An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing

Weijie Shi
Dept. of Computer Science
The University of Hong Kong
wjshi@cs.hku.hk

Linquan Zhang
Dept. of Computer Science
University of Calgary
linqzhan@ucalgary.ca

Chuan Wu
Dept. of Computer Science
The University of Hong Kong
cwu@cs.hku.hk

Zongpeng Li
Dept. of Computer Science
University of Calgary
zongpeng@ucalgary.ca

Francis C.M. Lau
Dept. of Computer Science
The University of Hong Kong
fcmlau@cs.hku.hk

ABSTRACT

Auction mechanisms have recently attracted substantial attention as an efficient approach to pricing and resource allocation in cloud computing. This work, to the authors' knowledge, represents the first online combinatorial auction designed in the cloud computing paradigm, which is general and expressive enough to both (a) optimize system efficiency across the temporal domain instead of at an isolated time point, and (b) 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 $\sim e+\frac{1}{e-1} \simeq 3.30$ 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 an additive loss of $\frac{1}{e-1}$ in competitive ratio, (2) a randomized auction sub-framework that applies primal-dual optimization for translating a centralized cooperative social welfare approximation algorithm into an auction mechanism, retaining a similar approximation ratio while adding truthfulness, and (3) a primal-dual update plus dual fitting algorithm for approximating the one-shot optimization with a ratio λ close to e. 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, as well as its three independent modules, can be instructive in auction design for other related problems.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies; Modeling techniques; I.1.2 [**Algorithms**]: Analysis of algorithms

General Terms

Algorithms; Design; Economics

Keywords

Cloud Computing; Combinatorial Auction; Resource Allocation; Pricing; Online Algorithms; Truthful Mechanisms

1. 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 [4], 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-VM usage. For example, Table 1 shows the available VM types at Amazon EC2, and their hourly prices at different datacenters. Despite their apparent simplicity, fixed-price policies are inherently lack of market agility and efficiency, failing to rapidly adapt to realtime 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 1: 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

Towards effectively discovering the market value of VMs, auction-based mechanisms have been at the focal point of recent literature on cloud resource allocation and pricing [9,

28]. For example, Spot Instance [2] is a first-step attempt to apply the auction mechanism on Amazon EC2, which was enhanced by subsequent work [24, 28]. A series of recent work further study auction mechanism design in cloud markets from different perspectives [27, 29, 25]. Unfortunately, all existing cloud auction mechanisms 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 is modelled more naturally by an online auction that incorporates the time dimension. Most cloud computing customers, enterprise or individual, are on a pre-allocated budget for a given time period (*e.g.*, a year or a month like that in an ad-auction [7]). 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 [29].
[Cloud auctions should be combinatorial.] A cloud computing job in practice often demands a bundle of heterogenous 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* [9], 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 heterogenous types are allocated in multiple consecutive time slots. The final result is an online auction framework that simultaneously guarantees the following properties. (i) 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. (ii) 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. (iii) 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 realtime user demand. (iv) 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 $\frac{1}{3.30}$ 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 sub-framework, 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 [13], and successfully applied in the literature of computer networking [30]. We also propose a new technique which can significantly improve the performance of this translation.

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.

In the rest of the paper, we discuss related work in Sec. 2, and define the problem model in Sec. 3. Sec. 4 and Sec. 5 present the online algorithm framework, the auction algorithm and the single round allocation algorithm. Simulations are presented in Sec. 6. Sec. 7 concludes the paper.

2. RELATED WORK

Resource allocation in computing and communication systems is a classic problem that has been extensively studied, including from a game theoretical view by analyzing the incentive compatibility of the allocation algorithm [11] [15]. An alternative approach is designing pricing mechanisms with maximized social welfare [19]. Auctions are mechanisms that combine these two approaches and simultaneously target both truthfulness and economic efficiency. Classic applications of auctions are found in a wide range of research areas, such as network bandwidth allocation [26] and wireless spectrum allocation [30].

The celebrated VCG mechanism [23] is a well known type of auctions. 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 [20], VCG becomes computationally infeasible. When polynomial-time approximation algorithms are applied to solving the underlying social welfare maximization 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* [14]. Another relatively new alternative is to resort to the LP decomposition technique [13], 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.* [24] 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. Zaman and Grosu [27] study the modeling of VM provisioning, but their model does not include dynamic VM assembling. Shanmuganathan *et al.* [21] introduce the concept of *computing resources bundles* in VM allocation. Zhang *et al.* [29] is among the first to study dynamic VM provisioning, and designs 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 [18]. 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 of a *supply curve* [12], as applied by Zhang *et al.* [28] 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 VMs, 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 [25], and their model also focuses on one type of VMs only.

There have been some proposals considering bandwidth allocation in datacenters. Ballani *et al.* [5] propose Oktopus, a system that provides virtual network abstractions to the cloud users. Bansal *et al.* [6] discuss the allocation of VMs while considering network congestion. Meng *et al.* [16] take both intra- and inter-datacenter traffic into consideration in deciding the placement of VMs. Allocating resources including both network bandwidth and computing resources, such as CPU and storage, makes the problem more difficult than focusing on only one of them. We leave the possible discussions on bandwidth auctions as future work.

