An Online Emergency Demand Response Mechanism for Cloud Computing

RUITING ZHOU, Wuhan University and University of Calgary
ZONGPENG LI, Wuhan University
CHUAN WU, The University of Hong Kong

This article studies emergency demand response (EDR) mechanisms from a data center perspective, where a cloud participates in a mandatory EDR program while receiving computing job bids from cloud users in an online fashion. We target a realistic EDR mechanism where (i) the cloud provider dynamically packs different types of resources on servers into requested VMs and computes job schedules to meet users' requirements, (ii) the power consumption of servers in the cloud is limited by the grid through the EDR program, and (iii) the operation cost of the cloud is considered in the calculation of social welfare, measured by an electricity cost that consists of both volume charge and peak charge. We propose an online auction for dynamic cloud resource provisioning that is under the control of the EDR program, runs in polynomial time, achieves truthfulness, and close-to-optimal social welfare for the cloud ecosystem. In the design of the online auction, we first propose a new framework, *compact exponential* LPs, to handle job scheduling constraints in the time domain. We then develop a posted pricing auction framework toward the truthful online auction design, which leverages the classic primal-dual technique for approximation algorithm design. We evaluate our online auctions through both theoretical analysis and empirical studies driven by real-world traces.

CCS Concepts: • **Information systems → Data centers**; • **Theory of computation → Packing and covering problems**; **Algorithmic mechanism design**; • **Hardware → Enterprise level and data centers power issues**;

Additional Key Words and Phrases: Cloud computing, demand response, mechanism design, approximation algorithms

ACM Reference format:
Ruiting Zhou, Zongpeng Li, and Chuan Wu. 2018. An Online Emergency Demand Response Mechanism for Cloud Computing. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3, 1, Article 5 (February 2018), 25 pages.
https://doi.org/10.1145/3177755

1 INTRODUCTION

As in the traditional power grid, the quintessential problem in a smart grid is the realtime supply-demand balance, for the stability of the power network. Demand response facilitates the efficiency, reliability, and sustainability of smart grids by reducing and temporally shifting peak loads (Zhou

This project was supported in part by NSFC (61628209, 61571335), by Hubei Science Foundation (2016CFA030, Major Project CXZD2017000121), and by the Research Grants Council of Hong Kong (17204715, 17225516, C7036-15G).
Authors' addresses: R. Zhou, SKLSE, School of Computer Science, Wuhan University, Wuchang, Wuhan, Hubei, China, 430072; Department of Computer Science, University of Calgary, 2500 University Dr NW, Calgary, AB, Canada, T2N 1N4; email: rzho@ucalgary.ca; Z. Li, SKLSE, School of Computer Science, Wuhan University, Wuchang, Wuhan, Hubei, China, 430072; email: zongpeng@whu.edu.cn; C. Wu, Department of Computer Science, The University of Hong Kong, Pok Fu Lam, Hong Kong; email: cwu@cs.hku.hk.

https://doi.org/10.1145/3177755

et al. 2015). Data centers are ideal candidates for participation in such demand response programs, as they represent a substantial fraction of the total power demand witnessed by the grid, part of which naturally exhibit an elastic nature (Wierman et al. 2014). In 2011, data centers consumed approximately 1.5% of all electricity worldwide, and the ratio is predicted to increase to 8% by 2020 (Liu et al. 2014). U.S. data centers consumed an estimated 91 billion kilowatt hours of electricity and incurred 9 billion for electricity bills in 2013 (NRDC 2016). Furthermore, computing jobs in data centers are often elastic and hence can be scheduled flexibly across the temporal domain (Wierman et al. 2014), amenable to demand curtailing and temporal shifting.

A representative scenario for data centers to participate in a demand response program is coordinated consumption reduction dictated by the grid in *emergency demand response* (EDR), when stability of the grid is otherwise jeopardized. When an emergency is imminent (e.g., extreme weather conditions), EDR coordinates the power usage of large electricity users to prevent blackouts. Because of their huge and flexible demand, data centers now serve as a main force in EDR. For example, on July 22, 2011, hundreds of data centers participated in EDR by shifting their workload to reduce the power consumption, preventing a nationwide blackout in the USA and Canada (Misra 2016). Since then, the grid has witnessed a rapid increase in EDR participation. In PJM, a major regional transmission organization in the USA, EDR participation is projected to increase from under 1,700MW in 2006–2007 to close to 15,000MW in 2015–2016, based on existing capacity commitments (PJM 2014). A typical type of EDR program is mandatory EDR (PJM 2014; PJM 2016). Data centers sign a contract ahead with the smart grid, and commit to reduce load or only consume electricity up to a certain level when an EDR signal is dispatched. They receive monetary remuneration when their actual power consumption is below the commitment level, and face a heavy penalty if they fail to do so. EDR helps consumers save billions of dollars each year—for instance, PJM credited $11.8 billion in one year to consumers, the majority of which is produced by EDR (ENERNOC 2016).

We focus on EDR in cloud data centers that run jobs from many users. Such a cloud data center faces a highly non-trivial optimization problem in the event of EDR, in which it strives to satisfy the power consumption reduction dictated by the grid, make judicious online decisions on accepting/declining job bids submitted by cloud users, and compute the most efficient execution schedules for the accepted jobs to minimize operating cost. Operating cost of the cloud comes mainly from electricity cost, which in turn is directly coupled with the processing power of the cloud data center, i.e., how fast the cloud can serve the admitted jobs. A cloud user's job bid specifies (a) the number of each type of virtual machine (VM) required, which can be directly mapped to the amount of each type of cloud resources (e.g., CPU, RAM, disk) required; (b) the length of the job, measured in the number of time slots required for job execution; (c) the preferred deadline for job completion, as well as a penalty function that describes the cost incurred by different degrees of deadline violation; and (d) the amount of monetary remuneration the cloud user is willing to pay.

The goal of this work is to design an online auction for execution at the cloud upon an EDR event, such that (i) the auction runs in an online fashion, making job admission and scheduling decisions immediately upon the arrival of a bid; (ii) the auction mechanism is time efficient and executes in polynomial time; (iii) the auction is truthful, in that it guarantees truthful bidding and constitutes a dominant strategy for each cloud user; and (iv) social welfare, including the net utility of both the cloud and its users, are maximized. By definition, the social welfare depends on which cloud jobs are served as well as the cloud's operating cost for serving them. Such operating cost is comprised of primarily electricity cost, which in turn is composed of two components today: a *volume charge*, which is the product of the total volume of power consumption and a pre-determined per-unit price, as well as a *peak charge*, which is the product of a per-unit peak consumption price and the maximum per-slot volume of consumption. In real-world scenarios, the grid sets the two per-unit

charges such that the volume charge and the peak charge are often comparable, with the latter exceeding the former in many occasions (Zhang et al. 2015a; Hydro 2016).

To focus on the challenges introduced by the EDR program, online job scheduling and the truthful bidding requirement, we first restrict our attention to the case where electricity bills paid by the cloud follows the simple volume charge rule. The extra challenge associated with the peak charge term is handled later in the article. We first formulate the social welfare maximization problem into a natural integer convex program. Such natural formulation consists of both conventional constraints (resource capacity limits) that are well understood and easy to handle in a primal-dual framework, as well as non-conventional constraints (job completion deadlines) whose corresponding dual variables are hard to interpret and update in a primal-dual algorithm. A key contribution of this work is a new technique based on a compact exponential formulation of the convex program, as well as an accompanying dual oracle, which can work in concert with the primal-dual optimization framework for effectively handling non-conventional constraints such as deadline requirements. More specifically, we reformulate the social welfare maximization problem into a compact convex program that consists of traditional capacity-type constraints only, at the cost of introducing an exponential number of variables, each corresponding to a different job schedule in the temporal domain. Correspondingly, the dual program has an exponential number of constraints. Nonetheless, we show a dual oracle that can efficiently identify a polynomial number of dual constraints that need to be considered.

