An Efficient Cloud Market Mechanism for Computing Jobs with Soft Deadlines

Ruiting Zhou, Zongpeng Li, Chuan Wu, Zhiyi Huang

***Abstract*—This work studies the cloud market for computing jobs with completion deadlines, and designs efficient online auctions for cloud resource provisioning. A cloud user bids for future cloud resources to execute its job. Each bid includes (a) a utility, reflecting the amount that the user is willing to pay for executing its job, and (b) a soft deadline, specifying the preferred finish time of the job, as well as a penalty function that characterizes the cost of violating the deadline. We target cloud job auctions that executes in an online fashion, runs in polynomial time, provides truthfulness guarantee, and achieves optimal social welfare for the cloud ecosystem. Towards these goals, we leverage the following classic and new auction design techniques. First, we adapt the posted pricing auction framework for eliciting truthful online bids. Second, we address the challenge posed by soft deadline constraints through a new technique of compact exponential-size LPs coupled with dual separation oracles. Third, we develop efficient social welfare approximation algorithms using the classic primal-dual framework based on both LP duals and Fenchel duals. Empirical studies driven by real-world traces verify the efficacy of our online auction design.**

***Index Terms*—Cloud computing, Auction Mechanism, Online Algorithm.**

I. INTRODUCTION

Cloud computing has emerged as a new computing paradigm that offers users rapid on-demand access to resources such as CPU, RAM and disk storage, with minimal management overhead. In the past decade, two types of cloud platforms blossomed on the Internet, including (i) large-scale Internet data centers, exemplified by Amazon EC2 [1], Microsoft Azure and Linode [2], [3], which organize a shared resource pool for serving their users; and (ii) co-location data centers, often found in metropolitan areas, where smaller clouds from different users are physically co-located, jointly managed and serviced by the co-location [4].

Virtualization technologies help cloud providers pack their resources into different types of virtual machine (VM), for allocation to cloud users. For example, Amazon EC2 [1] currently offers 23 types of different VM types in 7 categories. Each type of VM has its focus and forte, and a large computing job often requires cooperation among multiple VM instances. For example, social games [5] and enterprise applications [6] are often composed of a front-end web server tier, a load balancing tier and a back-end data storage tier, each suited for execution on a VM that is abundant in a particular type of resource: bandwidth, CPU, or storage.

Cloud computing jobs can be categorized into two types, depending on whether their computing need is elastic or not. Cloud jobs such as large scale web servers utilize cloud service as a utility, and require the rented VMs to be always active, with possible dynamic size scaling. These jobs are similar to the *power users* in a power grid who demands always-on power supply. Other jobs such as big data analytics and Google crawling data processing often have a batch processing nature. They require a certain computing job to be completed without demanding always-on VM service, and may tolerate a certain level of delay in the job completion. These users are similar to the *energy users* in a power grid who needs to draw a fixed quantity of energy for powering a given job, but in a flexible time window.

Existing market mechanisms for cloud computing, particularly the auction type mechanisms, have been implicitly targeting the first type of non-elastic cloud jobs. In such one-round [7] and online [8] cloud resource auctions, once a bid is accepted, the service time window of the corresponding VMs is fixed, *i.e.* either in the current round [8] or between the start and finish times prescribed in the bid [9]. Such auction algorithms do not need to consider the scheduling of accepted jobs. In sharp contrast, a well designed market mechanism for the second type of elastic jobs must pay close attention to not only whether to accept a bid, but when to schedule its execution based on its deadline information. For example, consider a cloud user who bids for a VM bundle tailored for human genome analysis. Its job can be processed within 3 hours if the specified VM bundle is provisioned; however, as long as the computing result is available within the next 24 hours, the user is happy. This leaves ample space for job scheduling in the temporal domain, which a well-designed auction algorithm should judiciously exploit to maximize resource utilization and social efficiency — for example, scheduling a job within its tolerance window to time slots with relatively low demand.

This work generalizes existing auction design in the cloud market by proposing online auctions that explicitly handle jobs with prescribed deadlines. We further allow a cloud user to express soft deadlines, described by both a preferred job completion time, coupled with a penalty function that encodes how much penalty is associated with different degrees of dead-

R. Zhou is with the University of Calgary, Calgary, AB, Canada (e-mail: rzho@ucalgary.ca).

Z. Li is with the University of Calgary, Calgary, AB, Canada, and also with the School of Computer and Collaborative Innovation Center of Geospatial Technology, Wuhan University, Wuhan, China (e-mail: zongpeng@ucalgary.ca).

C. Wu is with the University of Hong Kong, Kowloon, Hong Kong (e-mail: cwu@cs.hku.hk).

Z. Huang is with the University of Hong Kong, Kowloon, Hong Kong (e-mail: zhiyi@cs.hku.hk).

This work was supported in part by NSERC, Wedge Networks, and HK-RGC (grant #: HKU 17202115E, HKU 718513, HKU 17204715, HKU 17225516, C7036-15G (CRF)).

line violation. Compared with simple market mechanisms such as fixed pricing, a well-designed auction provides automatic price discovery, promptly adapts prices with the fluctuation of supply and demand, and allocates cloud resources to jobs who value them the most, maximizing the overall "happiness" of everyone in the cloud ecosystem.

We simultaneously target the following goals in our cloud auction design. First, we require the cloud auction to be computationally efficient and executes in polynomial time. Second, the auction should be truthful, so that bidding true job valuation is the dominant strategy for a cloud user. Third, the auction should maximize the social welfare of everyone in the system including both the cloud provider and the cloud users. Such cloud auction design is faced with a number of challenges. First, truthfulness is a rather strong property that comes only with a pair of carefully prepared VM allocation and payment algorithms that work in concert with each other. Furthermore, even if the cloud users can be assumed to be altruistic and truthful bids are given for free, the winner determination problem for social welfare maximization is an integer linear program (ILP) that is NP-hard to solve. A new challenge unique to this work is the non-traditional type of soft deadline constraints, which is hard to model and handle with traditional LP formulation and algorithm design. Last but not least, we require the auction to be online, immediately making a decision upon the arrival of each bid, without knowing future bids in the market, yet still guaranteeing near-optimal decision making as compared to the offline optimum.

We first consider a basic setting where resources in the cloud are free of cost up to a known capacity limit, and that the soft deadline can be expressed by enumerating a few hard deadline options and their corresponding bidding prices. We first present a natural ILP formulation of the social welfare maximization problem. While polynomial in size, this ILP involves both conventional constraints (capacity limits) and unconventional constraints (job deadlines). The latter further lead to unconventional dual variables that are hard to interpret and update in a primal-dual algorithm framework we will leverage. We convert the natural ILP into a *compact-exponential* ILP that has a compact formulation of conventional constraints only, at the price of involving an exponential number of variables.

We apply the *posted pricing* primal-dual framework to the compact-exponential ILP for online social welfare maximization. Although the dual has an exponential number of constraints, we show fast dual oracles that can quickly update the dual variables, which are interpreted as unit cost of cloud resources in different time slots. We maintain carefully estimated resource costs based on recently designed exponential cost functions [10]. Upon receiving a bid, we compare the bidding price with the estimated cost of the bid. If the bidding price is higher, the bid is accepted and dual variables are updated; otherwise the bid is rejected. The posted pricing framework charges winning jobs an estimated cost that is independent from the bidding price, and is truthful [11]. We conduct theoretical analysis on the competitive ratio and prove its upper-bound.

We proceed to generalize our cloud auction design by addressing two practical concerns. First, we model the cost of resource provisioning in data centers, using a convex cost function that characterizes server cost with Dynamic Voltage Frequency Scaling [12]. Second, we consider the general form of a soft deadline, specified by (i) a preferred deadline and (ii) a non-decreasing penalty function for deadline violation. The new social welfare maximization problem is an integer convex program. We resort to a new primal-dual solution framework for well-structured convex programs based on Fenchel dual [13], and adapt our posted pricing auction framework from the previous scenario to this general setting.

In the rest of the paper, we discuss related work in Sec. II, and introduce the system model in Sec. III. Design and analysis of the online cloud auctions are presented in Sec. IV and Sec. V. Sec. VI presents simulation studies, and Sec. VII concludes the paper.

II. Previous Research

Market mechanism design for cloud computing, particularly auction mechanisms for cloud resource trading, has attracted substantial interest from the research community, with a large number of VM auctions spawned in the past few years [7]–[9], [14]–[16].

The earliest VM auctions are simple in that they are one-round auctions, and assume that the cloud provisions a single type of VM, or that VM configurations are equivalent up to linear scaling [14]. They also assume the scenario of static VM provisioning, where the number and type of VMs to be sold are predetermined prior to the auction start [15].

Dynamic VM provisioning, in which the cloud provider makes decision on which VMs to assemble and how many based on demand learned from user bid during the auction, has been studied in the past two years [7]–[9]. Zhang *et al.* design a randomized auction for dynamic resource provisioning in cloud computing based on a convex decomposition technique, which is truthful and guarantees a small approximation ratio in social welfare [7]. Shi *et al.* further study dynamic resource provisioning where cloud users are subject to budget constraints, and design online auctions where decision making is coupled in the time domain due to fixed user budgets [8].

Online cloud auctions appear later than their one-round counterparts. Zhang *et al.* is among the first to study online cloud auction design, but they assume all VMs are of a uniform type [16]. The work of Shi *et al.* [8] designs online auctions, but does not consider the temporal correlation in decision making due to jobs spanning multiple time slots. A recent work of Zhang *et al* [9] study online cloud auctions where a user bids into a fixed time window for job execution; hence the scheduling dimension is non-present in their solution space.

There have been recent studies on mechanism deign for batch jobs with deadlines. Lucier *et al.* study two scheduling algorithms for jobs with deadlines in cloud computing clusters [17]. They analyze the competitive ratio for non-committed scheduling, which does not require to finish executing a job that has started execution. They do not provide any performance guarantee on the competitive ratio for committed scheduling. Navendu *et al.* design a truthful allocation and pricing mechanism for computing jobs with deadlines, but

restrict attention to the offline setting [18]. Azar *et al.* construct an online mechanisms for preemptive scheduling with deadlines [19]. Their mechanism is truthful and achieves a constant competitive ratio. All of those work consider only one fixed deadline for each job, and fail to model the server's operation cost.

Compared with existing literature on cloud auctions, this work is the first to design cloud auctions that explicitly consider job elasticity and job execution deadlines, which are important for practical applications to batch processing jobs. Accordingly, we propose the compact-exponential optimization technique that can effectively handle the new job deadline constraints in social welfare maximization for the cloud.