3. PROBLEM MODEL

3.1 The 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 virtual machines (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 k-th bundle in time slot t, and $b_{n,k}^{(t)}$ be its valuation for this k-th bundle in t. The k-th bundle in the bid of user n in the auction at t is described by a $(M \times Q + 1)$-tuple of elements $d_{n,k,m,q}^{(t)}, \forall m \in [M], \forall q \in [Q]$, 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 auction, upon receiving users' bids, the cloud provider computes its resource allocation and produces the auction results, $y_{n,k}^{(t)} \in \{0, 1\}, \forall n \in [N], \forall k \in [K]$, where $y_{n,k}^{(t)}$ is 1 if user n wins bundle k and 0 otherwise, as well as user n's payment $\Pi_n^{(t)}$, for actually acquiring VMs in its winning

[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 2: 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 kth bundle at t
$d_{n,k,m,q}^{(t)}$	# of type-m VM at dc q in n's kth bundle at t
$c_{n,k,r,q}^{(t)}$	amount of resource r at dc q in n's kth bundle at t
B_n	user n's total budget
$y_{n,k}^{(t)}$	user n wins its kth 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 kth bundle at time t
B_{max}	max ratio: a single bundle bid / a user's budget
γ	$(1 + B_{max})^{1/B_{max}}$

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, *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 Sec. 5. 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 pre-allocated 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 2.

3.2 The 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. (i) *Truthfulness* (Definition 1): Bidding true valuations is a dominant strategy at the users, and consequently, both bidding strategies and auction design are simplified. (ii) *Individual rationality*: Each bidder obtains a non-negative utility by participating in the auction in any time slot, *i.e.*, $u_n^{(t)} \geq 0$, $\forall n \in [N], \forall t \in [T]$. (iii) *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, $\sum_{t\in[T]}\sum_{n\in[N]}\sum_{k\in[K]} b_{n,k}^{(t)}y_{n,k}^{(t)} - \sum_{t\in[T]}\sum_{n\in[N]}\Pi_n^{(t)}$, which equals aggregated user valuation of the winning bundles (under truthful bidding), $\sum_{t\in[T]}\sum_{n\in[N]}\sum_{k\in[K]} b_{n,k}^{(t)}y_{n,k}^{(t)}$. Payment from the users and revenue received by the cloud provider 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 below an offline social welfare optimization problem which provides the "ideal" optimal resource allocation strategies for the cloud provider to address users' VM demands in the entire system lifespan $[T]$, assuming truthful bids are known. 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 k-th bundle.

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

subject to

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

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

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

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

Constraint (1a) specifies that each user can win at most one bundle each round. (1b) is the budget constraint at each user. (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 ignore 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 Sec. 4:

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

subject to

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

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

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 realtime resource allocation, which also guarantees truthful bidding. We achieve the goals in two steps. First, in Sec.4, we assume that a truthful auction mechanism to be carried out *in each time slot* is known, and guarantees an approximation ratio ν, and 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. Second, in Sec. 5, we design a single-round randomized auction, which achieves the approximation ratio of ν as well as individual rationality and truthfulness.

4. AN ONLINE ALGORITHM FRAMEWORK

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

4.1 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). In the optimization below, $w_{n,k}^{(t)}$, a reduced valuation of user n for bundle k from the actual valuation $b_{n,k}^{(t)}$ in its bid according to the level of its remaining budget, is used in the objective function. The rationale will be detailed in Sec. 4.2. 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, $\forall n, k, r, q$, the one-round optimization problem decides the optimal resource allocation $y_{n,k}^{(t)}, \forall n, k$, at t.

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

subject to

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

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

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

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

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

s.t.

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

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

The primal problem (3) is a special case of the multi-dimensional multiple-choice 0-1 knapsack problem [8], which is both NP-hard and more strongly, has no fully polynomial-time approximation schemes unless P=NP [10]. What we will pursue in $\mathcal{A}_{round}$ is an auction mechanism, which not only guarantees individual rationality and truthfulness, but also employs a primal-dual approximation algorithm that solves problem (3) and (4) to decide resource allocation in polynomial time with a small approximation ratio. We delay the discussion of the auction mechanism to Sec. 5, 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.

4.2 The Online Algorithm

When a good approximation algorithm for one-round resource allocation (with budget constraint relaxed) is in place, the difficulty of designing an online algorithm to achieve 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, arises from the budget constraint at each user. 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 resource allocation decisions at the cloud 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 Alg. 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 Lines 7 and 10 of 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. $x_n^{(t)}$ is increased if user n wins a new bundle in round t (Line 7) — thus user n's remaining budget decreases, and remains unchanged otherwise (Line 10). The increment in Line 7 is carefully computed (see proof of Thm. 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 (Line 13). In this way, the adjustment of $x_n^{(t)}$ in each round can be understood as the adjustment of the dual variable x_n towards an optimal solution to the offline dual problem (2).

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

```
1: $x_n^{(0)} \leftarrow 0, \forall n \in [N]$
2: // Loop for each time slot
3: for all $1 \le t \le T$ do
4:
   $$w_{n,k}^{(t)} = \begin{cases} 0 & \text{if } x_n^{(t-1)} \ge 1 \\ b_{n,k}^{(t)}(1 - x_n^{(t-1)}) & \text{otherwise} \end{cases}, \forall n \in [N], k \in [K].$$
5:    Run $\mathcal{A}_{round}$. Let $\mathcal{N}$ be the set of winning users, and
      $k_n$ be the index of their corresponding winning bundle,
      for each winning user $n \in \mathcal{N}$.
6:    for all $n \in \mathcal{N}$ do
7:
      $$x_n^{(t)} \leftarrow x_n^{(t-1)} \left(1 + \frac{b_{n,k_n}^{(t)}}{B_n}\right) + \frac{b_{n,k_n}^{(t)}}{B_n(\gamma - 1)}$$
8:    end for
9:    for all $n \notin \mathcal{N}$ do
10:       $x_n^{(t)} \leftarrow x_n^{(t-1)}$
11:   end for
12: end for
13: $x_n \leftarrow x_n^{(T)}, \forall n \in [N]$
```

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

Theorem 1. *If we can find an auction mechanism in $\mathcal{A}_{round}$ that carries out resource allocation in each round to produce feasible solutions for (3) and (4), and guarantees $\nu p \ge d$ (hence the competitive ratio of the auction algorithm is also ν), $\mathcal{A}_{online}$ 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 $d = \sum_{n\in[N]} s_n^{(t)} + \sum_{q\in[Q]} \sum_{r\in[R]} A_{q,r}^{(t)} z_{q,r}^{(t)}$ 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.