By combining the compact exponential optimization technique with the classic primal-dual method, we are able to adapt the recent posted-pricing auction framework to design an online cloud EDR auction that runs in polynomial time, guarantees truthful bidding, and achieves near-optimal social welfare. In particular, we update the dual variables that correspond to primal capacity constraints carefully in the online auction, to filter out low-price bids and to reserve cloud resources for potential future bids with high bidding prices. Consequently, a good competitive ratio in social welfare can be theoretically proven. Upon the arrival of a job bid, our posted-pricing auction compares its bidding price with the optimal cost for serving this job. Here we design an efficient scheduling algorithm that computes the minimum cost of serving a job with a soft deadline, given static resource prices in different time windows. Whether a job is accepted depends solely on whether its bidding price exceeds its minimum serving cost. As a result, our auction mechanism is both online and truthful.

We further extend our studies of online mechanism design with the peak charge component considered, taking on an extra dimension of challenge corresponding to the online optimization nature of peak electricity charges. We formulate the compact exponential version of the social welfare maximization problem, as well as its dual problem. In the primal-dual framework, we now need to handle the dual variables that correspond to the peak charge constraints in the primal compact exponential program. Based on two different strategies of handling the dual variables for peak charges, we present two versions of the online posted-pricing mechanism: a simple version that handles the dual variables in a peak-oblivious fashion and an improved version that updates the dual variables by taking into account the hitherto peak consumption rate. We analyze and compare these two versions of online mechanisms for peak charges through both theoretical studies and simulations driven by real-world traces.

The rest of the article is structured as follows. In Section 2, we review related work in demand response and cloud market mechanisms. Section 3 outlines the problem model and assumptions. Section 4 presents and analyzes an online cloud EDR auction based on a volume charge model for electricity cost; Section 5 further generalizes the studies to the more general and realistic case where electricity charges consist of both a volume charge term and a peak charge term. Section 6 presents performance evaluation, and Section 7 concludes the article.

2 RELATED WORK

This work investigates efficient mechanisms for data centers to admit and schedule cloud jobs while participating in a mandatory EDR program. We first review some related works about auction design in clouding computing. The earliest cloud auctions do not consider the dynamic provisioning of VMs based on user demand, and assume that there are a signal type of VMs or the number and types of VMs are fixed before the auction starts (Wang et al. 2012; Zaman and Grosu 2013). Subsequently, auction design for dynamic VM provisioning, in which the cloud provider assembles VMs based on demand expressed in user bids, starts to appear in the past two years. Zhang et al. (2014) formulate the dynamic resource provisioning problem in the cloud into a combinatorial optimization problem with a packing nature, and propose a truthful randomized auction to solve it. Their approach is based on an LP decomposition technique, computing an α-approximate solution in polynomial time, with $\alpha \sim 2.72$ in typical scenarios. Zhang et al. (2015c) employ smoothed analysis and randomized reduction techniques to design a randomized cloud resource auction. Their randomized mechanism achieves truthfulness, polynomial running time, and $(1 - \epsilon)$-optimal social welfare, all in expectation. These mechanisms focus on a one-round auction, without considering the more realistic scenario where bids from could users arrive online.

Recently, a series of studies focused on the online auction design for dynamic VM allocation in cloud computing. Shi et al. present the first online combinatorial auction for cloud computing (Shi et al. 2014). They assume that cloud users are subject to budget constraints and bids arrive at the beginning of each round of the auction. Their online framework first decomposes the long-term optimization problem into a series of one-round auctions, and then applies the primal-dual technique to design a truthful one-round auction. Zhang et al. (2015) consider a more practical scenario where cloud users' bids arrive randomly over a long time span. They further include server cost when computing the social welfare and the cloud provider's profit. The end result is a truthful online auction that approximately maximizes the social welfare and provider's profit in polynomial running time. Compared with existing studies, a cloud user in our EDR auction bids for a schedule rather than a fixed time window for its job execution. This work is among the first in online EDR auction design for dynamic cloud resource provisioning that presents the scheduling dimension in solution spaces.

Well-designed auction mechanisms are also a natural candidate for incentivizing demand response participation. Zhou et al. (2015) present an efficient randomized approach for carrying out demand response between the power grid and large electricity users, microgrids, and electricity storage devices. Samadi et al. propose a VCG mechanism that aims to maximize the social welfare of a smart grid. Their design requires electricity users to report their energy demand, and computes the electricity payment based on the demand (Samadi et al. 2012). The majority of the demand response literature studies demand response from the smart grid's perspective, and is not dedicated to the design of data center demand response.

Along the direction of data center demand response, Wierman et al. survey the opportunities and challenges for data centers to participate in EDR (Wierman et al. 2014). Zhang et al. (2015b) study EDR in multi-tenant colocation data centers, where each tenant manages its own servers and participates in the EDR by reducing its power consumption. They propose a truthful reverse auction to provide monetary remuneration to tenants, consummate to their energy reduction. Zhou et al. (2015) consider the electricity trade between smart grids and green data centers in the demand response service. They tailor a pricing scheme for geo-distributed green data centers to minimize the energy cost. Zhang et al. (2015a) propose online electricity cost saving algorithms for colocation data centers, when the electricity charge contains both *volume charge* and *peak charge*. They consider two approaches to incentivize tenants to shed energy consumption: a pricing approach and an auction approach. Different from the above studies, this work is from the data

center's perspective and focuses on the admission and scheduling of the cloud user's jobs to satisfy the power consumption constraint in EDR, while striving to maximize social welfare of the cloud ecosystem. We consider the realistic scenario where electricity cost comprises both volume charge and peak charge, making the auction design more challenging.

3 SYSTEM MODEL

We consider a typical type of demand response program, mandatory EDR (PJM 2014; PJM 2016). The data center signs a contract with the smart grid *a priori* (e.g., one year ahead with PJM (2014)) and receives financial rebates for its committed load reduction, whereas failure to cut load as required during EDR incurs a heavy penalty. In the event of EDR, the grid sends a signal to the data center at the beginning of the auction, specifying the amount of energy reduction that the data center needs to reduce in each time slot. Based on that, the data center then calculates the amount of available power in each slot, $E_t, \forall t \in [T]$, to schedule its job execution.

The cloud data center hosts S servers and offers K types of resources, as exemplified by CPU, RAM, and disk storage, which can be dynamically assembled into different types of VMs. Let $[X]$ denote the integer set $\{1, 2, \ldots, X\}$. We assume the amount of type-k resource available in server $s \in [S]$ is c_{ks} units. The cloud service provider acts as the auctioneer to lease VMs to cloud users through an online auction.

There are I cloud users, each submitting one bid for executing its job, during a large time span $1, 2, \ldots, T$. User bids arrive randomly, each requesting a bundle of tailor-made VMs for job execution, mapping to a required amount of each type of resource. We consider batch jobs such as big data analytics and Google crawling data processing. They don't request always-on VM services, and may tolerate a certain level of delay in the job completion. Let B_i denote user i's bid. It submits at time t_i and contains (i) r_i^k, the amount of type-k resource required to configure the tailor-made VMs; (ii) w_i, the number of time slots (not necessarily consecutive) needed to complete the job by the tailor-made VMs; (iii) d_i, the desired deadline for job completion; and (iv) $g_i(\tau_i)$, a penalty function defined over deadline violation, τ_i:

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

where $d_i + \tau_i$ is the job completion time. Let b_i denote user i's bidding price if its job is completed before the deadline d_i, and then $b_i - g_i(\tau_i)$ is the corresponding bidding price with completion time $d_i + \tau_i$. $g_{c_i}(\tau_i)$ is a nondecreasing function with $g_{c_i}(0) = 0$. User i's bidding language can be expressed as follows: $B_i = \{t_i, \{r_i^k\}_{k\in[K]}, w_i, d_i, b_i, g_i(\tau_i)\}$.