The online primal dual method (see [20] for a detailed survey) is a power algorithmic technique that that has witnessed broad applications, such as solving the ski rental problem, maximizing revenue in ad-auctions, and solving the general packing problem. The original primal dual framework works on linear programs, and is not used to solve problems modelled by convex programs in our work. Rather recently, new techniques were introduced to help apply the primal dual framework for algorithm design to convex programs. Blum *et al.* study online combinatorial auctions with production costs using the online primal dual framework [21]. They presents algorithms for various cost functions. Huang *et al.* further investigate the same problem and propose mechanisms with improved competitive ratio [11]. All those work don't consider the scheduling of jobs, and they cannot handle VM departures and resource recycling.

III. System Model and Preliminaries

We consider a cloud data center hosting a pool of K types of resource, including CPU, RAM and disk storage that can be dynamically assembled into different types of VMs. Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$. There are a total c_k unit of type-k resource in this cloud. The cloud service provider acts as the auctioneer to lease VMs to cloud users through an auction. User's bid arrives randomly in a large time span $1, 2, \ldots, T$. Note that multiple bids can arrive simultaneously, and would be ordered randomly. There are I users participating in the auction, and each user requests multiple types of VM, and specifies in its bid: (i) r_i^k, the total amount of type-k resource, and (ii) w_i, the number of slots required to finish the job by the designated VMs. Job execution doesn't need to be continuous. A user i's job can be executed at any time slot as long as the total execution time meets w_i before the deadline. We consider two soft deadline models in this work: a basic model with alternative deadlines and a general model with penalty function and server operation cost.

A. Jobs with Alternative Deadlines

We first consider a basic scenario where each user submits J optional bids to express disjunctive deadline options. A bid from user i consists of a list of desired types of resource r_i^k, $\forall k$; the number of requested slots w_i, and deadlines for job completion d_{ij}, $\forall j$, each with a corresponding bidding price b_{ij}. We use B_i to denote the bidding language of user i's bids submitted at time t_i:

$$B_i = \{t_i, \{r_i^k\}_{k\in[K]}, w_i, \{d_{ij}, b_{ij}\}_{j\in[J]}\}.$$

We adopt the XOR bidding rule that assumes a user can win at most one bid among its J optional bids [7]. Upon the arrival of each bid, the cloud provider decides immediately whether to accept it, and if so, which deadline to choose and how to schedule the job. A binary x_{ij} equals 1 if user i's jth bid wins, and 0 otherwise. Let another binary variable $y_i(t)$ encode the scheduling of user i's job: $y_i(t) = 1$ if user i's job is scheduled to run at time t, and 0 otherwise. The cloud provider also calculates the payment p_i for each winner i.

Let v_{ij} be the true valuation of user i's jth bid, then the utility of that bid with is $u_{ij}(b_{ij}) = v_{ij} - p_i$ if $x_{ij} = 1$, and is 0 if $x_{ij} = 0$. In practice, user are assumed to be selfish with a natural goal to maximize their own utilities; they may lie about their true valuations in the hope of a higher utility. The cloud provider instead pursues highest social welfare possible to make everyone in the cloud system "happy". Thus, it is important for the cloud provider to elicit truthful bids.

Definition 1. (Truthful Auction): A cloud auction is *truthful* if the dominant strategy for each user is to report its true valuation, which always maximizes its utility: for all $b_{ij} \neq v_{ij}$, $u_{ij}(v_{ij}) \geq u_{ij}(b_{ij})$.

Definition 2. (Social Welfare): The social welfare in the cloud market with alternative deadlines is the aggregate user utility $\sum_{i\in[I]}\sum_{j\in[J]} v_{ij}x_{ij} - \sum_{i\in[I]} p_i$ plus the cloud provider's utility $\sum_{i\in[I]} p_i$. Payments cancel themselves, and the social welfare becomes $\sum_{i\in[I]}\sum_{j\in[J]} v_{ij}x_{ij}$.

B. Jobs with Penalty Function and Operation Cost

We further consider a more general model where each user submits a single preferred deadline d_i, with a penalty function $g_i(\tau_i)$ defined over deadline violation τ_i:

$$g_i(\tau_i) = \begin{cases} g_{c_i}(\tau_i), & \text{if } \tau \in [0, T - d_i] \\ +\infty, & \text{otherwise} \end{cases} \quad (1)$$

where $d_i + \tau_i$ is the job completion time; $b_i - g_i(\tau_i)$ is the bidding price, decreasing with job completion time; $g_{c_i}(\cdot)$ is a nondecreasing function and $g_{c_i}(0) = 0$. User i's bid with this model is: $B_i = \{t_i, \{r_i^k\}_{k\in[K]}, w_i, d_i, b_i, g_i(\tau_i)\}$.

Existing studies on cloud auction design often ignore the server operation cost of the cloud provider. It is natural to include server cost in the computation of social welfare, albeit the fact that it makes social welfare optimization substantially more challenging (from linear to non-linear integer programming). The operation cost in the cloud comprises mainly of power consumption for provisioning the virtual machines, increasing as the amount of resources used grows. Let $z_k(t)$ be the amount of type-k resource used at time t in the cloud, then the cost function of type-k resource is defined as:

$$f_k(z_k(t)) = \begin{cases} \beta_k z_k(t)^{1+\gamma_k}, & \text{if } z_k(t) \in [0, c_k] \\ +\infty, & \text{otherwise} \end{cases} \quad (2)$$

Parameter β_k is the coefficient determined by the power consumption of each type of resource. Recent measurement studies suggest that the power consumption of memory, disk are significantly lower than that of CPU [22]. $\gamma_k \geq 0$ modulates the shape of the cost function, following the the operational model of physical servers in the cloud. For

example, Dynamic Voltage Frequency Scaling (DVFS) is a technique widely adopted in virtualization platforms, adjusting the frequency or voltage of CPUs to save power consumption [12]. γ_k is roughly 2 if the voltage is proportional to the usage of CPU when DVFS is enabled, and equals 0 when DVFS in disabled [23]. The shape of RAM and disk cost function is different from that of CPU, with $\gamma_k \in [0.5, 1]$ [22].

Similar to the notations in Sec. III-A, let a binary x_i be an auction decision and p_i be the payment. $v_i - g_i(\tau_i)$ is the true valuation of user i's bid. The cloud provider's utility equals the aggregate user payment minus the operation cost, *i.e.,* $\sum_{i\in[I]} p_i - \sum_{k\in[K]} \sum_{t\in[T]} f_k(z_k(t))$. The definitions of user i's utility, truthful auction and social welfare are omitted here as similar ones can be found in Sec. III-A. Table I summaries notation for ease of reference.

TABLE I: Summary of Notations

I	# of users	$[X]$	integer set $\{1, \ldots, X\}$
T	# of time slots	J	# of bids per user
f	cost function	f^*	convex conjugate of f
g	penalty function	t_i	user i's arrival time
p_i	user i's payment	u_i	user i's utlity

r_i^k	demand of type-k resource by user i
w_i	# slots requested by user i
τ_i	# slots that passes the deadline for user i
$d_{ij}(d_i)$	deadline of user i's jth (user i's) bid
$b_{ij}(b_i)$	bidding price of user i's jth (user i's) bid
$v_{ij}(v_i)$	true valuation of user i's jth (user i's) bid
$x_{ij}(x_i)$	user i's jth (user i's) bid wins (1) or not (0)
$y_i(t)$	whether or not to allocate user i's job in slot t
c_k	capacity of type-k resource
$p_k(t)$	marginal price of type-k resource at time t
$z_k(t)$	amount of allocated type-k resource at time t
$U_k(L_k)$	maximum (minimum) value per unit of type-k resource per unit of time
θ_k	$\max\{2, (1+\gamma_k)^{\frac{1}{\gamma_k}}\}$
ρ_k	$\max\{\frac{\theta_k}{c_k}\gamma_k, \frac{\theta_k}{c_k(\theta_k-1)} \ln(\frac{U_k'}{\beta_k(1+\gamma_k)c_k^{\gamma_k}})\}$
$\alpha_1(\alpha_2)$	competitive ratio of $A_{online1}$ ($A_{online2}$)

IV. Online Auction Mechanism for Jobs with Alternative Deadlines

In this section, we focus on the scenario where each user's job has J alternative deadlines. Sec. IV-A presents the social welfare maximization problem and the framework to handle such deadline problems. We design an online auction in Sec. IV-B and conduct theoretical analysis in Sec. IV-C.

A. Social Welfare Maximization Problem

Under the assumption of truthful bidding ($b_{ij} = v_{ij}$), the social welfare maximization problem with alternative deadlines can be formulated into the following ILP:

$$\text{maximize} \quad \sum_{i\in[I]} \sum_{j\in[J]} b_{ij} x_{ij} \tag{3}$$

$$\text{subject to:} \quad y_i(t)t \le \sum_{j\in[J]} d_{ij} x_{ij}, \forall t \in [T], \forall i \in [I] : t_i \le t, \tag{3a}$$

$$\sum_{j\in[J]} w_i x_{ij} \le \sum_{t\in[T]:t_i\le t} y_i(t), \ \forall i \in [I], \tag{3b}$$

$$\sum_{i\in[I]:t_i\le t} r_i^k y_i(t) \le c_k, \ \forall k \in [K[, \forall t \in [T], \tag{3c}$$

$$\sum_{j\in[J]} x_{ij} \le 1, \ \forall i \in [I], \tag{3d}$$

$$x_{ij}, y_i(t) \in \{0,1\}, \forall i \in [I], \ \forall t \in [T], \forall j \in [J]. \tag{3e}$$

Note that the following constraint is redundant, and is not explicitly included in the ILP above: $y_i(t) \le \sum_{j\in[J]} x_{ij}, \forall i \in [I], \forall t \in [T]$. Constraint (3a) ensures that a job is scheduled to run between its arrival time and deadline. Constraint (3b) guarantees that the number of allocated slots is sufficient for serving a successful bid. The capacity limit of each type of resource is expressed in constraint (3c), and the alternative deadlines are modelled with the XOR bidding rule by (3d).