5. A RANDOMIZED AUCTION MECHANISM

We now present a randomized auction mechanism $\mathcal{A}_{round}$ which 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 VCG (Vickrey-Clarke-Groves) mechanism [23] 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 which 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 the following three subsections.

5.1 The Fractional VCG Auction

In the fractional VCG auction, the auctioneer solves the LP relaxation of (3) by relaxing constraint (3c) to $0 \le y_{n,k}^{(t)} \le 1, \forall n, k$, 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: $\Pi_n^{(t)F} = \widetilde{V}_{-n}^{(t)} - \sum_{n' \ne n} \sum_{k\in[K]} y_{n',k}^{(t)F} w_{n',k}^{(t)}$.

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:

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

Such a utility function is consistent with the social welfare calculation in the one-round allocation problem (3). In this way, a user's budget can potentially last longer, enabling its acquirement of a better bundle with the same consumption of budget at a later time, contributing to social welfare efficiency over all T rounds of auctions.

We show in Thm. 2 that under this utility function, bidding true valuations is the best strategy for each user in the fractional VCG auction. A non-negative utility is guaranteed for each bidder, based on VCG auction theory [23]. Due to space limit, most proofs in this section are omitted in this paper and we refer the interested readers to [22] for the details.

Theorem 2. *The fractional VCG auction which 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.*

5.2 Decomposing the Fractional Solution

Since fractional VM bundles are impractical in real-world cloud systems, we next decompose the fractional allocation solution into a convex combination of integer solutions, which will be used by our randomized auction mechanism.

We apply a LP duality based decomposition technique [13]. 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 $\sum_{l\in\boldsymbol{L}} \beta_l y^{(t)l}_{n,k} = y^{(t)\boldsymbol{F}}_{n,k}$, $\forall n \in [N], k \in [K]$, and $\sum_{l\in\boldsymbol{L}} \beta_l = 1$. The randomized auction in each round can choose the l^{th} 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, $\sum_{l\in\boldsymbol{L}} \beta_l y^{(t)l}_{n,k}$, that equals the fraction solution $y^{(t)\boldsymbol{F}}_{n,k}$, because otherwise, the expected social welfare achieved by these integer solutions equals that achieved by the fractional solution, which is apparently a contradiction to the fact that the fractional solution achieves a higher social welfare than any possible integer solution. Therefore, to achieve a feasible decomposition, we need to scale down the optimal fractional solution by a certain factor. According to [13], 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) , then we can use λ as the scaling factor, and rest assured that a feasible solution to the following decomposition problem exists:

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

s.t.

$$\sum_{l\in\boldsymbol{L}} \beta_l y^{(t)l}_{n,k} = y^{(t)F}_{n,k}/\lambda, \qquad \forall n \in [N], k \in [K], \tag{6a}$$

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

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

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

5.2.1 A primal-dual algorithm for one-round resource allocation

Alg. 2 is our primal-dual approximation algorithm to the NP-hard allocation problem (3). $C^{(t)}_{q,r} = \max_{n,k}\{c^{(t)}_{n,k,r,q}\}$ is the maximum amount of type-r resource at datacenter q required by any bundle in t. $C^{(t)}_{min} = \min_{r\in[R],q\in[Q]}\{A^{(t)}_{q,r}/C^{(t)}_{q,r}\}$ is the minimum ratio between the total amount of available resource of a type in datacenter and the amount of the resource in the datacenter required by one bundle. In practice, the resource pool is substantially larger than a single user's demand, and hence $C^{(t)}_{min} \gg 1$. The main idea of the algorithm is to introduce an auxiliary variable $z^{(t)}_{q,r}$ for each type of resources (which is the dual variable associated with constraint (3(b)), acting as the unit price in allocation decision. The unit price $z^{(t)}_{q,r}$ 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 users with a higher bid on a lower valued bundle as the winner.

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

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 with 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.

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

```
1: $\mathcal{N} \leftarrow \emptyset, z_{base} \leftarrow QR \cdot exp((C^{(t)}_{min} - 1))$
2: $y^{(t)}_{n,k} \leftarrow 0, s^{(t)}_n \leftarrow 0, z^{(t)}_{q,r} \leftarrow 1/A^{(t)}_{q,r}, \forall n \in [N], k \in [K], r \in [R], q \in [Q]$
3: while $\sum_{r\in[R]}\sum_{q\in[Q]} A^{(t)}_{q,r} z^{(t)}_{q,r} < z_{base}$ AND $|\mathcal{N}| \neq N$ do
4:     for all $n \notin \mathcal{N}$ do
5:         $k(n) = \arg\max_{k\in[K]}\{w^{(t)}_{n,k}\}$
6:     end for
7:     $n^* = \arg\max_{n\in[N]}\{\frac{w^{(t)}_{n,k(n)}}{\sum_{r\in[R]}\sum_{q\in[Q]} c^{(t)}_{n,k(n),r,q} z^{(t)}_{q,r}}\}$
8:     $y^{(t)}_{n^*,k(n^*)} \leftarrow 1, s^{(t)}_{n^*} \leftarrow w^{(t)}_{n^*,k(n^*)}, \mathcal{N} \leftarrow \mathcal{N} \cup \{n^*\}$
9:     for all $r \in [R], q \in [Q]$ do
           $z^{(t)}_{q,r} \leftarrow z^{(t)}_{q,r} \cdot z_{base}^{c^{(t)}_{n^*,k(n^*),q,r}/(A^{(t)}_{q,r} - C^{(t)}_{q,r})}$
10:    end for
11: end while
```