Upon the arrival of each bid, the cloud provider immediately computes the resource allocation and announces the auction results: (i) $x_{is} \in \{0, 1\}$, where $x_{is} = 1$ if user i's job is accepted and allocated on server s, and 0 otherwise; (ii) $y_{is}(t) \in \{0, 1\}$ encodes the scheduling of user i's job, where $y_{is}(t) = 1$ if user i's job is scheduled to run on server s at time t, and 0 otherwise; (iii) p_i, user i's payment. Let v_i and $g_i'(\tau_i)$ be user i's true valuation if its job is completed before d_i and true penalty function, and then $v_i - g_i'(\tau_i)$ is the true valuation of user i's bid. User i's utility with bidding price $b_i - g_i(\tau_i)$ is $u_i(b_i - g_i(\tau_i)) = \sum_{s\in[S]} v_i x_{is} - g_i'(\tau_i) - p_i$. Each user is assumed to be selfish and rational, with a natural aim to maximize its own utility. They may choose to lie about the true valuation if doing so leads to a higher utility. In our online auction design, social "happiness" is the target of optimization; towards this goal, it is important to elect truthful bids.

Definition 3.1. (Truthful Auction): A cloud auction is *truthful* if bidding true valuation is a dominant strategy for a cloud user, always maximizing the user utility: for all $b_i - g_i(\tau_i) \neq v_i - g_i'(\tau_i)$, $u_i(v_i - g_i'(\tau_i)) \geq u_i(b_i - g_i(\tau_i))$.

Table 1. Summary of Notations

I	# of users	$[X]$	integer set $\{1, \ldots, X\}$
T	# of time slots	S	# of servers
K	# of resource types	t_i	user i's arrival time
h_t	electricity price at t	h_{peak}	peak electricity price
g	penalty function	p_i	user i's payment
r_i^k	demand of type-k resource by user i		
w_i	# slots requested by user i		
τ_i	# slots that pass the deadline for user i		
d_i	deadline of user i's job		
b_i	bidding price of user i's job if completed before d_i		
v_i	true valuation of user i's job if completed before d_i		
x_{is}	serve bid i on server s (1) or not (0)		
$y_{is}(t)$	whether to allocate user i's job on server s at t		
c_{ks}	capacity of type-k resource on server s		
$e(t)$	amount of power consumption at slot t		
E_t	amount of available power at slot t, from EDR		

It is natural to consider the operating cost when we aim to maximize the social welfare. The operating cost mainly comprises power consumption of the servers, increasing with the increment of the resource occupied on the server. The power consumption of a server can typically be modelled as $\sum_{k\in[K]} \beta_k u_k$ (Tian and Zhao 2014), where u_k is the utilization of type-k resource, and β_k represents the power consumption when type-k resource is in full usage. β_k usually takes the value within $[20, 60]$ for CPU and $[0.2, 2]$ for RAM (Tian and Zhao 2014; Kansal et al. 2010). The operating cost equals the electricity charge paid by the data center to the utility company. In this article, we consider two charge models: a basic model with volume charges only, and a more realistic model with both volume charges and peak charges (Hydro 2016; Zhang et al. 2015a).

3.1 The Basic Volume Charge Model

Let $e(t)$ be the actual power consumption in slot t, the operating cost of the data center at t can be defined as

$$f_t(e(t)) = \begin{cases} h_t e(t), & \text{if } e(t) \le E_t \\ +\infty, & \text{otherwise} \end{cases}, \tag{2}$$

where h_t is electricity price at time t, known by the data center before the auction starts.

Definition 3.2. (Social Welfare): The social welfare is the aggregate of users' utilities $\sum_{i\in[I]}(\sum_{s\in[S]} v_i x_{is} - g_i'(\tau_i) - p_i)$ plus cloud provider's utility $\sum_{i\in[I]} p_i - \sum_{t\in[T]} f_t(e(t))$. Since payments cancel themselves, the social welfare becomes $\sum_{i\in[I]} (\sum_{s\in[S]} v_i x_{is} - g_i'(\tau_i)) - \sum_{t\in[T]} f_t(e(t))$.

3.2 The Volume and Peak Charge Model

Let h_{peak} be the peak electricity cost, occurring in the slot with the maximum power consumption. The utility of the cloud provider equals the aggregate of users' payments minus the operating cost. i.e., $\sum_{i\in[I]} p_i - \sum_{t\in[T]} f_t(e(t)) - h_{peak} \max_t e(t)$. The social welfare in this model is $\sum_{i\in[I]}(\sum_{s\in[S]} v_i x_{is} - g_i'(\tau_i)) - \sum_{t\in[T]} f_t(e(t)) - h_{peak} \max_t e(t)$. We summarize important notations in Table 1 for easy reference.

4 ONLINE AUCTION DESIGN UNDER THE VOLUME CHARGE MODEL

We start with a basic electricity cost model, in which the data center pays electricity charges according to a fixed unit cost and its total volume of consumption (the *volume charge model*). We first formulate the social welfare maximization problem and introduce the new framework to handle job deadline constraints in Section 4.1. We then design an online auction in Section 4.2 and present the theoretical analysis in Section 4.3.

4.1 Social Welfare Maximization Problem

Under the assumption of truthful bidding, the social welfare maximization problem can be formulated into the following convex program:

$$\text{maximize} \sum_{i\in[I]}\left(\sum_{s\in[S]} b_i x_{is} - g_i(\tau_i)\right) - \sum_{t\in[T]} f_t(e(t)), \tag{3}$$

subject to

$$\sum_{s\in[S]} x_{is} \le 1, \forall i \in [I], \tag{3a}$$

$$y_{is}(t)t \le d_i + \tau_i, \forall t \in [T], \forall s \in [S], \forall i \in [I] : t_i \le t, \tag{3b}$$

$$w_i x_{is} \le \sum_{t\in[T]:t_i\le t} y_{is}(t), \forall i \in [I], \forall s \in [S], \tag{3c}$$

$$\sum_{i\in[I]:t_i\le t} r_i^k y_{is}(t) \le c_{ks}, \forall k \in [K[, \forall s \in [S], \forall t \in [T], \tag{3d}$$

$$\sum_{s\in[S]}\sum_{k\in[K]} \beta_{ks}\left(\frac{\sum_{i\in[I]:t_i\le t} r_i^k y_{is}(t)}{c_{ks}}\right) \le e(t), \forall t \in [T], \tag{3e}$$

$$\tau_i, e(t) \ge 0, x_{is}, y_{is}(t) \in \{0,1\}, \forall i \in [I], \forall s \in [S], \forall t \in [T]. \tag{3f}$$

Note that the following constraint is redundant, and is not explicitly included in the above convex problem: $y_{is}(t) \le x_{is}, \forall i \in [I], \forall s \in [S], \forall t \in [T]$. Constraint Equation (3a) indicates that each user's job is executed on at most one server. Constraint Equation (3b) ensures that a job is scheduled to run only after its arrival time. Constraint Equation (3c) guarantees that the number of time slots allocated to an accepted bid is sufficient for completing the job. The capacity limit of each type of resource is modelled in constraint Equation (3d), and constraint Equation (3e) records the total power consumption in each time slot into $e(t)$. Setting $f_t(e(t)) = +\infty$ when $e(t) > E_t$ ensures that the actual power consumption is capped at the EDR-specified amount.