Even in the offline setting, ILP (3) without constraints (3a) and (3b) is still a NP-hard combinatorial optimization problem, equivalent to the classic knapsack problem. The challenge further escalates when we involve the jobs' deadlines and pursue online decision making. To address these challenges, we resort to the primal-dual algorithm design technique. In preparation, we first design a new framework to handle the unconventional constraints for deadline modelling. More specifically, we reformulate the original ILP (3) into a simplified *compact-exponential* ILP with a packing structure, at the price of involving an exponential number of variables:

$$\text{maximize} \quad \sum_{i\in[I]} \sum_{l\in\zeta_i} b_{il} x_{il} \tag{4}$$

$$\text{subject to:} \quad \sum_{i\in[I]} \sum_{l:t\in l} r_i^k x_{il} \le c_k, \ \forall k \in [K], \forall t \in [T], \tag{4a}$$

$$\sum_{l\in\zeta_i} x_{il} \le 1, \ \forall i \in [I], \tag{4b}$$

$$x_{il} \in \{0,1\}, \forall i \in [I], \ \forall l \in \zeta_i. \tag{4c}$$

Constraints (4a) and (4b) are equivalent to (3c) and (3d). ζ_i is the set of time schedules that satisfy constraints (3a) and (3b) for user i. The value of b_{il} is based on schedule l, and equals the corresponding b_{ij}. We relax the integrality constraints of x_{il} to $x_{il} \ge 0$ and formulate the dual problem. By introducing dual variables $p_k(t)$ and u_i to constraints (4a) and (4b) respectively, the dual LP of the relaxed (4) is:

$$\text{minimize} \quad \sum_{i\in[i]} u_i + \sum_{t\in[T]} \sum_{k\in[K]} c_k p_k(t) \tag{5}$$

$$\text{subject to:} \quad u_i \ge b_{il} - \sum_{k\in[K]} \sum_{t\in l} r_i^k p_k(t), \ \forall i \in [I], \forall l \in \zeta_i, \tag{5a}$$

$$p_k(t), u_i \ge 0, \forall i \in [I], \ \forall k \in [K], \forall t \in [T]. \tag{5b}$$

As we can observe, a feasible solution to ILP (4) has a corresponding feasible solution in ILP (3), and the optimal objective value of (4) is equal to that of (3). The number of variables in ILP (4) is exponential as the number of possible time schedules for user i is exponential in size. We next design an efficient primal-dual allocation scheme that only updates a polynomial number of variables, and can simultaneously solve optimization problems (3), (4) and (5).

B. Online Auction Design

In the auction algorithm, the cloud provider needs to deicide whether to accept a user i's job and if so, how to schedule its job to meet its deadline. If user i's jth bid with schedule

l is accepted, then let $x_{ij} = 1$, and update the variable $y_i(t)$ according to schedule l. To solve ILP (3), we adopt the primal-dual technique to the compact-exponential ILP (4) and its dual (5). For each primal variable x_{il} in (4), there is a dual constraint associated to it. Complementary slackness indicates the update of the primal variable is based on its dual constraint. x_{il} is zero unless its associated dual constraint (5a) is tight. Because the dual variable $u_i \geq 0$, we let u_i be the maximum of 0 and the right hand side (RHS) of (5a),

$$u_i = \max\{0, \max_{l \in \zeta_i}\{b_{il} - \sum_{t \in l} \sum_{k \in [K]} r_i^k p_k(t)\}\}. \quad (6)$$

Accordingly, the cloud provider accepts user i if $u_i > 0$, and serves user i's job according to the schedule that maximizes RHS of constraint (5a); if $u_i \leq 0$, the bid is rejected.

Algorithm 1 A Primal-dual Online Auction $A_{online1}$

Input: bidding language $\{B_i\}, \{c_k\}$

1: Define function $p_k(z_k(t))$ according to (7);
2: Initialize $x_{ij} = 0, y_i(t) = 0, z_k(t) = 0, u_i = 0, p_k(t) = 0, \forall i \in [I], \forall j \in [J], \forall k \in [K], \forall t \in [T]$; Let $x_{il} = 0, \forall i \in [I], \forall l \in \zeta_i$, by default;
3: **Upon the arrival of the ith user**
4: $(x_{ij}, \{y_i(t)\}, p_i, \{p_k(t)\}, \{z_k(t)\}) = A_{core1}(B_i, \{c_k\}, \{p_k(t)\}, \{z_k(t)\})$;
5: **if** $\exists j \in [J], x_{ij} = 1$ **then**
6: Accept user i's jth bid and allocated resources according to $y_i(t)$; Charge p_i for user i;
7: **else**
8: Reject user i.
9: **end if**

Algorithm 2 A Scheduling Algorithm A_{core1}

Input: bidding language $\{B_i\}, \{c_k\}, \{p_k(t)\}, \{z_k(t)\}$
Output: $x_{il}, p_i, \{p_k(t)\}, \{z_k(t)\}$

1: $c(t) = \sum_{k \in [K]} r_i^k p_k(t), \forall t \in [T]$; // price per slot
2: **for all** $j \in [J]$ **do**
3: Select w_i slots with minimum $(c(t))$ and $z_k(t) + r_i^k \leq c_k, \forall k \in [K]$ within $[t_i, d_{ij}]$, save the schedule in l_j;
4: $p_{ij} = \sum_{t \in l_j} c(t); u_{ij} = b_{ij} - p_{ij}$;
5: **end for**
6: $j^* = \arg\max_{j \in [J]}\{u_{ij}\}$;
7: **if** $u_{ij^*} > 0$ **then**
8: $x_{ij^*} = 1; y_i(t) = 1, \forall t \in l_{j^*}, p_i = p_{ij^*}$;
9: $x_{il_{j^*}} = 1$;
10: $u_i = u_{ij^*}; z_k(t) = z_k(t) + r_i^k, \forall k \in [K], t \in l_{j^*}$;
11: $p_k(t) = p_k(z_k(t)), \forall k \in [K], t \in l_{j^*}$;
12: **end if**
13: **Return** $x_{ij^*}, \{y_i(t)\}, p_i, \{p_k(t)\}, \{z_k(t)\}$

If we interpret dual variable $p_k(t)$ as the marginal price per unit of type-k resource at time t, then $\sum_{t \in l} \sum_{k \in [K]} r_i^k p_k(t)$ is the total charge that user i should pay when its job is assigned according to schedule l. The RHS of (5a) becomes the utility of bid i with schedule l. Thus, the assignment of u_i in (6) effectively maximizes user i's utility. This is a key step towards achieving social welfare maximization and truthfulness.

Note that although the calculation of u_i seems to take exponential time as the size of dual constraint (5a) is exponential, we design a dual oracle that selects only a polynomial number of dual constraints. We fix a set of schedules L_i with polynomial size through the dual oracle, and set $u_i = \max\{0, \max_{l \in L_i}\{b_{il} - \sum_{t \in l} \sum_{k \in [K]} r_i^k p_k(t)\}\}$. Then x_{il} is updated to 1 when $u_i > 0$. The dual oracle works as follow. For each deadline d_{ij} of user i's job, we select w_i slots with the minimum price for $t \in [t_i, d_{ij}]$, and let l_j be the corresponding schedule, and add l_j to set L_i. The schedule that maximizes user i's utility is the one with the minimum price in set L_i.

We next discuss the update of the dual variable $p_k(t)$. Recall that $p_k(t)$ represents the marginal price per unit of type-k resource at time t. We define a new variable $z_k(t)$ as the amount of allocated type-k resource at time t, and let the marginal price be a function of $z_k(t)$. $p_k(t)$ is increasing with the growth of z_k. Let U_k and L_k be the maximum and minimum values per unit of type-k resource per unit of time, respectively. $p_k(t)$ starts at L_k and exponentially increases when $z_k(t)$ is close to the capacity c_k. It reaches U_k when $z_k(t) = c_k$ because in this case, the cloud provider will never allocate any type-k resource to any user. In summary, $p_k(t)$ is defined as a function on $z_k(t)$ as following:

$$p_k(z_k(t)) = L_k \left(\frac{U_k}{L_k}\right)^{\frac{z_k(t)}{c_k}} \quad (7)$$

Where $U_k \leq \max_{i \in [I], j \in [J]} \frac{b_{ij}}{w_i r_i^k}$ and $L_k \geq \min_{i \in [I], j \in [J]} \frac{b_{ij}}{\sum_{k \in [K]} w_i r_i^k}$.

$A_{online1}$ in Alg. 1 with the schedule algorithm A_{core1} running for each user in Alg. 2 is the online auction. $A_{online1}$ first defines the price function and initializes the primal and dual variables in lines 1-2. Upon the arrival of each user i, we select the bid j^* with schedule l_{j^*} that maximizes user i's utility through the dual oracle (lines 2-5). If user i obtains positive utility, primal variables x_{ij^*} and $y_i(t)$ are updated according to schedule j^* (line 8). We then increase the usage for different resources ($z_k(t)$) and update the price ($p_k(t)$) for $t \in l_{j^*}$ (lines 10-11).

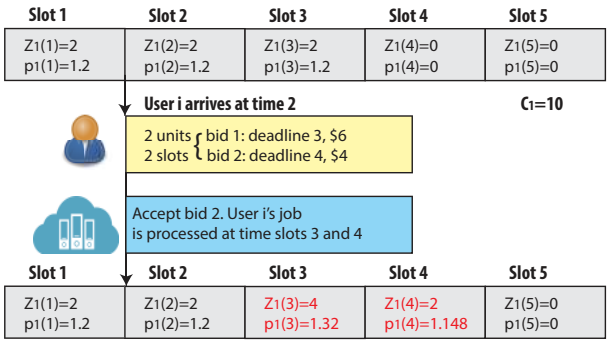


Fig. 1: An Example of the process in $A_{online1}$.

We next use an example to illustrate the winner determination process in $A_{online1}$, as shown in Fig. 1. Suppose the online system spans 5 time slots. A cloud data center hosts only one type of resource and the capacity is 10, *i.e.*, $c_1 = 10$. Assume $L_1 = 1$ and $U_1 = 2$. Before the arrival of user i, assume the marginal price per unit of resource at time t is $p_1(1) = p_1(2) = p_1(3) = 1.2; p_1(4) = p_1(5) = 0$. The amount of allocated resource at time t is $z_1(1) = z_1(2) = z_1(3) = 2; z_1(4) = z_1(5) = 0$. User i arrives at time 2,

requiring 2 units of resource and 2 time slots to execute its job. It submits two optional bids: it is willing to pay \$6 if its job is completed before time 3 or \$4 if its job is finished before time 4. The bidding price of the user i can be expressed as $B_i = \{2, 2, 2, \{3, \$6\}, \{4, \$4\}\}$. Upon the arrival of the user i, A_{core1} is executed to deicide whether to accept it and if so, how to schedule the job. The price per slot is calculated at line 1 in A_{core1} and $c(1) = c(2) = c(3) = 2.4; c(4) = c(5) = 0$. For the first bid of user i, lines 3-4 in A_{core1} compute the schedule, payment and utility of it: $l_1 = [2, 3]$, $p_{11} = 4.8$ and $u_{11} = 1.2$. For the second bid of user 1, $l_2 = [3, 4]$, $p_{12} = 2.4$ and $u_{12} = 1.6$. Now user i's maximum utility is larger than 0, *i.e.,* $u_{12} > 0$, primal and dual variables are updated accordingly at lines 8-11 in A_{core1}. Here $z_1(t)$ and $p_1(t)$ at slots 3 and 4 are updated, *i.e.,* $z_1(3) = 4, z_1(4) = 2$ and $p_1(3) = 1 \cdot 2^{\frac{4}{10}} \approx 1.32, p_1(4) = 1 \cdot 2^{\frac{2}{10}} \approx 1.148$. User i's second bid is accepted and its job is processed at time slots 3 and 4. The cloud provider charges \$2.4 for user i. This process is repeated until the last user's job is handled.