5.2.2 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. But 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 $v^{(t)}_{n,k}$ and τ associate with primal constraints (6a) and (6b), respectively:

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

s.t.

$$\sum_{n\in[N]}\sum_{k\in[K]} y^{(t)l}_{n,k} v^{(t)}_{n,k} + \tau \leq 1, \qquad \forall l \in \boldsymbol{L}, \tag{7a}$$

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

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. Alg. 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 [13]. 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 B.

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.*

5.3 The Randomized Auction

Alg. 3 gives our randomized auction to be carried out in each round of the online algorithm in Alg. 1. It selects an integer bundle allocation solution $\mathbf{y}^{(t)l}$ produced by the decomposition method with probability β_l (Line 8). The payment from a winning bidder n should satisfy 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 $\Pi_n^{(t)l}$ 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. We obtain the payment rule in Line 9, which satisfies the two conditions.

The following theorem provides the properties achieved by the randomized auction.

THEOREM 4. *$\mathcal{A}_{round}$ 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 $\mathcal{A}_{round}$ into the online algorithm framework $\mathcal{A}_{online}$ in Alg. 1. The competitive ratio of the online auction can be derived readily from Thm. 1 using $\nu = \lambda(1 + B_{max})$, the competitive ratio of the one-round randomized auction given in Thm. 4.

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

The complete proof of Thm. 5 can be found in Appendix C. We note that when $B_{max} \to 0$, the competitive ratio tends to $\lambda + \frac{1}{e-1}$. Following the discussions on Thm. 3 in Sec. 5.2.1, when λ tends to e, the competitive ratio of the online auction tends to $e + \frac{1}{e-1} \simeq 3.30$.

5.4 Improving the scale-down factor

We have decomposed the fractional allocation solution in Sec. 5.2 after scaling it down by the approximation ratio λ of the one-round allocation Alg. 2, such that a feasible solution to the decomposition problem (6) is guaranteed [13]. According to Thm. 5, λ (as the scale-down factor in the decomposition method) is closely related to the competitive ratio of our online auction, such that a smaller scale-down factor may potentially lead 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-based algorithm in Alg. 4 to compute the smallest scale-down factor that enables feasible decomposition.

Algorithm 3 One-Round Randomized Auction $\mathcal{A}_{round}$ 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 $y_{n,k}^{(t)F}, \forall n \in [N], k \in [K]$.
2: for all $n \in [N]$ do
3:     $\forall n' \in [N], k \in [K], w_{n',k}^{'(t)} = max\{0, (1 - x_n^{(t-1)})b_{n,k}^{(t)}\}$, if $n' \neq n$. Otherwise $w_{n',k}^{'(t)} = 0$.
4:     Solve LP relaxation of (3), with $w_{n',k}^{'(t)}$'s. Denote the optimal objective function value by $\widetilde{V}_{-n}^{(t)}$.
5:     $\Pi_n^{(t)F} = \widetilde{V}_{-n}^{(t)} - \sum_{n'\neq n}\sum_k y_{n',k}^{(t)F} w_{n',k}^{(t)}$
6: end for
7: Solve the pair of primal-dual decomposition LPs in (6) and (7) using the ellipsoid method, using Alg. 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, $\beta_l$, $\forall l \in \boldsymbol{L}$.
8: Choose $\mathbf{y}^{(t)l}$ with probability $\beta_l$, $\forall l \in \boldsymbol{L}$
9: $\forall n \in [N], \Pi_n^{(t)l} = \Pi_n^{(t)F} \cdot \frac{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)l}}{\sum_{k\in[K]} w_{n,k}^{(t)} y_{n,k}^{(t)F}}$
```

The algorithm is designed based on a property of the scale-down factor, as given in Thm. 6 . 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 in Thm. 5 in the worst case. We will investigate the effectiveness of the improved scale-down factor in our trace-driven simulations.

Algorithm 4 Binary searching smallest scale-down factor