If we let $g_i(\tau_i) = f_t(e(t)) = 0$, even in the offline setting, problem Equation (3) without constraint Equations (3b), (3b), (3c), and (3e) is still an NP-hard combinatorial optimization problem, equivalent to the classic knapsack problem. The challenge further escalates when we involve the jobs' soft deadlines and operating cost. We shall resort to the primal-dual algorithm design technique to address some of these challenges. However, the technique cannot be directly applied to Equation (3) since it involves unconventional constraints for modelling job deadlines. We first propose a new framework to handle these unconventional constraints. More specifically, we reformulate the

original problem Equation (3) into a *compact exponential* convex problem with a compact packing structure, at the price of involving an exponential number of decision variables:

$$\text{maximize} \sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il}x_{il} - \sum_{t\in[T]} f_t(e(t)), \tag{4}$$

subject to

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

$$\sum_{i\in[I]}\sum_{l:t\in T(l),s\in S(l)} r_i^k x_{il} \le c_{ks}, \forall k \in [K], \forall s \in [S], \forall t \in [T], \tag{4b}$$

$$\sum_{s\in[S]}\sum_{k\in[K]} \beta_{ks}\Big(\frac{\sum_{i\in[I]}\sum_{l:t\in T(l),s\in S(l)} r_i^k x_{il}}{c_{ks}}\Big) \le e(t), \forall t \in [T], \tag{4c}$$

$$e(t) \le 0, x_{il} \in \{0,1\}, \forall t \in [T], \forall i \in [I], \forall l \in \zeta_i. \tag{4d}$$

Here ζ_i is the set of feasible time schedules for user i's job. A feasible time schedule is the vector $l = (\{x_{is}\}_{\forall s\in[S]}, \{y_{is}(t)\}_{\forall s\in[S],t\in[T]}, \tau_i)$ that satisfies constraint Equations (3a), (3b), and (3c). x_{il} is the binary decision variable where $x_{il} = 1$ if user i's job is accepted and executed according to schedule $l \in \zeta_i$, and 0 otherwise. b_{il} is the value based on schedule l, which equals $b_i - g_i(\tau_i)$ where τ_i is the duration of deadline violation according to l. $T(l)$ and $S(l)$ represent the set of time slots and the server when and where user i's job is executed in schedule l, respectively. Constraint Equations (4b) and (4c) are equivalent to Equations (3d) and (3e). Constraint Equation (4a) guarantees that a job can only be accepted according to one schedule. A feasible solution to Equation (3) corresponds to a feasible solution in Equation (4) and vice versa, and hence the optimal objective values of both problems are equal.

We relax $x_{il} \in \{0,1\}$ to $x_{il} \ge 0$ and introduce dual variables u_i,$p_{ks}(t)$ and $m(t)$ to Equations (4a), (4b), and (4c). The Fenchel dual (Boyd and Vandenberghe 2004; Devanur et al. 2016) of the relaxed problem Equation (4) is

$$\text{minimize} \sum_{i\in[i]} u_i + \sum_{k\in[K]}\sum_{s\in[S]}\sum_{t\in[T]} c_{ks}p_{ks}(t) + \sum_{t\in[T]} f_t^*(m(t)), \tag{5}$$

subject to

$$u_i \ge b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)}\sum_{s\in S(l)} r_i^k\left(p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}}\right), \forall i \in [I], \forall l \in \zeta_i, \tag{5a}$$

$$p_{ks}(t), u_i, m(t) \ge 0, \forall i \in [I], \forall k \in [K], \forall s \in [S], \forall t \in [T], \tag{5b}$$

where $f_t^*(m(t))$ is the convex conjugate (Bauschke and Lucet 2012) of the cost function $f_t(\cdot)$, defined as

$$f_t^*(m(t)) = \sup_{e(t)\ge 0}\{m(t)e(t) - f_t(e(t))\}.$$

Proposition 4.1. *The explicit expression of* $f_t^*(m(t))$ *is*

$$f_t^*(m(t)) = \begin{cases} 0, & m(t) \le h_t \\ (m(t) - h_t)E_t, & m(t) > h_t \end{cases}. \tag{6}$$

Proof. By the definition of $f_t^*(m(t))$, $f_t^*(m(t)) = (m(t) - h_t)e(t)$ if $e(t) \le E_t$ and $f_t^*(m(t)) = -\infty$ otherwise. Thus, we only need to consider the case when $e(t) \le E_t$. If $m(t) - h_t \ge 0$,

$(m(t) - h_t)e(t)$ is maximized when $e(t) = E_t$ with the maximum value $(m(t) - h_t)E_t$; if $m(t) - h_t \leq 0$, the maximum value of $(m(t) - h_t)e(t)$ is 0 when $e(t) = 0$. □

4.2 Online Auction Design

A key problem in the online auction design is to decide whether to accept user i's job and how to schedule its job to maximize its utility, while the power consumption in each slot is limited by the EDR program. If the cloud provider accepts user i's job on server s with schedule l, then $x_{is} = 1$, τ_i is assigned according to the completion time, $y_{is}(t)$ is updated and $e(t)$ is increased for slots in $T(l)$. To solve the original convex problem Equation (3), we seek the help of the compact exponential convex problem Equation (4) and its dual Equation (5). We observe that for each primal variable x_{il}, there is a dual constraint Equation (5a) associated to it. Complementary slackness in the primal-dual technique indicates that x_{il} is updated based on its dual constraint. x_{il} remains zero unless its associated dual constraint becomes tight. Because dual variable u_i is nonnegative, we let u_i be the maximum of 0 and the right-hand side (RHS) of constraint Equation (5a), that is,

$$u_i = \max\left(0, \max_{l \in \zeta_i}\left\{b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)}\sum_{s\in S(l)} r_i^k\left(p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}}\right)\right\}\right). \tag{7}$$

Accordingly, the winner is determined based on u_i: the cloud provider accepts user i's job if $u_i > 0$, and serves it according to the schedule that maximizes the RHS of Equation (5a). The cloud provider rejects user i's job if $u_i \leq 0$.

The rational can be explained as follows. If we interpret $p_{ks}(t)$ as the unit capital price of type-k resource on server s at time t and $m(t)$ as the unit electricity price at time t, then $\sum_{k\in[K]}\sum_{t\in T(l)}\sum_{s\in S(l)} r_i^k(p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}})$ is the total cost of user i's job if it is accepted and scheduled by l. Furthermore, the RHS of Equation (5a) is user i's utility with schedule l. If we interpret u_i as user i's utility, the assignment of u_i in Equation (7) guarantees that user i's job is always served with the schedule that effectively maximizes its utility based on the current price, which leads to social welfare maximization and truthfulness.

Although there are an exponential number of dual constraints involved in the computation of u_i, most of them can be filtered by a dual oracle based on dynamic programming. This is realized through the selection of schedules. We fix a polynomial number of schedules by the dual oracle (lines 1–14 in Algorithm 2), and let u_i be the maximum of zero and the RHS of Equation (5a) with these schedules. For each server $s \in [S]$, we construct a set of best schedules. We fix the job completion time to be t_c ($t_c \in [t_i + w_i - 1, T)$, then the best schedule is the one with the minimum price among all schedules of the same completion time. The output of the dual oracle is such S sets of best schedules. The construction of the best schedules can be accomplished through dynamic programming method. The base case is the schedule l_0 with slots in $[t_i, t_i + w_i - 1]$. We push the completion time one slot forward each time. We calculate the price $c(t)$ for user i's job running at time t, i.e., $c(t) = \sum_{k\in[K]} r_i^k(p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}})$. If the completion time passes the deadline d_i, the price will be increased by the corresponding penalty, i.e., $c(t) = \sum_{k\in[K]} r_i^k(p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}}) + g_i(t - d_i)$. In the process of replacing the completion time, we only need to compare the price of the old competition time and $w_i - 1$ slots preceding the old competition time. We next use a simple example to illustrate how the algorithm works. If user i arrives at time 3 with $w_i = 4$, then the basic schedule is $l_0 = \{3, 4, 5, 6\}$. The next step is to construct the best schedule with completion time 7. Assume that $\arg\max_{t\in\{3,4,5\}} c(t) = 3$, the best schedule is $\{6, 4, 5, 7\}$ if $c(6) < c(3)$ and $\{3, 4, 5, 7\}$ otherwise. This process is repeated until the completion time reaches T.