C. Theoretical Analysis

i) Correctness, Polynomial Time, and Truthfulness.

Theorem 1. $A_{online1}$ computes a feasible solution to ILP (3), ILP (4) and LP (5) in polynomial time.

Proof: *(Correctness):* $A_{online1}$ outputs a feasible solution for ILP (3) because line 3 in A_{core1} guarantees that the schedule l_j for user i's jth bid satisfies constraints (3a), (3b) and (3c). Constraint (3d) holds as only one bid per user can be accepted by A_{core1} in line 6. Furthermore, the corresponding relation between x_{ij} and x_{il} implies x_{il} is a feasible solution for ILP (4). For the dual problem (5), A_{core1} assigns 0 to u_i if $b_{il} \leq \sum_{k\in[K]} \sum_{t\in l} r_i^k p_k(t)$, and $b_{il} - \sum_{k\in[K]} \sum_{t\in l} r_i^k p_k(t)$ to u_i otherwise, ensuring the feasibility of $A_{online1}$.

(Polynomial running time): Lines 1-2 can be executed in linear time for the initialization of the cost function, primal and dual variables. Upon the arrival of user i, Algorithm A_{core1} first takes T steps to calculate the price of each slot. The `for` loop iterates J times to select the best slots for each bid. Line 3 in Alg. 2 takes $O(TK)$ time to schedule the job and check the capacity limit. Line 4 can be done in $O(1)$ steps. Thus, the running time of the `for` loop in Alg. 2 is $O(JKT)$. Then line 6 records the bid with the maximum utility in J steps. The body of the `if` statement (line 8-11) takes $O(KT)$ time to update the primal and dual variables and compute the payment. To sum up, the running time of A_{core1} is $O(JKT)$. The last step of $A_{online1}$ (lines 5-9) is to announce the auction decision, which can be done in constant time. In conclusion, $A_{online1}$ runs in polynomial time ($O(IJKT)$). □

Theorem 2. The online auction $A_{online1}$ is truthful.

Proof: Our auction $A_{online1}$ belongs to the family of *posted pricing mechanisms* [11]. Upon the arrival of user i, the payment that user i needs to pay to the cloud provider if it wins, depends only on the amount of resource that has been sold, and user i's demand. It is independent of user i's bidding price. Consequently, user i cannot improve its utility by lying about its bidding price as its utility equals its valuation minus the payment, *i.e.,* $u_{ij} = v_{ij} - p_i$. Furthermore, $A_{online1}$ always selects the schedule with the maximum utility among all possible schedules for user i. Hence, truthful bidding guarantees that each user obtains its maximum utility in $A_{online1}$. □

ii) Competitive Ratio.

We next examine the competitive ratio of our online auction. The competitive ratio is the upper-bound ratio of the social welfare achieved by the optimal solution of ILP (3) to the social welfare achieved by our online auction $A_{online1}$. We first introduce the primal-dual analysis framework in Lemma 1, which guides the final proof of the competitive ratio.

Let OPT_1 and OPT_2 denote the optimal objective values of ILP (3) and (4), respectively. We know that $OPT_1 = OPT_2$. Let P_i and D_i be the objective value of primal problem (4) and that of dual problem (5) returned by an algorithm after processing user i's bids. Let $P_0 = D_0 = 0$ be the initial values. Then P_I and D_I are the final primal and dual objective values achieved by the algorithm.

Lemma 1. If there exists a constant $\alpha_1 \geq 1$ such that $P_i - P_{i-1} \geq \frac{1}{\alpha_1}(D_i - D_{i-1})$ for all i, then the algorithm is α_1-competitive in social welfare.

Proof: When we sum up the inequalities for each i, we have $P_I = \sum_i (P_i - P_{i-1}) \geq \frac{1}{\alpha_1} \sum_i (D_i - D_{i-1}) = \frac{1}{\alpha_1} D_I$. According to weak duality [24], $D_I \geq OPT_2$, therefore $P_I \geq \frac{1}{\alpha_1} OPT_2 = \frac{1}{\alpha_1} OPT_1$. So we can conclude that the algorithm is α_1 competitive. □

$A_{online1}$ guarantees $P_0{=}D_0{=}0$. We next define an *Allocation-Price Relationship* and show that if it holds for a given α_1, then the primal and dual objective values achieved by $A_{online1}$ satisfy the inequality in Lemma 1. $p_k^i(t)$ denotes the price of type-k resource after handling user i. $z_k^i(t)$ is the amount of allocated type-k resource after processing i's job.

Definition 3. The Allocation-Price Relationship for $A_{online1}$ with $\alpha_1 \geq 1$ is $p_k^{i-1}(t)(z_k^i(t) - z_k^{i-1}(t)) \geq \frac{1}{\alpha_1} c_k(p_k^i(t) - p_k^{i-1}(t))$, $\forall i \in [I], \forall k \in [K], \forall t \in l$.

Lemma 2. If the Allocation-Price Relationship holds for a given $\alpha_1 \geq 1$, then $A_{online1}$ guarantees $P_i - P_{i-1} \geq \frac{1}{\alpha_1}(D_i - D_{i-1})$ for all $i \in [I]$.

Proof: If user i is rejected, then $P_i - P_{i-1} = D_i - D_{i-1} = 0$. In the following analysis, we assume that user i's jth bid is accepted, and let l be the schedule of user i's job. The increment of the primal objective value is: $P_i - P_{i-1} = b_{il}$. Note that $A_{online1}$ makes the constraint (5a) tight when bid b_{ij} with schedule l is accepted. Thus, $b_{il} = u_i + \sum_{k\in[K]} \sum_{t\in l} p_k^{i-1}(t)(z_k^i(t) - z_k^{i-1}(t))$.

The increase of the dual objective value is: $D_i - D_{i-1} = u_i + \sum_{k\in[K]} \sum_{t\in l} c_k(p_k^i(t) - p_k^{i-1}(t))$. By summing up the Allocation-Price Relationship over all $k \in [K]$ and $t \in l$, we can obtain: $P_i - P_{i-1} \geq u_i + \frac{1}{\alpha_1}(D_i - D_{i-1} - u_i)$. Since $u_i \geq 0$ and $\alpha_1 \geq 1$, it is obvious that $P_i - P_{i-1} \geq \frac{1}{\alpha_1}(D_i - D_{i-1})$. □

The Allocation-Price Relationship involves only the variables for type-k resource, we next try to find the corresponding $\alpha_{1,k}$ for each resource k that satisfies the Allocation-Price Relationship. The value of the approximation ratio α_1 is just the maximum value among all $\alpha_{1,k}$. In order to compute $\alpha_{1,k}$,

we assume that $r_i^k \ll c_k$, then $z_k^i(t) - z_k^{i-1}(t)$ can be expressed as $dz_k(t)$. The derivative of the Allocation-Price Relationship under the above assumption is:

Definition 4. The Differential Allocation-Price Relationship for $A_{online1}$ with $\alpha_{1,k} \geq 1$ is $p_k(t)dz_k(t) \geq \frac{c_k}{\alpha_{1,k}} dp_k(t), \forall i, k, t$.

Lemma 3. $\alpha_{1,k} = \ln \frac{U_k}{L_k}$ and the marginal price defined in (7) satisfies the Differential Allocation-Price Relationship.

Proof: The derivative of the marginal price function is: $dp_k(t) = p_k'(z_k(t)) = L_k(U_k/L_k)^{\frac{z_k(t)}{c_k}} \ln(U_k/L_k)^{\frac{1}{c_k}}$. The Differential Allocation-Price Relationship is:

$$L_k \left(\frac{U_k}{L_k}\right)^{\frac{z_k(t)}{c_k}} dz_k(t) \geq \frac{c_k}{\alpha_{1,k}} L_k \left(\frac{U_k}{L_k}\right)^{\frac{z_k(t)}{c_k}} \frac{1}{c_k} \ln \frac{U_k}{L_k} dz_k(t)$$
$$\Rightarrow \alpha_{1,k} \geq \ln \frac{U_k}{L_k}.$$

Therefore this lemma holds for $\alpha_{1,k} = \ln \frac{U_k}{L_k}$. □

Theorem 3. The online auction $A_{online1}$ in Alg. 1 is α_1-competitive in social welfare with $\alpha_1 = \max_{k\in[K]} \ln \frac{U_k}{L_k}$.

Proof: According to the proof in Lemma 3, $\alpha_1 = \max_{k\in[K]} \ln \frac{U_k}{L_k}$ satisfies the Differential Allocation-Price Relationship. Under the assumption that $dz_k(t) = z_k^i(t) - z_k^{i-1}(t))$ is much smaller than the capacity of type-k resource (c_k), we have $dp_k(t) = p_k'(z_k(t))dz_k(t) = p_k^i(t) - p_k^{i-1}(t)$. As a result, we can obtain the Allocation-Price Relationship holds for $\alpha_1 = \max_{k\in[K]} \ln \frac{U_k}{L_k}$. Then, combining Lemma 1 and Lemma 2 we finish the proof. □

V. Online Auction Design for the General Model with Penalty Function and Operation Cost

In this section, we present the online auction design for the general model that includes a penalty function and operation cost. We focus on the more challenging case of superlinear cost function with $\gamma_k > 0$. The auction design for linear cost with $\gamma_k = 0$ is similar and is omitted here.