```
Require: allowable error $\delta$
1: Replace Line 7 of Alg. 3 with the following steps:
2: $\lambda_l \leftarrow 1$, $\lambda_r \leftarrow \lambda + \delta$
3: while $\lambda_r - \lambda_l > \delta$ do
4:     $\lambda_m \leftarrow (\lambda_l + \lambda_r)/2$
5:     Solve (7) with scale-down factor $\lambda_m$.
6:     If Decomposing success then $\lambda_r \leftarrow \lambda_m$ Else $\lambda_l \leftarrow \lambda_m$
7: end while
8: Solve (7) with scale-down factor $\lambda_r$.
```

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 $\lambda_2 > \lambda_1$.*

6. PERFORMANCE EVALUATION

We evaluate our online auction design using trace-driven simulations. We investigate 6 types of VMs distributed in Q (default 3) datacenters, assembled from three types of resources (CPU, RAM, Disk capacity, $R = 3$), following the configurations in Table 1. Users' resource demands are extracted from Google cluster-usage data [3], 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 calculate the numbers and types of VMs in Table 1 that altogether can make up the resource demands in the job request, and compose a bidding bundle based on the numbers and types of VMs; 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 1 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 compare the performance of three algorithms:

▷ *Alloc*, a pure online resource allocation algorithm, with the one-round resource allocation algorithm Alg. 2 serving in the place of A_{round} in $\mathcal{A}_{online}$ in Alg. 1.

▷ *Auc*, our online auction algorithm presented in Sec. 5.3, *i.e.*, $\mathcal{A}_{online}$ combined with $\mathcal{A}_{round}$ in Alg. 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 $\mathcal{A}_{round}$ in $\mathcal{A}_{online}$.

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.

6.1 Different numbers of cloud users

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 Figure 1. The offline/online ratio of *Auc* declines when N is large ($N > 2000$), which is consistent with our theoretical analysis in Thm. 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 Alg. 4 in decomposing the fractional solution into better integer solutions in practical scenarios.

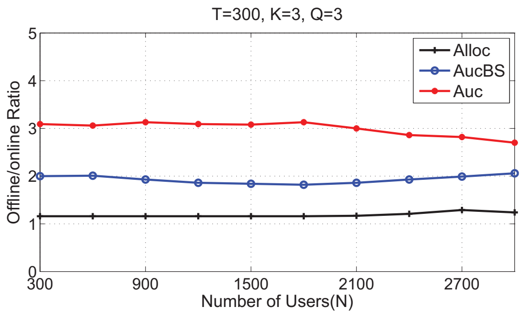

Figure 1: Offline/online ratio under different numbers of users N

6.2 Different numbers of rounds

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 one hour. A number of rounds in the range of $[300, 3000]$ corresponds to 12.5 days to about 4 months, which represents a reasonable period of time for a user to set a total amount of budget to use in. We observe in Figure 2 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.

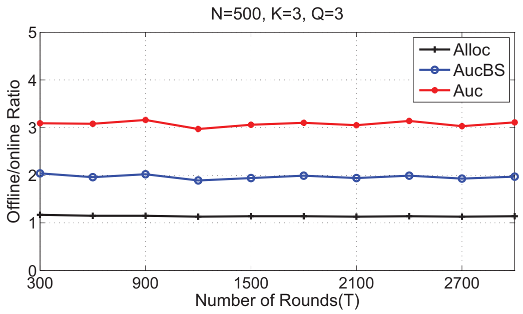

Figure 2: Offline/online ratio under different total numbers of rounds T

6.3 Different numbers of bundles and datacenters

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. 3 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 Thm. 5) is related to λ, the approximation ratio of the one-round resource allocation algorithm in Alg. 2, which is further closely related to $\epsilon^{(t)}$. When K is smaller, $\epsilon^{(t)}$ is potentially smaller (recall $\epsilon^{(t)}$ is the maximum ratio between the overall demands for any type of resource in any two bundles among the K bundles a user bids for in t), 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 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. 3.

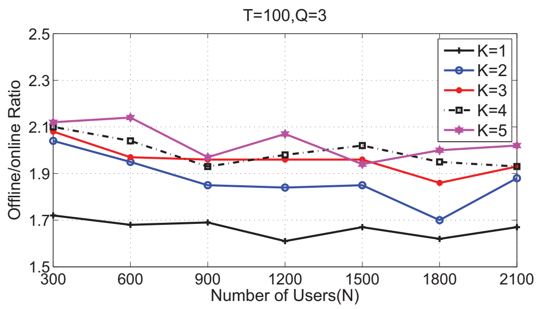

Figure 3: Offline/online ratio of *AucBS* with different numbers of bundles K.

Fig. 4 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 Alg. 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 2.70 when Q is 10.

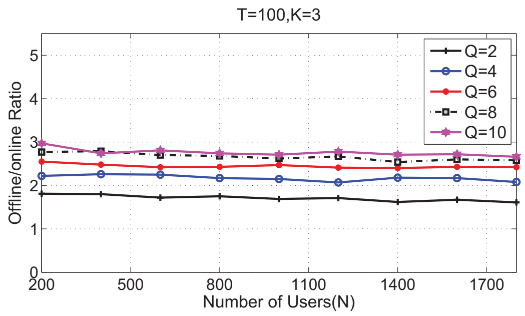

Figure 4: Offline/online ratio of *AucBS* with different numbers of datacenters Q.

We note that the performance ratios obtained in our simulations are much smaller than the theoretical ratios computed based on Thm. 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.

6.4 User satisfaction

Finally we study the performance of the improved online auction algorithm *AucBS* in terms of user satisfaction, evaluated as the percentage of users who win a bundle in a round, averaged over the number of rounds the online auction runs. We set the total number of rounds to $T = 100$, vary the number of users N, and derive the user satisfaction under different values of K and Q.

Fig. 5 shows that the fewer datacenters, the better user satisfaction results. This can be explained as follows: Once a specific type of resource is used up at one datacenter, bundles requesting resources containing this type of resource in this datacenter cannot obtained, even though other resource is available in other datacenters; a larger number of datacenters leads to more dispersed distribution of resources, resulting in high probability for the above scenario to happen.