We next discuss the design of the two prices: unit capital price $p_{ks}(t)$ and unit electricity price $m(t)$. Recall that h_t is the unit electricity price at time t charged by the power grid; thus we let $m(t) = h_t$ based on the interpretation of dual variable $m(t)$. For the design of $p_{ks}(t)$, we introduce a new variable $z_{ks}(t)$, representing the amount of allocated type-k resource on server s at time t. Let U_k and L_k be the maximum and minimum values per unit of type-k resource per unit of time, respectively. U_k and L_k represent how users evaluate a unit of type-k resource, considering both the capital cost and electricity cost. Hence, we assume that $L_k > h_t, \forall t \in [T]$, without loss of generality. We propose a price function such that the total unit price $p_{ks}(t) + m(t)$ equals L_k at the beginning and reaches U_k ultimately. Because $m(t) = h(t)$, we let $p_{ks}(t)$ start at $L_k - h_t$ and exponentially increase with the growth of the current usage $z_{ks}(t)$. $p_{ks}(t)$ equals $U_k - h_t$ when $z_{ks}(t)$ exceeds its capacity c_{ks}. In this case, the cloud provider will not accept any more jobs. To sum up, $p_{ks}(t)$ and $m(t)$ are defined as follows:

$$p_{ks}(t)(z_{ks}(t)) = (L_k - h_t)\left(\frac{U_k - h_t}{L_k - h_t}\right)^{\frac{z_{ks}(t)}{c_{ks}}}, \tag{8}$$

$$m(t) = h_t, \forall t \in [T], \tag{9}$$

where $U_k = \max_{i\in[I]:r_i^k>0}\{\frac{b_i}{r_i^k}\}$ and $L_k = \min_{i\in[I]:r_i^k>0}\{\frac{b_i - g_i(T-d_i)}{w_i \sum_{k\in[K]} r_i^k}\}$ with truthful bidding.

ALGORITHM 1: A Primal-Dual Online Auction $A_{online1}$

Input: bidding language $\{B_i\}, \{c_{ks}\}, \{\beta_k\}, \{E_t\}, \{h_t\}$

1: Define cost function $f_t(e(t))$ according to Equation (2);
2: Define function $p_{ks}(z_{ks}(t))$ according to Equation (8);
3: Initialize $x_{is} = 0, y_{is}(t) = 0, z_{ks}(t) = 0, \tau_i = 0, x_{il} = 0, u_i = 0, p_{ks}(t) = 0, m(t) = h_t, e(t) = 0, \forall i \in [I], l \in \zeta_i, k \in [K], s \in [S], t \in [T]$;
4: **Upon the arrival of the ith user**
5: $\left(x_{is}, \{y_{is}(t)\}, p_i, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{e(t)\}\right) = A_{core}\left(B_i, \{c_{ks}\}, \{E_t\}, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{m(t)\}, \{e(t)\}\right)$;
6: **if** $\exists s \in [S], x_{is} = 1$ **then**
7: Accept user i's bid and allocate resources to server s according to $y_{is}(t)$; Charge p_i from user i;
8: **else**
9: Reject user i.
10: **end if**

The online auction $A_{online1}$ is shown in Algorithm 1 with scheduling algorithm A_{core1} in Algorithm 2 running for each user. $A_{online1}$ first defines the cost function and price function in lines 1 and 2. Line 3 initializes all primal and dual variables. Upon the arrival of the ith user, the scheduling algorithm A_{core1} selects the best schedule $\hat{l}$ that maximizes user i's utility through the dual oracle (lines 1–15). If user i can obtain positive utility, primal variables x_{is}, y_{is}, and x_{il} are updated accordingly (line 17). Then line 18 calculates the utility and the payment. Line 19 increases the usage of K resources on the specified server and records the current power consumption level. Finally, unit resource price is updated in line 20.

4.3 Theoretical Analysis

(i) Correctness, Polynomial Running Time, and Truthfulness.

We first analyze the running time of $A_{online1}$ and prove its correctness in Theorem 4.2, and then present the proof of its truthfulness in Theorem 4.3.

ALGORITHM 2: A Scheduling Algorithm A_{core1}.

Input: $B_i, \{c_{ks}\}, \{E_t\}, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{m(t)\}, \{e(t)\}$
Output: $x_{is}, \{y_{is}(t)\}, p_i, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{e(t)\}$

1: **for all** $s \in [S]$ **do**
2: Add slot $t \in [t_i, T]$ to set $\mathcal{T}$ if $z_{ks}(t) + r_i^k \le c_{ks}, \forall k \in [K]$ and $\sum_{k\in[K]} \beta_{ks} r_i^k / c_{ks} + e(t) \le E_t$;
3: Let schedule l_0 include the first w_i slots $(t_1, t_2, \ldots, t_{w_i})$ in $\mathcal{T}$; Define $j = 1$;
4: **while** $w_i + j \le |\mathcal{T}|$ **do**
5: $l_j = l_{j-1}$;
6: Let t_c is the $(w_i + j)$th slot in $\mathcal{T}$;
7: $c(t) = \sum_{k\in[K]} r_i^k (p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}}), \forall t \in \{t_1, t_2, \ldots, t_{w_i}, t_c\}$;
8: If $t_c > d_i$, $c(t_c) = c(t_c) + g_i(t_c - d_i)$;
9: $t_m = \arg\max_{t\in\{t_1,\ldots,t_{w_i-1}\}} c(t)$;
10: If $c(t_{w_i}) < c(t_m)$, for schedule l_j, replace the slot t_m with t_{w_i}; Save t_c into t_{w_i};
11: $\mathcal{P}_j = \sum_{t\in T(l_j)} c(t)$; $j = j + 1$;
12: **end while**
13: $s^* = \arg\min_j \{\mathcal{P}_j\}$; $\mathcal{P}_s^* = \mathcal{P}_{s^*}$; $l_s^* = l_{s^*}$;
14: **end for**
15: $\hat{s} = \arg\min_s \{\mathcal{P}_s^*\}$; $\hat{\mathcal{P}} = \mathcal{P}_{\hat{s}}^*$, $\hat{l} = l_{\hat{s}}^*$;
16: **if** $b_i - \hat{\mathcal{P}} > 0$ **then**
17: $x_{i\hat{s}} = 1$; $y_{i\hat{s}}(t) = 1, \forall t \in T(\hat{l})$; $x_{i\hat{l}} = 1$;
18: $u_i = b_i - \hat{\mathcal{P}}$; $p_i = \sum_{k\in[K]} \sum_{t\in T(\hat{l})} r_i^k (p_{k\hat{s}}(t) + m(t)\frac{\beta_{k\hat{s}}}{c_{k\hat{s}}})$;
19: $z_{k\hat{s}}(t) = z_{k\hat{s}}(t) + r_i^k, \forall k \in [K], t \in T(\hat{l})$; $e(t) = e(t) + \sum_{k\in[K]} r_i^k \beta_{k\hat{s}} / c_{k\hat{s}}, \forall t \in T(\hat{l})$;
20: $p_{k\hat{s}}(t) = p_{k\hat{s}}(z_{k\hat{s}}(t)), \forall k \in [K], t \in T(\hat{l})$;
21: **end if**
22: **Return** $x_{is}, \{y_{is}(t)\}, p_i, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{e(t)\}$

THEOREM 4.2. *Algorithm $A_{online1}$ terminates in polynomial time, and returns a feasible solution for problem Equations (3), (4), and (5).*

PROOF. *Correctness*: $A_{online1}$ generates a feasible solution for problem Equation (4) because constraint Equation (4b) is satisfied by the `if` condition at line 2 in Algorithm A_{core1}. Line 19 in A_{core1} guarantees that the LHS of constraint Equation (4c) equals its RHS. Line 17 updates the value of $x_{i\hat{l}}$ to 1 for one particular schedule $\hat{l}$, satisfying constraint Equations (4a) and (4d). Furthermore, the corresponding relation between convex problems Equations (3) and (4) implies that the solution returned by $A_{online1}$ will never violate the constraints in Equation (3). For the dual problem Equation (5), if the maximum value of the RHS of constraint Equation (5a) is nonnegative, u_i equals the maximum value, and remains 0 otherwise. Therefore, $A_{online1}$ ensures feasibility of the dual problem Equation (5).