A. Social Welfare Maximization Problem

Under the assumption of truthful bidding, the social welfare maximization problem in the general model is:

$$\text{maximize} \quad \sum_{i\in[I]} (b_i x_i - g_i(\tau_i)) - \sum_{t\in[T]} \sum_{k\in[K]} f_k(z_k(t)) \tag{8}$$

$$\text{subject to:} \quad y_i(t)t \leq d_i + \tau_i, \forall t \in [T], \forall i \in [I] : t_i \leq t, \tag{8a}$$

$$w_i x_i \leq \sum_{t\in[T]:t_i\leq t} y_i(t), \forall i \in [I], \tag{8b}$$

$$\sum_{i\in[I]:t_i\leq t} y_i(t) r_i^k \leq z_k(t), \forall k \in [K[, \forall t \in [T], \tag{8c}$$

$$\tau_i, z_k(t) \geq 0, x_i, y_i(t) \in \{0,1\}, \quad \forall i \in [I], \forall t \in [T], \forall k \in [K]. \tag{8d}$$

Again, constraint $y_i(t) \leq x_i, \forall i, t$ is redundant and is implied by the other constraints. Recall the definition of the cost function in (2) ($f_k(z_k(t)) = +\infty$ if $z_k(t) > c_k$), constraint (8c) guarantees that the amount of allocated resource never exceeds its capacity.

Let ζ_i be the set of time schedules that satisfy constraints (8a) and (8b) for user i, we adopt the same framework to reformulate the above convex optimization to the following compact-exponential convex problem:

$$\text{maximize} \quad \sum_{i\in[I]} \sum_{l\in\zeta_i} b_{il} x_{il} - \sum_{t\in[T]} \sum_{k\in[K]} f_k(z_k(t)) \tag{9}$$

$$\text{subject to:} \quad \sum_{i\in[I]} \sum_{l:t\in l} r_i^k x_{il} \leq z_k(t), \forall k \in [K], \forall t \in [T], \tag{9a}$$

$$\sum_{l\in\zeta_i} x_{il} \leq 1, \forall i \in [I], \tag{9b}$$

$$x_{il} \in \{0,1\}, z_k(t) \geq 0, \forall i \in [I], \forall l \in \zeta_i, \forall k \in [K], \forall t \in [T]. \tag{9c}$$

We introduce dual variables $p_k(t)$ and u_i to (9a) and (9b). The Fenchel dual [13] of the relaxed convex problem (9) is:

$$\text{minimize} \quad \sum_{i\in[i]} u_i + \sum_{t\in[T]} \sum_{k\in[K]} f_k^*(p_k(t)) \tag{10}$$

$$\text{subject to:} \quad u_i \geq b_{il} - \sum_{k\in[K]} \sum_{t\in l} r_i^k p_k(t), \forall i \in [I], \forall l \in \zeta_i, \tag{10a}$$

$$p_k(t), u_i \geq 0, \forall i \in [I], \forall k \in [K], \forall t \in [T]. \tag{10b}$$

Where $f_k^*(p_k(t))$ is the convex conjugate [25] of the cost function $f_k(\cdot)$, defined as: $f_k^*(p_k(t)) = \sup_{z_k(t)\geq 0}\{p_k(t)z_k(t) - f_k(z_k(t))\}$. The explicit expression of the conjugate is as following:

$$f_k^*(p_k(t)) = \begin{cases} \left(\frac{p_k(t)}{1+\gamma_k}\right)^{\frac{1+\gamma_k}{\gamma_k}} \cdot \frac{\gamma_k}{\beta_k^{\frac{1}{\gamma_k}}}, & z_k^0(t) \leq c_k \\ c_k p_k(t) - \beta_k c_k^{1+\gamma_k}, & z_k^0(t) > c_k \end{cases} \tag{11}$$

where $z_k^0(t) = (\frac{p_k(t)}{\beta_k(1+\gamma_k)})^{\frac{1}{\gamma_k}}$.

Proof:

$$f_k^*(p_k(t)) = \sup_{z_k(t)\geq 0} \begin{cases} p_k(t)z_k(t) - \beta_k z_k(t)^{1+\gamma_k}, & z_k(t) \in [0, c_k] \\ p_k(t)z_k(t) - \infty, & z_k(t) > c_k \end{cases}$$

We observe that $p_k(t)z_k(t) - \infty = -\infty$ when $z_k(t) > c_k$, thus we only need to obtain the conjugate of f when $z_k(t) \in [0, c_k]$,

Let $\psi_k(z_k(t)) = p_k(t)z_k(t) - \beta_k z_k(t)^{1+\gamma_k}$. The derivative of $\psi_k(z_k(t))$ with respect to $z_k(t)$ is :

$$\psi_k(z_k(t))' = p_k(t) - \beta_k(1+\gamma_k) z_k(t)^{\gamma_k}.$$

When we let $\psi_k(z_k^0(t))' = 0$, the local maximum happens at the point $z_k^0(t)$ and $z_k^0(t) = (\frac{p_k(t)}{\beta_k(1+\gamma_k)})^{\frac{1}{\gamma_k}}$.

Note that the domain of $z_k(t)$ is within the range $[0, c_k]$, therefore the supremum of $\psi_k(z_k(t))$ is $z_k^0(t)$ only if $z_k^0(t) \in [0, c_k]$. Otherwise, when $z_k^0(t) > c_k$, we can obtain that $\psi_k(z_k(t))' > 0$, which means $\psi_k(z_k(t))$ monotonically increases with the increment of $z_k(t)$ and the supremum happens at $z_k(t) = c_k$.

To sum up, we derive the conjugate of the cost function as shown in (11). □

B. Online Auction Design

We adopt the same posted pricing primal-dual framework from Sec. IV to solve the convex problem (8). Similarly, the primal-dual technique is applied to its compact-exponential problem (9) and its dual (10): the primal variable x_{il} remains zero unless its dual constraint (10a) becomes tight. The assignment of u_i is the same as that in (6).

Although there are an exponential size of dual constraints in the computation of u_i, we design a dual oracle based on dynamic programming to output a polynomial size of schedules, then only dual constraints associated to this set of schedules need to be considered.

The basic idea of the dual oracle is as follows. We fix the completion time of user i's job to be t_c ($t_c \in [t_i + w_i - 1, T]$), and construct the best schedule l_j with the minimum price in this case. The set that includes all such l_j has polynomial size, and is the output of the dual oracle. The construction of l_j is based on the dynamic programming method. The base case is the schedule l_0 with $t \in [t_i, t_i + w_i - 1]$. We move the completion time one slot forward each time. Let $c(t) = \sum_{k\in[K]} r_i^k p_k(t)$ be the price of user i's job running at time t. If the completion time t_c passes the deadline d_i, the corresponding penalty is added to the price, *i.e.,* $c(t) = \sum_{k\in[K]} r_i^k p_k(t) + g(t_c - d_i)$. When the old competition time is replaced with the new one, we only need to compare the price of the old competition time and $w_i - 1$ slots before the old competition time. For example, if user i arrives at time 1 with $w_i = 4$, then the basic case is $l_0 = \{1, 2, 3, 4\}$. Assume that $\arg\max_{t\in\{1,2,3\}} c(t) = 2$. We next fix the completion time to 5, the best schedule is then $\{1, 4, 3, 5\}$ if $c(4) < c(2)$ and $\{1, 2, 3, 5\}$ otherwise. The process is repeated until the completion time reaches T.

The marginal price $p_k(t)$ per unit of type-k resource at time t can be defined as the derivative of the cost function, *i.e.,* ${f_k}'(\hat{z_k}(t))$ if the overall demand of resource k at t ($\hat{z_k}(t)$) is known. But in the online setting, it is impossible for the cloud provider to acquire the complete knowledge of the system. The cloud provider predicts the final demand at future slots as $\theta_k (\theta_k > 1)$ times of the current demand at those slots if the predicted final demand is below the capacity, and set the marginal price to ${f_k}'(\theta_k z_k(t))$ where $z_k(t)$ is the amount of current allocated resource k at t. Let U_k' be the maximum value per unit of type-k resource per unit of time. The marginal price grows exponentially when the predicted demand is larger than the capacity, and reaches U_k' if $z_k(t) = c_k$. More specifically, the marginal price function is defined as:

$$p_k(z_k(t)) = \begin{cases} f_k'(\theta_k z_k(t)), & z_k(t) \le \frac{c_k}{\theta_k} \\ f_k'(c_k) e^{\rho_k (z_k(t) - \frac{c_k}{\theta_k})}, & z_k(t) > \frac{c_k}{\theta_k} \end{cases} \quad (12)$$

with parameters $\theta_k = \max\{2, (1+\gamma_k)^{\frac{1}{\gamma_k}}\}$,

$$\rho_k = \max\{\frac{\theta_k}{c_k}\gamma_k, \frac{\theta_k}{c_k(\theta_k - 1)} \ln(\frac{U_k'}{\beta_k(1+\gamma_k)c_k^{\gamma_k}})\},$$

where $U_k' \le \max_{i\in[I]} \frac{b_i}{w_i r_i^k}$.

The online auction $A_{online2}$ for the general model is presented in Alg. 3. Upon the arrival of the ith user, $A_{online2}$ calls A_{core2} in Alg. 4 to make decision. A_{core2} computes the best schedule for user i through the dual oracle (lines 1-10) to maximize its utility u_i. If $u_i > 0$, the corresponding primal and dual variables are updated in lines 14-17.

Algorithm 3 A Primal-dual Online Auction $A_{online2}$

Input: bidding language $\{B_i\}, \{c_k\}, \{\beta_k, \gamma_k\}$

1: Define cost function $f_k(z_k(t))$ according to (2);
2: Define function $p_k(z_k(t))$ according to (12);
3: Initialize $x_i = 0, y_i(t) = 0, z_k(t) = 0, \tau_i = 0, u_i = 0, p_k(t) = 0, \forall i \in [I], \forall k \in [K], \forall t \in [T]$; Let $x_{il} = 0, \forall i \in [I], \forall l \in \zeta_i$, by default;
4: **Upon the arrival of the ith user**
5: $(x_i, \{y_i(t)\}, p_i, \{p_k(t)\}, \{z_k(t)\}) = A_{core2}(B_i, \{c_k\}, \{p_k(t)\}, \{z_k(t)\})$;
6: **if** $x_i = 1$ **then**
7: Accept user i's bid and allocated resources according to $y_i(t)$; Charge p_i for user i;
8: **else**
9: Reject user i.
10: **end if**

Algorithm 4 A Scheduling Algorithm A_{core2}.