Fig. 6 shows that a smaller K leads to better user satisfaction. When K is larger, λ is larger, and hence the performance of the online auction degrads. On the other hand, when a user bids for more bundles, the cloud provider has a larger decision space for resource allocation, which should potentially lead to a higher chance for a user to be allocated one of the requested bundles. Therefore, the results in Fig. 6 reveal that the impact of λ dominates that of the latter.

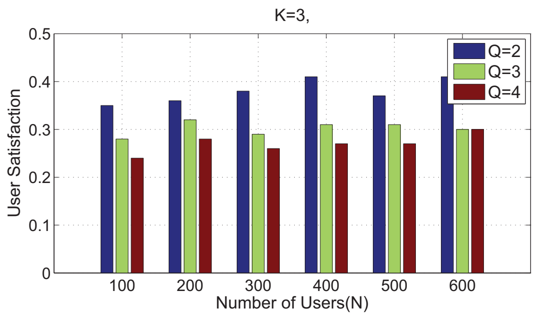

Figure 5: User satisfaction of *AucBS* with different numbers of datacenters Q.

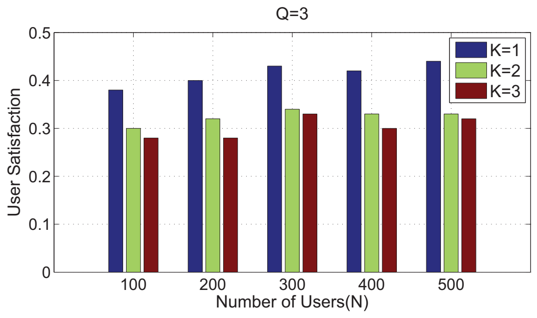

Figure 6: User satisfaction of *AucBS* with different numbers of bundles K.

7. CONCLUDING REMARKS

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 of 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 sub-framework 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. Our overall online VM auction guarantees a theoretical competitive ratio close to 3.30 in typical scenarios, and its design may shed light to similar auction problems in related settings.

8. ACKNOWLEDGMENT

The research was supported in part by a grant from Hong Kong RGC under the contract HKU 718513.

9. REFERENCES

[1] Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/.

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

[3] *Google Cluster Data*, https://code.google.com/p/googleclusterdata/.

[4] Windows Azure: Microsoft's Cloud Platform. http://www.windowsazure.com/.

[5] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 242–253. ACM, 2011.

[6] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer. Minimum congestion mapping in a cloud. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 267–276. ACM, 2011.

[7] N. Buchbinder, K. Jain, and J. S. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms*, 2007.

[8] N. Cherfi and M. Hifi. A column generation method for the multiple-choice multi-dimensional knapsack problem. *Computational Optimization and Applications*, 46(1):51–73, 2010.

[9] H. Fu, Z. Li, and C. Wu. Core-selecting auction design for dynamically allocating heterogeneous vms in cloud computing. Submitted to INFOCOM 2014.

[10] G. Gens and E. Levner. Complexity of approximation algorithms for combinatorial problems: a survey. *ACM SIGACT News*, 12(3):52–65, 1980.

[11] P. Godfrey, M. Schapira, A. Zohar, and S. Shenker. Incentive compatibility and dynamics of congestion control. In *ACM SIGMETRICS Performance Evaluation Review*, pages 95–106, 2010.

[12] R. Lavi and N. Nisan. Competitive analysis of incentive compatible on-line auctions. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 233–241. ACM, 2000.

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

[14] D. Lehmann, L. I. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM (JACM)*, 49, 2002.

[15] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau. A game theoretic approach to provide incentive and service differentiation in p2p networks. In *ACM SIGMETRICS Performance Evaluation Review*, pages 189–198, 2004.

[16] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[17] A. Mu'Alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. *Games and Economic Behavior*, 64(2):612–631, 2008.

[18] N. Nisan. *Algorithmic game theory*. Cambridge University Press, 2007.

[19] D. Niu, C. Feng, and B. Li. Pricing cloud bandwidth reservations under demand uncertainty. In *ACM SIGMETRICS Performance Evaluation Review*, pages 151–162, 2012.

[20] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinational auctions. *Management science*, 44(8):1131–1147, 1998.

[21] G. Shanmuganathan, A. Gulati, and P. Varman. Defragmenting the cloud using demand-based resource allocation. In *ACM SIGMETRICS Performance Evaluation Review*, pages 67–80, 2013.

[22] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. M. Lau. An online auction framework for dynamic resource provisioning in cloud computing. Technical report, 2014. `http://i.cs.hku.hk/~cwu/papers/sigmetrics14-techreport.pdf`.

[23] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[24] Q. Wang, K. Ren, and X. Meng. When cloud meets ebay: Towards effective pricing for cloud computing. In *Proc. of IEEE INFOCOM*, pages 936–944, 2012.

[25] W. Wang, B. Liang, and B. Li. Revenue maximization with dynamic auctions in iaas cloud markets. In *Proc. IEEE ICDCS*, 2013.

[26] C. Wu, Z. Li, X. Qiu, and F. C. M. Lau. Auction-based p2p vod streaming: Incentives and optimal scheduling. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2012.