Polynomial Running Time: We first examine the running time of A_{core1}. During each iteration of the `for` loop, line 2 takes $O(KT)$ steps to initialize the feasible slot set $\mathcal{T}$. Line 2 defines a schedule l_0 in w_i steps. The `while` loop iterates almost $T - w_i$ rounds to compute the best schedule with the fixed completion time. Inside the `while` loop, lines 5–8 take $O(w_i + 1)$ steps to update the value of $c(t)$. The slot with the largest price can be founded in $O(w_i - 1)$ steps in line 9. The comparison and addition in lines 10 and 11 can be executed in constant time. Hence, the running time of the `while` loop is $O((T - w_i)w_i)$. The running time of line 13 is linear to $T - w_i$. Then the execution time of the `for` loop is $O(S(T - w_i)w_i)$. The `if` statement (lines 16–21) updates the primal and dual variables in $O(Kw_i)$ steps. In summary, the running time of A_{core1} is $O(KST^2)$.

We then can investigate the running time of $A_{online1}$. The definition of the cost and price function in lines 1 and 2 takes constant time. Line 3 initializes all the primal and dual variables in linear time. Upon the arrival of the ith user, A_{core1} makes the decision and computes the schedule in $O(KST^2)$ steps. Lines 6–10 process user i's request in constant time. Thus, the overall running time of $A_{online1}$ is $O(IKST^2)$. □

THEOREM 4.3. *The online auction $A_{online1}$ is a truthful auction.*

PROOF. Our auction $A_{online1}$ falls into the family of *posted pricing mechanism* (Huang and Kim 2015), where the winner determination process and the payment calculation depend only on the current prices of resources. The price that a winning user i pays for its job execution depends on the amount of resources allocated, and user i's demand. It is independent of user i's bidding price. Consequently, a user cannot improve its utility by lying about its bidding price as the utility is the difference between its valuation and the price. In addition, our algorithm A_{core1} always computes the best schedule for user i to maximize its utility given the current prices. Therefore, our online auction $A_{online1}$ is a truthful auction that guarantees the maximum utility is achieved by truthful bidding. □

(ii) Competitive Ratio.

We proceed to analyze the competitiveness of $A_{online1}$ in social welfare, measured by the competitive ratio. The *competitive ratio* is the upper-bound ratio of the social welfare achieved by the optimal solution of convex problem Equation (3) to the social welfare achieved by $A_{online1}$. We first introduce a primal-dual analysis framework in Lemma 4.4, which states that if there exists a bound between the increase of the primal objective value and the increase of the dual objective value, then the competitive ratio is also bounded. We next define an *Allocation-Price Relationship* for $A_{online1}$ and the differential version of it in Definition 4.5 and Definition 4.7, respectively. We prove that if the Allocation-Price Relationship holds for a given α_1, $A_{online1}$ satisfies the inequality in Lemma 4.4. We then present the value of α_1 in Lemma 4.8 and prove that $A_{online1}$ is α_1-competitive in Theorem 4.9.

Let OPT_1 and OPT_2 be the objective value of convex problem Equations (3) and (4), respectively. It is clear that $OPT_1 = OPT_2$. Let P_i and D_i denote the objective value of primal problem Equation (4) and that of dual problem Equation (5) returned by an algorithm after handling user i's bid. Let $P_0 = D_0 = 0$ be the initial values. Then P_I and D_I are the final primal and dual objective values at the end of T.

LEMMA 4.4. *If there exists a constant $\alpha \geq 1$ such that $P_i - P_{i-1} \geq \frac{1}{\alpha}(D_i - D_{i-1})$ for every i, then the algorithm is α-competitive in social welfare.*

PROOF. If we sum up the inequality for each i, we can obtain

$$P_I = \sum_i (P_i - P_{i-1}) \geq \frac{1}{\alpha} \sum_i (D_i - D_{i-1}) = \frac{1}{\alpha} D_I.$$

The above inequality holds because $P_0 = D_0 = 0$. By weak duality (Boyd and Vandenberghe 2004), $D_I \geq OPT_2$, therefore $P_I \geq \frac{1}{\alpha} OPT_2 = \frac{1}{\alpha} OPT_1$. So we can conclude that the algorithm is α-competitive in social welfare. □

Let $p^i_{ks}(t)$ denote the price of type-k resource in server s after processing user i's bid. $z^i_{ks}(t)$ is the amount of allocated type-k resource in server s after handling user i's job. Note that $A_{online1}$ guarantees $P_0 = D_0 = 0$.

Definition 4.5. The Allocation-Price Relationship for $A_{online1}$ with $\alpha_1 \geq 1$ is

$$p_{ks}^{i-1}(t)(z_{ks}^{i}(t) - z_{ks}^{i-1}(t)) \geq \frac{1}{\alpha_1} c_{ks}(p_{ks}^{i}(t) - p_{ks}^{i-1}(t)),$$
$$\forall i \in [I], \forall k \in [K], \forall t \in T(l), \forall s \in S(l).$$

Lemma 4.6. *If 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's job is rejected by the cloud provider, then $P_i - P_{i-1} = D_i - D_{i-1} = 0$. In the following analysis, we assume that user i's job is accepted, and let l be the schedule of it. The increment of the primal objective function after handling user i's job is

$$\begin{aligned} P_i - P_{i-1} &= b_{il} - \sum_{t \in T(l)} \left(f_t(e^i(t)) - f_t(e^{i-1}(t)) \right) \\ &= u_i + \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} r_i^k \left(p_{ks}^{i-1}(t) + m^{i-1}(t) \frac{\beta_{ks}}{c_{ks}} \right) \\ &\quad - \sum_{t \in T(l)} \left(f_t(e^i(t)) - f_t(e^{i-1}(t)) \right). \end{aligned}$$

The second equation holds because when user i's job is accepted by the server s with schedule l, the left hand side of constraint Equation (5a) equals the RHS of it. Since $f_t(e(t)) = h_t e(t)$, $\sum_{t \in T(l)} (f_t(e^i(t)) - f_t(e^{i-1}(t))) = \sum_{t \in T(l)} h_t (\sum_{k \in [K]} \sum_{s \in S(l)} r_i^k \beta_{ks}/c_{ks})$. Also, we know that $m^{i-1}(t) = h_t$, hence,

$$P_i - P_{i-1} = u_i + \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} r_i^k p_{ks}^{i-1}(t).$$

According to the expression of $f_t^*(m(t))$ in Equation (6) and the definition of $m(t)$ in Equation (9), $f_t^*(m(t)) = 0$, thus,

$$D = \sum_{i \in [i]} u_i + \sum_{k \in [K]} \sum_{s \in [S]} \sum_{t \in [T]} c_{ks} p_{ks}(t).$$

The increase of the dual objective value is

$$D_i - D_{i-1} = u_i + \sum_{t \in T(l)} \sum_{s \in S(l)} \sum_{k \in [K]} c_{ks} \left(p_{ks}^{i}(t) - p_{ks}^{i-1}(t) \right).$$

Note that $z_{ks}^{i}(t) - z_{ks}^{i-1}(t) = r_i^k$. By summing up the Allocation-Price Relationship over all $s \in S(l)$, $k \in [K]$ and $t \in T(l)$, 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 $\alpha \geq 1$, it is obvious that $P_i - P_{i-1} \geq \frac{1}{\alpha_1}(D_i - D_{i-1})$. □

We observe that each inequality in the Allocation-Price Relationship involves variables only for type-k resource in server s. Next, we are trying to identify the corresponding $\alpha_{1,ks}$ for each pair of k and s that satisfies the Allocation-Price Relationship. Then α_1 is just the maximum value among all $\alpha_{1,ks}$. To compute the value of $\alpha_{1,ks}$, we make the following mild assumption and define the differential version of the Allocation-Price Relationship based on it:

Assumption 1. The job demand is much smaller than the sever's capacity, i.e., $r_i^k \ll c_{ks}$.