Input: $B_i, \{c_k\}, \{p_k(t)\}, \{z_k(t)\}$
Output: $x_i, \{y_i(t)\}, p_i, \{p_k(t)\}, \{z_k(t)\}$

1: Add slot $t \in [t_i, T]$ to set $\mathcal{T}$ if $z_k(t) + r_i^k \le c_k, \forall k \in [K]$;
2: Let schedule l_0 include the first w_i slots $(t_1, t_2, \ldots, t_{w_i})$ in $\mathcal{T}$; Define $j = 1$;
3: **while** $w_i + j \le |\mathcal{T}|$ **do**
4: $l_j = l_{j-1}$;
5: Let t_c is the $(w_i + j)$th slot in $\mathcal{T}$;
6: $c(t) = \sum_{k\in[K]} r_i^k p_k(t), \forall t \in \{t_1, t_2, \ldots, t_{w_i}, t_c\}$;
7: If $t_c > d_i$, $c(t_c) = c(t_c) + g(t_c - d_i)$;
8: $t_m = \arg\max_{t\in\{t_1,\ldots,t_{w_i-1}\}} c(t)$;
9: If $c(t_{w_i}) < c(t_m)$, for schedule l_j, replace the slot t_m with t_{w_i} and save t_c into t_{w_i};
10: $\mathcal{P}_j = \sum_{t\in l_j} c(t)$; $j = j + 1$;
11: **end while**
12: $j^* = \arg\min_j \{\mathcal{P}_j\}$;
13: **if** $b_i - \mathcal{P}_{j^*} > 0$ **then**
14: $x_i = 1; y_i(t) = 1, \forall t \in l_{j^*}$; $x_{il_{j^*}} = 1$;
15: $u_i = b_i - \mathcal{P}_{j^*}; p_i = \sum_{k\in[K]} \sum_{t\in l_{j^*}} r_i^k p_k(t)$;
16: $z_k(t) = z_k(t) + r_i^k, \forall k \in [K], t \in l_{j^*}$;
17: $p_k(t) = p_k(z_k(t)), \forall k \in [K], t \in l_{j^*}$;
18: **end if**
19: **Return** $x_i, \{y_i(t)\}, p_i, \{p_k(t)\}, \{z_k(t)\}$

C. Theoretical Analysis

i) Correctness, Polynomial Time, and Truthfulness.

Lemma 4. The running time of A_{core2} is $O(KT + T^2)$.

Proof: Line 1 initializes a feasible slot set $\mathcal{T}$ in $O(KT)$ steps. Line 2 takes w_i steps to define a schedule l_o The `while` loop (lines 3-11) is to compute the best schedule l_j if the completion time is fixed, will iterate at most $T - w_i$ times. Within the `while` loop body, lines 4-7 takes $O(w_i + 1)$ steps to update $c(t)$. The running time of finding the maximum price in line 8 is linear to w_i. Lines 9-10 takes constant time for the comparison and addition. Thus, the `while` loop can be executed in $O((T - w_i)w_i)$ steps. Line 12 can be done in $O(T - w_i)$ steps to find the schedule with the minimum price. The running time to execute the `if` body is $O(KT)$. In summary, the running time of A_{core2} is $O(KT + w_i(T - w_i)) \le O(KT + T^2)$. □

Theorem 4. $A_{online2}$ in Alg. 3 is a truthful auction that returns

feasible solutions for convex problems (8), (9) and (10) in polynomial running time.

Proof: (*Polynomial running time:*) The running time of the initialization process in lines 1-3 is linear. By Lemma 4, A_{core2} in line 5 processes each user in $O(KT+T^2)$ time. The `If` statement in lines 6-10 can be done within constant time. Therefore, after handling the last user, the overall running time of $A_{online2}$ is $O(I(KT+T^2))$.

(*Correctness and Truthfulness:*) We omit the proof here as similar proofs can be found in Theorem 1 and Theorem 2. □

ii) Competitive Ratio.

The proof follows the same structure as that in Sec. IV-C. Let P_i and D_i be the primal (9) and dual (10) objective values achieved by $A_{online2}$ after handling user i's request. By Lemma 1, $A_{online2}$ is α_2-competitive in social welfare if there is a constant $a_2 \geq 1$ such that $P_i - P_{i-1} \geq \frac{1}{\alpha_2}(D_i - D_{i-1})$ for all i. We next define the Allocation-Price Relationship for $A_{online2}$, and show that if the Allocation-Price Relationship holds for a given α_2, then $P_i - P_{i-1} \geq \frac{1}{\alpha_2}(D_i - D_{i-1})$ also holds. The last step is to define the differential version of the Allocation-Price Relationship and prove there exists a $\alpha_{2,k}$ that satisfies this relationship. By setting $\alpha_2 = \max_{k\in[K]}\{\alpha_{2,k}\}$, we can obtain the competitive ratio of $A_{online2}$.

Definition 5. The Allocation-Price Relationship for $A_{online2}$ with $\alpha_2 \geq 1$ is $p_k^{i-1}(t)\big(z_k^i(t) - z_k^{i-1}(t)\big) - \big(f_k(z_k^i(t)) - f_k(z_k^{i-1}(t))\big) \geq \frac{1}{\alpha_2}\big(f_k^*(p_k^i(t)) - f_k^*(p_k^{i-1}(t))\big), \forall i, \forall k, \forall t \in l$.

Lemma 5. If the Allocation-Price Relationship for $A_{online2}$ holds with a given $\alpha_2 \geq 1$, then $A_{online2}$ guarantees $P_i - P_{i-1} \geq \frac{1}{\alpha_2}(D_i - D_{i-1})$ for all $i \in [I]$.

Proof: If user i is rejected, then $P_i - P_{i-1} = D_i - D_{i-1} = 0$. In the next analysis, we assume that user i is accepted, and let l be the schedule of user i's job. The increment of the primal objective value is:

$$\begin{aligned} P_i - P_{i-1} &= b_{il} - \sum_{t\in l}\sum_{k\in[K]}\big(f_k(z_k^i(t)) - f_k(z_k^{i-1}(t))\big) \\ &= u_i + \sum_{k\in[K]}\sum_{t\in l} p_k^{i-1}(t)\big(z_k^i(t) - z_k^{i-1}(t)\big) \\ &\quad - \sum_{t\in l}\sum_{k\in[K]}\big(f_k(z_k^i(t)) - f_k(z_k^{i-1}(t))\big). \end{aligned}$$

The second equality holds because $A_{online2}$ update the value of dual variables such that dual constraint becomes tight and $r_i^k = z_k^i(t) - z_k^{i-1}(t)$. Then the increase of the dual objective value is:

$$D_i - D_{i-1} = u_i + \sum_{t\in l}\sum_{k\in[K]}\big(f_k^*(p_k^i(t)) - f_k^*(p_k^{i-1}(t))\big)$$

By summing up the Allocation-Price Relationship for $A_{online2}$ over all $k \in [K]$ and $t \in l$, we can obtain:

$$P_i - P_{i-1} \geq u_i + \frac{1}{\alpha_2}(D_i - D_{i-1} - u_i).$$

Since $u_i \geq 0$ and $\alpha_1 \geq 0$, it is obvious that $P_i - P_{i-1} \geq \frac{1}{\alpha_2}(D_i - D_{i-1})$. □

Definition 6. The Differential Allocation-Price Relationship for $A_{online2}$ with $\alpha_{2,k} \geq 1$ is: $p_k(t)dz_k(t) - f_k'(z_k(t))dz_k(t) \geq \frac{1}{\alpha_{2,k}} f_k^{*\prime}(p_k(t))dp_k(t), \forall i, \forall k, \forall t \in l$.

Lemma 6. $\alpha_{2,k} = \max\{4(1+\gamma_k), \frac{2(1+\gamma_k)}{\gamma_k}\ln(\frac{U_k'}{\beta_k(1+\gamma_k)c_k^{\gamma_k}})\}$ and the marginal price function defined in (12) satisfy the Differential Allocation-Price Relationship.

Proof: We first write down the explicit expressions for the differentials of the cost function (2) and its convex conjugate (11):

$$f_k'(z_k(t)) = \begin{cases} \beta_k(1+\gamma_k)z_k(t)^{\gamma_k}, & \text{if } z_k(t) \in [0, c_k] \\ +\infty, & \text{otherwise} \end{cases}$$

$$f_k^{*\prime}(z_k(t)) = \begin{cases} \Big(\frac{p_k(t)}{\beta_k(1+\gamma_k)}\Big)^{\frac{1}{\gamma_k}}, & p_k(t) \leq \beta_k(1+\gamma_k)c_k^{\gamma_k} \\ c_k, & p_k(t) > \beta_k(1+\gamma_k)c_k^{\gamma_k} \end{cases}$$

When the amount of allocated type-k resource reaches the capacity, *i.e.,* $z_k(t) = c_k$, according to the definition of marginal price in (12),

$$p_k(t) = \beta_k(1+\gamma_k)c_k^{\gamma_k}e^{\rho_k(c_k - \frac{c_k}{\theta_k})} \geq U_k'.$$

Recall that U_k' is the maximum value per unit of resource k per unit of time. It is clear when the marginal price is larger than U_k', no bids can win. Thus, we may assume $z_k(t) \leq c_k$ in the rest of the proof, and $f_k'(z_k(t)) = \beta_k(1+\gamma_k)z_k(t)^{\gamma_k}$. Next, we divide our proof into two cases:

Case 1: $z_k(t) \leq \frac{c_k}{\theta_k}$: Because $p_k(t) = f'(\theta_k z_k(t)) = \beta_k(1+\gamma_k)(\theta_k z_k(t))^{\gamma_k} \leq \beta_k(1+\gamma_k)c_k^{\gamma_k}$, the Differential Allocation-Price Relationship can be rewritten as:

$$\begin{aligned} &(\beta_k(1+\gamma_k)(\theta_k z_k(t))^{\gamma_k} - \beta_k(1+\gamma_k)z_k(t)^{\gamma_k})dz_k(t) \\ &\geq \frac{1}{\alpha_{2,k}}\Big(\frac{p_k(t)}{\beta_k(1+\gamma_k)}\Big)^{\frac{1}{\gamma_k}}\beta_k(1+\gamma_k)\theta_k^{\gamma_k}\gamma_k z_k(t)^{\gamma_k - 1}dz_k(t). \quad (13) \end{aligned}$$

Cancelling the common term on both sides, (13) becomes $(\theta_k^{\gamma_k} - 1) \geq \frac{1}{\alpha_{2,k}}\gamma_k\theta^{\gamma_k+1}$. **i)** If $\gamma_k \geq 1$, $\theta_k = \max\{2, (1+\gamma_k)^{\frac{1}{\gamma_k}}\} = 2$, we can obtain

$$\begin{aligned} \frac{\gamma_k\theta_k^{\gamma_k+1}}{\theta_k^{\gamma} - 1} &= \frac{\gamma_k 2\cdot 2^{\gamma_k}}{2^{\gamma} - 1} = \frac{\gamma_k(4\cdot 2^{\gamma_k} - 2\cdot 2^{\gamma_k})}{2^{\gamma_k} - 1} \\ &\leq \frac{4\gamma_k(2^{\gamma_k} - 1)}{2^{\gamma_k} - 1} \leq 4\gamma_k < \alpha_{2,k} \end{aligned}$$

ii) If $\gamma_k < 1$, then $\theta_k = (1+\gamma_k)^{\frac{1}{\gamma_k}} < e$, and

$$\frac{\gamma_k\theta_k^{\gamma_k+1}}{\theta_k^{\gamma} - 1} = \theta_k(1+\gamma_k) < e(1+\gamma_k) < \alpha_{2,k}.$$