[27] S. Zaman and D. Grosu. Combinatorial auction-based mechanisms for vm provisioning and allocation in clouds. In *IEEE/ACM CCGrid*, pages 729–734, 2012.

[28] H. Zhang, B. Li, H. Jiang, F. Liu, A. V. Vasilakos, and J. Liu. A framework for truthful online auctions in cloud computing with heterogeneous user demands. In *Proc. of IEEE INFOCOM*, 2013.

[29] L. Zhang, Z. Li, and C. Wu. Dynamic resource provisioning in cloud computing: A randomized auction approach. In *Proc. of IEEE INFOCOM*, 2014.

[30] Y. Zhu, B. Li, and Z. Li. Truthful spectrum auction design for secondary networks. In *Proc. of IEEE INFOCOM*, 2012.

APPENDIX

A. PROOF OF THEOREM 1

Proof. We prove the correctness and the competitiveness of $\mathcal{A}_{online}$ 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 t-th iteration, $\Delta P^{(t)} = P^{(t)} - P^{(t-1)}$, the same for $\Delta D^{(t)}$ in dual (2). Then $\mathcal{A}_{online}$ satisfies $\Delta D^{(t)} \le \left(\nu + \frac{1}{\gamma-1}\right) \Delta P^{(t)}$, at any round $\forall t \in [T]$.
3. The algorithm produces an almost feasible solution for primal (1). Specifically, its outputs $y^{(t)}_{n,k}$ satisfy (1a), (1c) and (1d). For constraint (1b), we achieve a slightly weaker property : For all user $n \in [N]$,

$$\sum_{t\in[T]} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k} \le B_n(1 + B_{max}) \qquad (8)$$

Proof of (1): Since $w^{(t)}_{n,k} \ge b^{(t)}_{n,k}(1 - x^{(t-1)}_n)$ no matter whether $x^{(t-1)}_n < 1$ or $x^{(t-1)}_n \ge 1$, constraint (4a) guarantees

$$s^{(t)}_n + \sum_{r\in[R]} \sum_{q\in[Q]} c^{(t)}_{n,k,r} z^{(t)}_{q,r} \ge b^{(t)}_{n,k}(1 - x^{(t-1)}_n)$$

Also notice $x^{(t)}_n$ is non-decreasing with t, so (2a) holds.

Proof of (2): At time t, $\Delta P^{(t)} = \sum_{n\in\mathcal{N}} b^{(t)}_{n,k_n}$.

$$\begin{aligned}\Delta D^{(t)} &= \sum_{n\in\mathcal{N}} B_n(x^{(t)}_n - x^{(t-1)}_n) + d \\ &= \sum_{n\in\mathcal{N}} \left(b^{(t)}_{n,k_n} x^{(t-1)}_n + \frac{b^{(t)}_{n,k_n}}{\gamma - 1} \right) + d \\ &\le \sum_{n\in\mathcal{N}} \left(b^{(t)}_{n,k_n} x^{(t-1)}_n + \frac{b^{(t)}_{n,k_n}}{\gamma - 1} \right) + \nu p\end{aligned}$$

Substitute $\sum_{n\in\mathcal{N}} b^{(t)}_{n,k_n}(1 - x^{(t-1)}_n)$ for p, we have:

$$\begin{aligned}\Delta D^{(t)} &\le \sum_{n\in\mathcal{N}} \left(\nu + \frac{1}{\gamma-1} - (\nu - 1) x^{(t-1)}_n \right) b^{(t)}_{n,k_n} \\ &\le \left(\nu + \frac{1}{\gamma - 1} \right) \Delta P^{(t)}\end{aligned}$$

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 inequality: $\forall n \in [N], t' \in [T]$,

$$x^{(t')}_n \ge \frac{1}{\gamma - 1} \left(\gamma^{\frac{\sum_{t=1}^{t'} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k}}{B_n}} - 1 \right), 0 \le t' \le T \qquad (9)$$

We prove (9) by induction. (9) 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 $n \in \mathcal{N}$:

$$\begin{aligned}x^{(t)}_n &= x^{(t-1)}_n \left(1 + \frac{b^{(t)}_{n,k_n}}{B_n} \right) + \frac{b^{(t)}_{n,k_n}}{B_n(\gamma - 1)} \\ &= \frac{1}{\gamma - 1} \left(\gamma^{\frac{\sum_{t=1}^{t'-1} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k}}{B_n}} \left(1 + \frac{b^{(t)}_{n,k_n}}{B_n} \right) - 1 \right)\end{aligned}$$

Comparing this with our target (9), obviously we only need to show: $1 + \frac{b^{(t)}_{n,k_n}}{B_n} \ge \gamma^{\frac{b^{(t)}_{n,k_n}}{B_n}}$. We utilize the inequality:

$$\frac{\ln(1+x)}{x} \ge \frac{\ln(1+y)}{y}, \forall 0 \le x \le y \le 1$$

Since $\frac{b^{(t)}_{n,k_n}}{B_n} \le B_{max}$, we have:

$$\ln(1 + \frac{b^{(t)}_{n,k_n}}{B_n}) \ge \frac{b^{(t)}_{n,k_n}}{B_n} \cdot \frac{\ln(1 + B_{max})}{B_{max}} = \frac{b^{(t)}_{n,k_n}}{B_n} \ln \gamma$$

Thus, $1 + \frac{b^{(t)}_{n,k_n}}{B_n} \ge \gamma^{\frac{b^{(t)}_{n,k_n}}{B_n}}$, and we prove (9).

Now we utilize the inequality (9) to prove (8). For some user n, suppose t' is the first time $\sum_{t=1}^{t'} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k} \ge B_n$. Then by (9), $x^{(t')}_n \ge 1$. The algorithm never gives user n any new bundles once $x_n \ge 1$, since the weight $w^{(t)}_{n,k}$ in $\mathcal{A}_{round}$ will be set to 0. So $\sum_{t=1}^{T} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k} = \sum_{t=1}^{t'} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k}$. We know t' is the first time user n's total winning bids exceeding its budget B_n. So

$$\sum_{t=1}^{t'-1} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k} < B_n, \text{ then:}$$

$$\begin{aligned}\sum_{t=1}^{T} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k} &= \sum_{t=1}^{t'-1} \sum_{k\in[K]} b^{(t)}_{n,k} y^{(t)}_{n,k} + \sum_{k\in[K]} b^{(t')}_{n,k} y^{(t')}_{n,k} \\ &\le B_n + \max_{k\in[K]} b^{(t')}_{n,k} \\ &\le B_n(1 + B_{max})\end{aligned}$$

