p/googleclusterdata/
[34] “CloudSigma: Cloud servers on a powerful IaaS platform,” [Online]. Available: http://www.cloudsigma.com
[35] “KVM CPU Hotplug,” [Online]. Available: http://www.linux-kvm.org/page/CPUHotPlug



Weijie Shi (S’14) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012, and is currently pursuing the Ph.D. degree in computer science at the University of Hong Kong, Hong Kong.

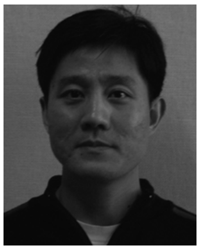
His research interests include cloud computing and mechanism design.

Linquan Zhang (S’13) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 2010, and the M.Phil. degree in computer science from the University of Hong Kong, Hong Kong, in 2012, and is currently pursuing the Ph.D. degree in computer science at the University of Calgary, Calgary, AB, Canada.

His research interests are mainly in cloud computing, network optimization, and game theory.

Chuan Wu (M’08) received the B.E. and M.E. degrees in computer science and technology from Tsinghua University, Beijing, China, in 2000 and 2002, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2008.

She is currently an Associate Professor with the Department of Computer Science, The University of Hong Kong, Hong Kong. Her research interests include cloud computing and online/mobile social network.



Zongpeng Li (SM’12) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 1999, and the M.S. degree in computer science and Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2001 and 2005, respectively.

He is currently an Associate Professor with the Department of Computer Science, University of Calgary, Calgary, AB, Canada. His research interests are in computer networks, particularly in network optimization, multicast algorithm design, network game theory, and network coding.

Francis C. M. Lau (M’93–SM’03) received the Ph.D. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1986.

He has been a faculty member with the Department of Computer Science, The University of Hong Kong, Hong Kong, since 1987, where he served as the Department Chair from 2000 to 2005. He was an Honorary Chair Professor with the Institute of Theoretical Computer Science, Tsinghua University, Beijing, China, from 2007 to 2010. His research interests include computer systems and networking, algorithms, HCI, and application of IT to arts.