Case 2: $z_k(t) > \frac{c_k}{\theta_k}$: In this case, the marginal price $z_k(t)$ is:

$$p_k(t) = \beta_k(1+\gamma_k)c_k^{\gamma_k}e^{\rho_k(z_k(t) - \frac{c_k}{\theta_k})}.$$

Note that $dp_k(t) = \rho_k p_k(t)dz_k(t)$, then the Differential Allocation-Price Relationship is:

$$(p_k(t) - f_k'(z_k(t)))dz_k(t) \geq \frac{1}{\alpha_{2,k}}c_k\rho_k p_k(t)dz_k(t). \quad (14)$$

By Lemma 7, we can obtain $p_k(t) - f_k'(z_k(t)) \geq p_k(t) - \frac{1}{1+\gamma_k}p_k(t) \geq \frac{\gamma_k}{1+\gamma_k}p_k(t)$, thus to prove (14), it is sufficient to prove:

$$\frac{\gamma_k}{1+\gamma_k}p_k(t)dz_k(t) \geq \frac{1}{\alpha_{2,k}}c_k\rho_k p_k(t)dz_k(t) \Rightarrow \rho_k \leq \frac{\gamma_k}{c_k(1+\gamma_k)}\alpha_{2,k}.$$

By the value of ρ_k, either **i)**

$$\begin{aligned} \rho_k &= \frac{\theta_k}{c_k}\gamma_k \leq \frac{e}{c_k}\gamma_k \\ &= \frac{\gamma_k}{c_k(1+\gamma_k)}e(1+\gamma_k) \leq \frac{\gamma_k}{c_k(1+\gamma_k)}\alpha_{2,k}. \end{aligned}$$

$$\text{Or ii) } \rho_k = \frac{\theta_k}{c_k(\theta_k-1)}\ln(\frac{U_k'}{\beta_k(1+\gamma_k)c_k^{\gamma_k}}) \le \frac{2}{c_k}\ln(\frac{U_k'}{\beta_k(1+\gamma_k)c_k^{\gamma_k}}) = \frac{\gamma_k}{c_k(1+\gamma_k)}\frac{2(1+\gamma_k)}{\gamma_k}\ln(\frac{U_k'}{\beta_k(1+\gamma_k)c_k^{\gamma_k}}) \le \frac{\gamma_k}{c_k(1+\gamma_k)}\alpha_{2,k}.$$

In conclusion, we have finished the proof for both cases. □

Lemma 7. When $z_k(t) > \frac{c_k}{\theta_k}$, the marginal price $p_k(t)$ is larger than the marginal cost by a factor of at least $1+\gamma_k$:

$$p_k(t) \ge (1+\gamma_k)f_k'(z_k(t)).$$

Proof: When $z_k(t) > \frac{c_k}{\theta_k}$, $p_k(t) = \beta_k(1+\gamma_k)c_k^{\gamma_k}e^{\rho_k(z_k(t)-\frac{c_k}{\theta_k})}$. So Lemma 7 is equivalent to verify

$$\frac{e^{\rho_k z_k(t)}}{z_k(t)^{\gamma_k}} \ge \frac{(1+\gamma_k)e^{\frac{\rho_k c_k}{\theta_k}}}{c_k^{\gamma_k}} \tag{15}$$

We first show that the inequality (15) holds when $z_k(t) = \frac{c_k}{\theta_k}$. If $z_k(t)$ takes the value of $\frac{c_k}{\theta_k}$, (15) becomes $\theta_k^{\gamma_k} \ge 1+\gamma_k$ which is obviously true.

Next, it suffices to show the left side of (15) is non-decreeing as $z_k(t)$ increases. Let $L(z_k(t))$ denote the left hand of (15). The derivative of $L(z_k(t))$ is

$$L'(z_k(t)) = \frac{e^{\rho_k z_k(t)}(\rho_k z_k(t) - \gamma_k)}{z_k(t)^{1+\gamma_k}}.$$

Because $\rho_k \ge \frac{\theta_k}{c_k}\gamma_k$ and $z_k(t) > \frac{c_k}{\theta_k}$, then $\rho_k z_k(t) - \gamma_k \ge 0$ and the derivative $L'(z_k(t))$ is nonnegative. Consequently, the lemma follows. □

Theorem 5. The online auction $A_{online2}$ in Alg. 3 is α_2-competitive in social welfare with $\alpha_2 = \max_{k\in[K]}\alpha_{2,k}$.

Proof: Because α_2 is the maximum number among all $\alpha_{2,k}$, then Differential Allocation-Price Relationship also holds with α_2. We assume that $dz_k(t) = z_k^i(t) - z_k^{i-1}(t)$ is much smaller than the capacity of type-k resource (c_k), then

$$f_k(z_k^i(t)) - f_k(z_k^{i-1}(t)) = f_k'(z_k^{i-1}(t))(z_k^i(t) - z_k^{i-1}(t)),$$
$$f_k^*(p_k^i(t)) - f_k^*(p_k^{i-1}(t)) = f_k^{*\prime}(p_k^{i-1}(t))(p_k^i(t) - p_k^{i-1}(t)).$$

Therefore, the Allocation-Price Relationship holds with α_2. Combinng Lemma 1 and Lemma 5, we finish the proof. □

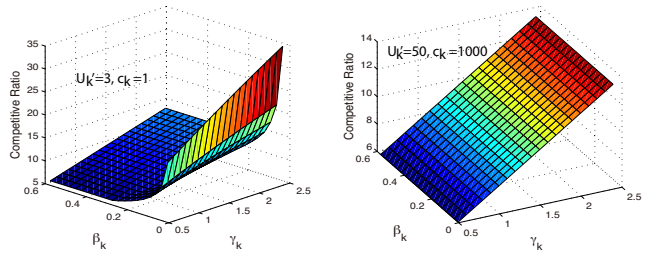


Fig. 2: An illustration of the competitive ratio of $A_{online2}$ (α_2) under different settings.

We plot the value of α_2 in Fig. 2 when we vary the value of γ_k, β_k, c_k and U_k' [22], [23]. We can observe that if we normalize c_k to 1, the competitive ratio of $A_{online2}$ is close to 6 with a small U_k' and γ_k, as demonstrated in the left figure. The right figure shows that if c_k is a large number, the competitive ratio is determined by γ_k and increases with the increment of γ_k.

VI. Performance Evaluation

We evaluate our online auctions $A_{online1}$ and $A_{online2}$ through trace-driven simulation studies. We exploit the trace version 1 in Google Cluster Data [26], which contains the information for each job including the start time, execution duration, and resource demands (CPU and RAM). We translate each job into a bid, arriving sequentially in 18 hours. We assume that each user's job consumes $[1, 12]$ slots and each time slot is 5 minutes [26]. User's job deadline is generated uniformly at random between its arrival time and the system end time. The bidding price of each job equals its overall resource demand times unit prices randomly picked in the range $[L_k, U_k]$. By default, $L_k = 1$ and $U_k' = U_k = 50$. The demand for CPU and RAM units is normalized so that the maximum capacity is 1. For the cost function, β_k is set within $[0.4, 0.6]$ for CPU and within $[0.005, 0.02]$ for RAM [22]. γ_k is set within $[1.7, 2.2]$ for CPU and within $[0.5, 1]$ for RAM [23].

Performance of $A_{online1}$. We examine the performance of $A_{online1}$ in terms of the competitive ratio, social welfare and user satisfaction.

Fig. 3 shows the competitive ratio of $A_{online1}$ with different number of users (I) and bids per user (J). The observed competitive ratio is much better than the theoretical bound and remains at a low level (< 2). It fluctuates with the increase of the number of users and sightly decreases when the number of bids per user grows. This is because when each user provides a larger number of optimal bids, $A_{online1}$ is more likely to optimize the schedule of its job, leading to a better performance. In $A_{online1}$, the marginal price function is defined based on the real value of U_k and L_k. We vary the value of U_k/L_k, and use the estimated values of U_k as the input of $A_{online1}$, to examine the performance. As shown in Fig. 4, there is a downward trend as the value of U_k/L_k decreases, while there is no large difference with either underestimation and overestimation. The observation confirms the analysis in Theorem 3 that the value of U_k/L_k determines the competitive ratio. Underestimation is more desirable than the overestimation, as compared to that achieved by the real U_k (labelled by 100%). Overestimation makes the price rise more rapidly, filtering out users that are supposed to be accepted.

We next study the social welfare achieved by $A_{online1}$ in Fig. 5 and Fig. 6. The 3d figure in Fig. 5 plots the social welfare under different number of users and bids per user. Our online auction $A_{online1}$ achieves a higher social welfare when there is larger number of users participating the auction. The change of bids per user doesn't have major influence on the social welfare. When the number of users grows, the number of bids with larger bidding price also increases. As a result, A_{online} returns a larger social welfare. The social welfare under different number of slots and U_k/L_k is illustrated in

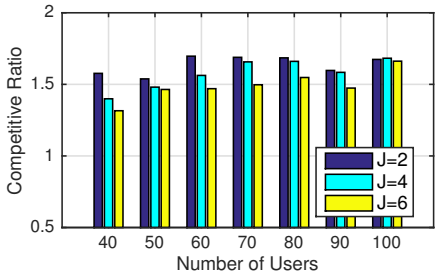


Fig. 3: Competitive ratio of $A_{online1}$ with different number of users and J.

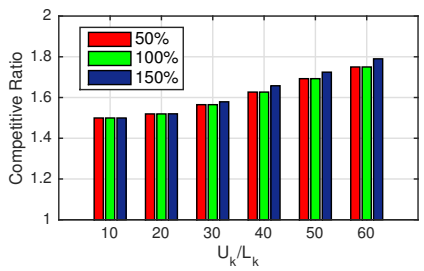


Fig. 4: Competitive ratio of $A_{online1}$ with different U_k/L_k.