In the real world, a job's demand is usually smaller than a server's capacity in a large data center. We make this assumption mainly to facilitate our theoretical analysis, such that techniques from calculus (differentiation) can be used. We don't consider extreme cases, which are rare in

practice. For example, if a high-valued bid demanding almost all the resource is rejected, because a small fraction of the resource is used by other users, then the worst-case competitive ratio can be infinitely large. It is also worth noting that similar assumptions are made customarily in relevant literature of online resource allocation (Zhang et al. 2015; Zhou et al. 2017; Agrawal et al. 2014; Jaillet and Lu 2012). In addition, we can relax Assumption 1 and assume an upper bound on $\frac{r_i^k}{c_{ks}}$. Instead of differential equation and integration, we can use difference equation and summation to derive similar results. We also relax this assumption completely in our simulation studies. Under Assumption 1, $z_{ks}^i(t) - z_{ks}^{i-1}(t) = dz_{ks}(t)$ and the Differential Allocation-Price Relationship is

Definition 4.7. The Differential Allocation-Price Relationship for $A_{online1}$ with $\alpha_{1,ks} \geq 1$ is

$$p_{ks}(t)dz_{ks}(t) \geq \frac{c_{ks}}{\alpha_{1,ks}} dp_{ks}(t),$$

$\forall i \in [I], \forall k \in [K], \forall t \in T(l), \forall s \in S(l)$.

Lemma 4.8. *$\alpha_{1,ks} = \ln \frac{U_k - h_t}{L_k - h_t}$ and the marginal price defined in Equation (8) satisfies the Differential Allocation-Price Relationship for $A_{online1}$.*

Proof. The deferential of the marginal price function is

$$dp_{ks}(t) = (L_k - h_t)\left(\frac{U_k - h_t}{L_k - h_t}\right)^{\frac{z_{ks}(t)}{c_{ks}}} \ln\left(\frac{U_k - h_t}{L_k - h_t}\right)^{\frac{1}{c_{ks}}}.$$

The Differential Allocation-Price Relationship is

$$\begin{aligned}
&(L_k - h_t)\left(\frac{U_k - h_t}{L_k - h_t}\right)^{\frac{z_{ks}(t)}{c_{ks}}} dz_{ks}(t) \\
&\geq \frac{c_k}{\alpha_{1,ks}}(L_k - h_t)\left(\frac{U_k - h_t}{L_k - h_t}\right)^{\frac{z_{ks}(t)}{c_{ks}}} \ln\left(\frac{U_k - h_t}{L_k - h_t}\right)\frac{1}{c_{ks}} dz_{ks}(t) \\
&\Rightarrow \alpha_{1,ks} \geq \ln \frac{U_k - h_t}{L_k - h_t}.
\end{aligned}$$

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

Theorem 4.9. *The online auction $A_{online1}$ in Algorithm 1 is α_1-competitive in social welfare with $\alpha_1 = \max_{k\in[K]}\{\ln \frac{U_k - h_{max}}{L_k - h_{max}}\}$, where $h_{max} = \max_{t\in[T]} h_t$.*

Proof. $\alpha_1 = \max_{k\in[K]}\{\ln \frac{U_k - h_{max}}{L_k - h_{max}}\}$ with $h_{max} = \max_{t\in[T]} h_t$, then $\alpha_1 = \max_{k\in[K], t\in[T]}\{\ln \frac{U_k - h_t}{L_k - h_t}\}$ as when U_k and L_k are fixed, $\max_{t\in[T]}\{\ln \frac{U_k - h_t}{L_k - h_t}\} = \ln \frac{U_k - h_{max}}{L_k - h_{max}}$. According to the proof in Lemma 4.8, α_1 satisfies the Differential Allocation-Price Relationship. Under Assumption 1, we have $dp_{ks}(t) = p'_{ks}(z_{ks}(t))dz_{ks}(t) = p_{ks}^i(t) - p_{ks}^{i-1}(t)$. As a result, we can obtain that the Allocation-Price Relationship holds for α_1. Lemma 4.4 and Lemma 4.6 together imply the theorem. □

5 ONLINE AUCTION DESIGN WITH VOLUME AND PEAK CHARGES

In this section, we consider a more realistic electricity charge model that involves a peak charge term in the operating cost. We formulate the social welfare maximization problem in Section 5.1, and design an online auction in Section 5.2, which is further improved in Section 5.3.

5.1 The Social Welfare Maximization Problem

Under the assumption of truthful bidding, the social welfare maximization problem with both volume charge and peak charge considered for electricity cost is

$$\text{maximize} \sum_{i\in[I]}\sum_{s\in[S]} b_i x_{is} - g_i(\tau_i) - \sum_{t\in[T]} f_t(e(t)) - h_{peak}\max_t e(t), \tag{10}$$

subject to constraint Equations $(3a)$–$(3f)$.

Let e_{max} be the maximum of $e(t)$, i.e., $e(t) \le e_{max}, \forall t \in [T]$, and move the upper bound of $e(t)$ from the cost function to the constraint; the above convex problem can be reformulated as follows:

$$\text{maximize} \sum_{i\in[I]}\sum_{s\in[S]} b_i x_{is} - g_i(\tau_i) - \sum_{t\in[T]} h_t e(t) - h_{peak} e_{max}, \tag{11}$$

subject to

$$\text{constraint Equations } (3a)\text{–}(3e),$$

$$e(t) \le E_t, \forall t \in [T], \tag{11f}$$

$$e(t) \le e_{max}, \forall t \in [T], \tag{11g}$$

$$\tau_i, e(t), e_{max} \ge 0, x_{is}, y_{is}(t) \in \{0,1\}, \forall i \in [I], \forall s \in [S], \forall t \in [T]. \tag{11h}$$

As the convex function in the objective function has been removed, the corresponding compact exponential integer linear programming (ILP) is

$$\text{maximize} \sum_{i\in[I]}\sum_{l\in\zeta_i} b_{il} x_{il} - \sum_{t\in[T]} h_t e(t) - h_{peak} e_{max}, \tag{12}$$

subject to

$$\text{constraint Equations } (4a)\text{–}(4c),$$

$$\text{constraint Equations } (11f)\text{–}(11g),$$

$$e(t), e_{max} \ge 0, x_{il} \in \{0,1\}, \forall t \in [T], \forall i \in [I], \forall l \in \zeta_i. \tag{12f}$$

We introduce dual variables u_i, $p_{ks}(t)$, $m(t)$, σ_t, and γ_t to Equations (12a), (12b), (12c), (12d), and (12e). The dual of the relaxed ILP Equation (12) is

$$\text{minimize} \sum_{i\in[i]} u_i + \sum_{k\in[K]}\sum_{s\in[S]}\sum_{t\in[T]} c_{ks} p_{ks}(t) + \sum_{t\in[T]} \sigma_t E_t, \tag{13}$$

subject to

$$u_i \ge b_{il} - \sum_{k\in[K]}\sum_{t\in T(l)}\sum_{s\in S(l)} r_i^k\left(p_{ks}(t) + m(t)\frac{\beta_{ks}}{c_{ks}}\right), \forall i \in [I], \forall l \in \zeta_i, \tag{13a}$$

$$h_t + \sigma_t + \gamma_t \ge m(t), \forall t \in [T], \tag{13b}$$

$$h_{peak} \ge \sum_{t\in[T]} \gamma_t, \tag{13c}$$

$$p_{ks}(t), u_i, m(t), \sigma_t, \gamma_t \ge 0, \forall i \in [I], \forall k \in [K], \forall s \in [S], \forall t \in [T]. \tag{13d}$$