Finally we put the above 3 claims together and calculate the actual total social welfare. Since the increment of valuation is the minimum between user's valuation and his remaining budget, total social welfare should be:

$$\begin{aligned}&\sum_{n\in[N]} \min\{B_n, \sum_{k\in[K]} \sum_{t\in[T]} b^{(n)}_{n,k} y^{(n)}_{n,k}\} \\ &\ge \sum_{n\in[N]} \sum_{k\in[K]} \sum_{t\in[T]} b^{(n)}_{n,k} y^{(n)}_{n,k} / (1 + B_{max}) \\ &= P^{(T)} / (1 + B_{max})\end{aligned}$$

By claim 2: $\Delta D^{(t)} \le \left(\nu + \frac{1}{\gamma-1}\right) \Delta P^{(t)}$, and recall $P^{(0)} = D^{(0)} = 0$, we have $D^{(T)} \le \left(\nu + \frac{1}{\gamma-1}\right) P^{(T)}$. Thus the social welfare is at least $D^{(T)}/[(1+B_{max})(\nu + \frac{1}{\gamma-1})]$. By duality, the approximation ratio of $\mathcal{A}_{online}$ is $(1 + B_{max})(\nu + \frac{1}{\gamma-1})$. $\square$

B. THE SEPARATION ORACLE AND PROOF OF LEMMA 1

First we describe the construction of the separation oracle. In each iteration of the ellipsoid method, a possible solution of (7) $\{v^{(t)}_{n,k}\}, \tau$ is generated, and is given as the input of the severation oracle. The separation oracle we present in Alg. 5 judges whether this solution is feasible, *i.e.*, not conflict with any constraints in (7a). And if it is not feasible, the separation oracle should return a conflict constraints as the separation plane, *i.e.*, a set of $y_{n,k}^{(t)l}$. So if the input solution has objective value smaller than 1, we use

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

as the separation plane. If the objective value equals to 1, then we find a feasible solution (later we will prove why in this case, it is a feasible solution). If the objective value is larger than 1, we need to find a set of $y_{n,k}^{(t)l}$, 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 :

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

So we find a conflict constraint if we can find $y_{n,k}^{(t)l}$ satisfies

$$\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} \geq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)}$$

Observe the left side of the inequality. It has exactly the same form with the objective function of (3). Furthermore, the value of $y_{n,k}^{(t)l}$ also should satisfy all the constraints in (3). So we can $\mathcal{A}_{round}$ to get such $y_{n,k}^{(t)l}$. Although there is a little difference here: $v_{n,k}^{(t)}$ can be negative, while $w_{n,k}^{(t)}$ in (3) can only be nonnegative value. In order to handle this problem, we introduce new variables $\widetilde{v}$ and $\widetilde{y}$ in the separation oracle, which is detailed in the proof.

Algorithm 5 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 $\mathcal{A}_{round}$ 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

$$\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)l} v_{n,k}^{(t)} \geq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)} \tag{10}$$

Note the $v_{n,k}^{(t)}$ here can be negative value, so we cannot invoke $\mathcal{A}_{round}$ directly. That is the reason we define $\widetilde{v}_{n,k}^{(t)}$. Then we use $\mathcal{A}_{round}$, with input vector $\widetilde{v}_{n,k}^{(t)}$ to get $\widetilde{y}_{n,k}^{(t)l}$ satisfying:

$$\begin{aligned}\sum_{n\in[N]}\sum_{k\in[K]} \widetilde{y}_{n,k}^{(t)l} \widetilde{v}_{n,k}^{(t)} &\geq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} \widetilde{v}_{n,k}^{(t)}\\ &\geq \frac{1}{\lambda}\sum_{n\in[N]}\sum_{k\in[K]} y_{n,k}^{(t)F} v_{n,k}^{(t)}\end{aligned}$$

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 such a 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. Its correctness under two cases: objective value equals 1 and smaller than 1 is obvious. When it is larger than 1, the oracle generates $\widetilde{v}_{n,k}^{(t)}$ and calls $\mathcal{A}_{round}$. The correctness of this method has been discussed. So this oracle solves (7) as we expect.

C. PROOF OF THEOREM 5

Proof. The only difference between Thm. 5 and Thm. 1 is we introduce randomness here. Recall the proof of Thm. 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,

$$E[\Delta P^{(t)}] = \sum_{n\in[N]}\sum_{k\in[K]} E[y_{n,k}^{(t)} b_{n,k}^{(t)}]$$

$$\begin{aligned}E[\Delta D^{(t)}] =& \sum_{n\in[N]} B_n E[x_n^{(t)} - x_n^{(t-1)}] + E[d]\\ &\leq \lambda(1+B_{max})E[p] + \sum_{n\in[N]} E[\sum_{k\in[K]} y_{n,k}^{(t)} b_{n,k}^{(t)}]\cdot\\ &\quad E[x_n^{(t-1)} + \frac{1}{\gamma-1}]\\ &\leq E[\Delta P^{(t)}]\Big(\lambda(1+B_{max})\sum_{n\in[N]} E[1-x_n^{(t-1)}]+\\ &\quad \frac{1}{\gamma-1} + \sum_{n\in[N]} E[x_n^{(t-1)}]\Big)\\ &\leq (\lambda(1+B_{max}) + \frac{1}{\gamma-1})E[\Delta P^{(t)}]\end{aligned}$$

So the final actual competitive ratio is $(1+B_{max})(\lambda(1+B_{max}) + \frac{1}{\gamma-1})$. □