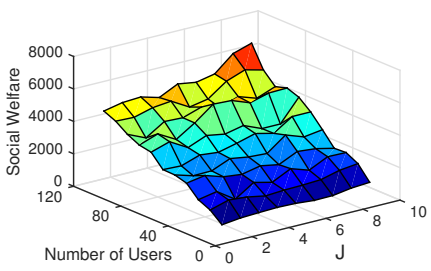


Fig. 5: Social welfare of $A_{online1}$ with different number of users and J.

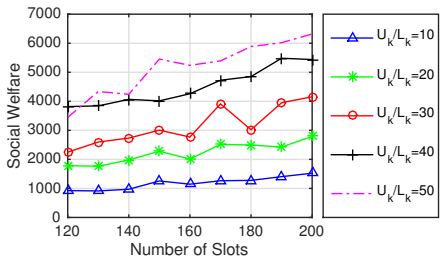


Fig. 6: Social welfare of $A_{online1}$ with different T and U_k/L_k

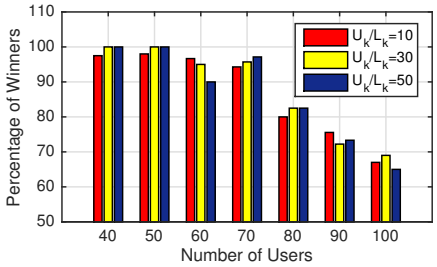


Fig. 7: Percentage of winners in $A_{online1}$ with different I and U_k/L_k

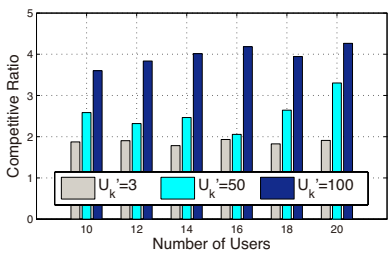


Fig. 8: Competitive ratio of $A_{online2}$ with different I and U_k'

Fig. 6. Both the number of slots and the value of U_k/L_k influence the social welfare. $A_{online1}$ is able to allocate more jobs when the length of the system increases. Furthermore, the bidding price rises with the increase of U_k/L_k, thus high value bids lead to a higher social welfare.

User satisfaction which is measured by the percentage of winners is demonstrated in Fig. 7. A higher fraction of users are accepted with a small number of users. This is because the number of winners is almost fixed due to the capacity limit. We also obverse that the value of U_k/L_k doesn't influence the percentage as the winner determination process is not affected by the change of U_k/L_k.

Performance of $A_{online2}$.

We first examine the competitive ratio of $A_{online2}$. We use CVX with the Gurobi Optimizer to solve the convex problem (8) exactly, and compute the competitive ratio by dividing the optimal social welfare by the social welfare returned by $A_{online2}$. However, CVX fails to solve in 24 hours even with a medium-size input. Thus, we reduce the input size and only consider 10-20 users. Fig. 8 shows the competitive ratio of $A_{online2}$ under different number of users and U_k'. It becomes larger with the increase of U_k'. The change of the number of users doesn't have much impact on the competitive ratio. As indicated in Theorem 5, a larger U_k' negatively influences the competitive ratio when we set c_k to 1. We can also observe that the competitive ratio is still less than 5 with a large U_k', which is much better than the theoretical bound.

We next study the performance of $A_{online2}$ in the aspects of social welfare and user satisfaction. Fig. 9 shows the social welfare and cloud provider's revenue with different number of users when we vary the value of U_k'. We can obverse that both the social welfare and revenue increase with the increment of number of users and U_k'. The reason for it has been explained

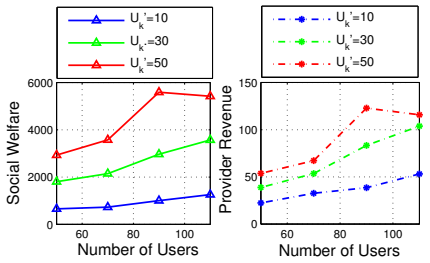


Fig. 9: Social welfare and cloud provider's revenue in $A_{online2}$ with I and U_k'.

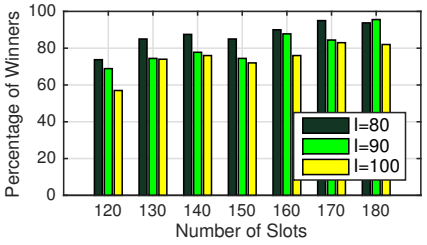


Fig. 10: Percentage of winners in $A_{online2}$

when we evaluate the performance of $A_{online1}$ and is omitted here, as the design of $A_{online2}$ follows the same primal-dual technique. Fig. 10 reflects the percentage of winners gradually rises when the number of slots increases. The possibility of wining becomes higher when the system spans a long period as there are more slots available for scheduling. In addition, a small number of users leads to higher user satisfaction.

VII. Conclusions

We studied the auction design for cloud computing jobs that have soft completion deadlines. Our main contribution is an online cloud job auction that is truthful and computationally efficient, and achieves a good competitive ratio in social

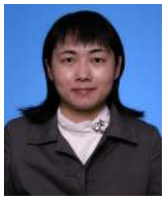
welfare. Techniques used in the auction design include the posted pricing framework for truthful online auctions, a new LP formulation and solution method for handling soft deadline constraints, as well as approximation algorithms based on LP dual and Fenchel dual. Our method for handling soft deadline constraints may be applicable to other auction design problems where deadline is involved, for example, in demand response auctions in a smart grid.

REFERENCES

[1] *Amazon EC2 instance type*, http://aws.amazon.com/ec2/instance-types/.
[2] *Microsoft Azure*, http://azure.microsoft.com/en-us/?rnd=1.
[3] *Linode*, https://www.linode.com/.
[4] L. Zhang, S. Ren, C. Wu, and Z. Li, "A truthful incentive mechanism for emergency demand response in colocation data centers," Tech. Rep., 2015.
[5] RightScale, *Social Gaming in the Cloud: A Technical White Paper*, 2013.
[6] M. Hajjat, X. Sun, Y. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: planning for beneficial migration of enterprise applications to the cloud," *Proc. of ACM SIGCOMM*, 2011.
[7] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proc. of IEEE INFOCOM*, 2014.
[8] W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," in *Proc. of ACM SIGMETRICS*, 2014.
[9] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. Lau, "Online auctions in IaaS clouds: welfare and profit maximization with server costs," in *Proc. of ACM SIGMETRICS*, 2015.
[10] N. Buchbinder and J. Naor, "Online primal-dual algorithms for covering and packing problems," in *ESA*. Springer, 2005, pp. 689–701.
[11] Z. Huang and A. Kim, "Welfare maximization with production costs: a primal dual approach," in *Proc. of the ACM-SIAM SODA*, 2015.
[12] Wikipedia, *Dynamic frequency scaling*, http://en.wikipedia.org/wiki/Dynamic_frequency_scaling.
[13] N. R. Devanur, "Fisher markets and convex programs," *JACM*, 2010.
[14] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing," in *Proc. of IEEE INFOCOM*, 2012.
[15] S. Zaman and D. Grosu, "Combinatorial auction-based dynamic vm provisioning and allocation in clouds," in *Proc. of IEEE Cloud CloudCom*, 2011.
[16] H. Zhang, B. Li, H. Jiang, F. Liu, A. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," in *Proc. of IEEE INFOCOM*, 2013.
[17] B. Lucier, I. Menache, J. S. Naor, and J. Yaniv, "Efficient online scheduling for deadline-sensitive jobs," in *Proc. of ACM SPAA*, 2013.
[18] N. Jain, I. Menache, J. S. Naor, and J. Yaniv, "Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters," *ACM Transactions on Parallel Computing*, vol. 2, no. 1, p. 3, 2015.
[19] Y. Azar, I. Kalp-Shaltiel, B. Lucier, I. Menache, J. S. Naor, and J. Yaniv, "Truthful online scheduling with commitments," in *Proc. of ACM EC*, 2015.
[20] N. Buchbinder and J. Naor, "The design of competitive online algorithms via a primal: dual approach," *Foundations and Trends® in Theoretical Computer Science*, vol. 3, no. 2-3, pp. 93–263, 2009.
[21] A. Blum, A. Gupta, Y. Mansour, and A. Sharma, "Welfare and profit maximization with production costs," in *Proc. of IEEE FOCS*, 2011.
[22] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. of ACM SoCC*, 2010.
[23] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of virtual machines for real-time cloud services," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 13, pp. 1491–1505, 2011.
[24] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[25] Wikipedia, *Convex conjugate*, http://en.wikipedia.org/wiki/Convex_conjugate.
[26] *Google Cluster Data, TraceVersion1*, https://code.google.com/p/googleclusterdata/wiki/TraceVersion1.

Ruiting Zhou received a B.E. degree in telecommunication engineering from Nanjing University of Post and Telecommunication, China, in 2007, a M.S. degree in telecommunications from Hong Kong University of Science and Technology, Hong Kong, in 2008 and a M.S. degree in computer science from University of Calgary, Canada, in 2012. Since March, 2016, she has been a PhD candidate at the Department of Computer Science, University of Calgary, Canada. Her research interests include smart grids, cloud computing and mobile network optimization.

Zongpeng Li received his B.E. degree in Computer Science and Technology from Tsinghua University (Beijing) in 1999, his M.S. degree in Computer Science from University of Toronto in 2001, and his Ph.D. degree in ECE from University of Toronto in 2005. He has been with the University of Calgary since 2005, where he is now Professor of Computer Science. In 2011-2012, Zongpeng was a visitor at the Institute of Network Coding, Chinese University of Hong Kong. His research interests are in computer networks, network coding, cloud computing, and energy networks.



Chuan Wu received her B.Engr. and M.Engr. degrees in 2000 and 2002 from the Department of Computer Science and Technology, Tsinghua University, and her Ph.D. degree in 2008 from the Department of Electrical and Computer Engineering, University of Toronto, Canada. Between 2002 and 2004, She worked in the Information Technology industry in Singapore. Since September 2008, Chuan Wu has been with the Department of Computer Science at the University of Hong Kong, where she is currently an Associate Professor. Her research is in the areas of cloud computing, data center networking, online and mobile social networks, and mobile wireless networks. She was the co-recipient of the best paper awards of HotPOST 2012 and ACM e-Energy 2016.

Zhiyi Huang is an assistant professor of Computer Science at the University of Hong Kong since 2014. He got his PhD under Sampath Kannan and Aaron Roth from Penn in 2013 and worked as a postdoc with Tim Roughgarden from 2013-2014. His research interest is in the area of theoretical computer science, including in particular algorithmic game theory, differential privacy, and online algorithms.