5.2 The First Online Auction with Peak Charges

We apply the posted pricing primal and dual framework to solve the convex problem Equation(10). We seek the help of the compact exponential ILP Equation (12) and its dual Equation (13): x_{il} remains zero unless its dual constraint Equation (13a) becomes tight. User i's utility u_i is assigned based on the current resource price, and is equal to the maximum of zero and the RHS of constraint Equation (13a). For the design of the electricity price function $m(t)$, we observe that there is a new set of dual variables γ_t associated with the peak consumption constraint ($e(t) \le e_{max}$), and the sum of all γ_t cannot exceed the peak price h_{peak}. Thus, we can interpret γ_t as the peak electricity price at time t. Let $t_{max} = \arg\max_t\{e(t)\}$, then $\gamma_{t_{max}} = h_{peak}, \gamma_t = 0, \forall t \neq t_{max}$. Furthermore, dual constraints Equation (13b) indicate that if we let $\sigma_t = 0$, $m(t) = h_t + \gamma_t$ is the electricity price at time t, consisting of both volume price h_t and peak price γ_t. However, it is impossible to determine which slot has the peak consumption, as we don't have complete knowledge about the system over its entire lifespan in the arrival of bids. We first adopt a straightforward way to allocate h_{peak} to all γ_t, by equally allocating the peak price to all slots. The capital price function $p_{ks}(t)$ is still a function of $z_{ks}(t)$, increasing from $L_k - m(t)$ to $U_k - m(t)$. More specifically, $p_{ks}(t)$ and $m(t)$ are defined as

$$p_{ks}(t)(z_{ks}(t)) = \left(L_k - h_t - \frac{h_{peak}}{T}\right)\left(\frac{U_k - h_t - \frac{h_{peak}}{T}}{L_k - h_t - \frac{h_{peak}}{T}}\right)^{\frac{z_{ks}(t)}{c_{ks}}}, \tag{14}$$

$$m(t) = h_t + \gamma_t = h_t + \frac{h_{peak}}{T}, \forall t \in [T]. \tag{15}$$

ALGORITHM 3: A Primal-Dual Online Auction $A_{online2}$

Input: bidding language $\{B_i\}, \{c_{ks}\}, \{\beta_k\}, \{E_t\}, \{h_t\}, h_{peak}$

1: Define function $p_{ks}(z_{ks}(t))$ according to Equation (14);
2: Initialize $x_{is} = 0$, $y_{is}(t) = 0$, $z_{ks}(t) = 0$, $\tau_i = 0$, $x_{il} = 0$, $u_i = 0, p_{ks}(t) = 0, \gamma_t = \frac{h_{peak}}{T}, m(t) = h_t + \gamma_t$, $e(t) = 0, e_{max} = 0, \sigma_t = 0, \forall i \in [I], l \in \zeta_i, k \in [K], s \in [S], t \in [T]$;
3: Line 4–10 in Algorithm 1;
4: $e_{max} = \max_{t\in[T]} e(t)$;

$A_{online2}$ in Algorithm 3 is the first online auction for the peak charge model, with the scheduling Algorithm 2 running for each user. The properties of $A_{online2}$ are described in Theorem 5.1.

Theorem 5.1. *The online auction $A_{online2}$ for the peak charge model is a truthful auction that outputs a feasible solution for problem Equations (11), (12), and (13) in polynomial time.*

Proof. Similar proofs for the polynomial running time and truthfulness can be found in Theorem 4.2 and Theorem 4.3, respectively, and we omit the details here. For the feasibility of $A_{online2}$, a similar proof can also be found in Theorem 4.2. The main difference lies in the dual constraint Equations (13b) and (13c). Constraints Equation (13b) are satisfied because $\sigma_t = 0$ and $m(t) = h_t + \gamma_t$. Constraints Equation (13c) hold as $h_{peak} = \sum_{t\in[T]} \gamma_t$. □

5.3 A More Intelligent Online Auction

In Algorithm 3, we did not keep track of the hitherto peak consumption, which makes the algorithm less intelligent in the update of peak price γ_t. We next manipulate γ_t in a more sophisticated way to take into account the current level of peak consumption. The basic idea is to let $\gamma_t = h_t$ at the beginning. After the ith user is handled, γ_t is increased by a certain value for the current peak slot. As there are a total of I users and the sum of all γ_t is at most h_{peak}, γ_t is increased by h_{peak}/I

in each round for the current peak slot, $p_{ks}(t)$ is updated based on the current value of $m(t)$, as shown in lines 5–7 in Algorithm 5.

ALGORITHM 4: The Improved Primal-Dual Online Auction $A_{online3}$

Input: bidding language $\{B_i\}, \{c_{ks}\}, \{\beta_k\}, \{E_t\}, \{h_t\}, h_{peak}$

1: Initialize $x_{is} = 0$, $y_{is}(t) = 0$, $z_{ks}(t) = 0$, $\tau_i = 0$, $x_{il} = 0$, $u_i = 0, p_{ks}(t) = 0, \sigma_t = 0, m(t) = h_t, e(t) = 0$, $e_{max} = 0, \gamma_t = 0, \forall i \in [I], l \in \zeta_i, k \in [K], s \in [S], t \in [T]$;
2: **Upon the arrival of the ith user**
3: $(x_{is}, \{y_{is}(t)\}, p_i, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{m(t)\}, \{e(t)\}) = A_{core2}(B_i, \{c_{ks}\}, \{E_t\}, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{m(t)\}, \{e(t)\})$;
4: Lines 6–10 in Algorithm 1;
5: $e_{max} = \max_{t \in [T]} e(t)$;

ALGORITHM 5: A Scheduling Algorithm A_{core2}.

Input: $B_i, \{c_{ks}\}, \{E_t\}, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{m(t)\}, \{e(t)\}$
Output: $x_{is}, \{y_{is}(t)\}, p_i, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{m(t)\}, \{e(t)\}$

1: Lines 1–15 in Algorithm 2;
2: **if** $b_i - \hat{\mathcal{P}} > 0$ **then**
3: Lines 17–19 in Algorithm 2;
4: **end if**
5: $t_{max} = \arg\max_{t \in [T]} e(t)$;
6: $\gamma_{t_{max}} = \gamma_{t_{max}} + h_{peak}/I, m(t_{max}) = h_{t_{max}} + \gamma_{t_{max}}$;
7: $p_{k\hat{s}}(t) = (L_k - m(t))(\frac{U_k - m(t)}{L_k - m(t)})^{\frac{z_{ks}(t)}{c_{ks}}}, \forall k \in [K], t \in \hat{I}$;
8: **Return** $x_{is}, \{y_{is}(t)\}, p_i, \{p_{ks}(t)\}, \{z_{ks}(t)\}, \{m(t)\}, \{e(t)\}$

$A_{online3}$ in Algorithm 4 is the improved primal-dual online auction. The new one-round algorithm A_{core2} in Algorithm 5 is executed for each user with the scheduling approach and new price functions. We next analyze the properties of $A_{online3}$, including correctness, polynomial running time, truthfulness, and competitiveness in social welfare. The two ways of updating dual variables for peak consumption are further compared in simulation studies in Section 6.

Theoretical Analysis.

(i) Correctness, Polynomial Running Time, and Truthfulness.

Theorem 5.2. *$A_{online3}$ is a truthful auction that computes a feasible solution for problem Equations (11), (12), and (13) with polynomial running time.*

Proof. There are a total of I users, so constraints Equation (13c) hold because $\sum_{t \in [T]} \gamma_t$ equals h_{peak} according to line 6 in Algorithm 5. The rest of the proof is similar to that of Theorem 5.1 and is omitted here. □

(ii) Competitive Ratio.

The proof of the competitive ratio follows the same structure as that in Section 4.3. Let P_i and D_i be the primal Equation (12) and dual Equation (13) objective values, respectively, returned by $A_{online3}$ after processing user i's job. By Lemma 4.4, $A_{online3}$ is α_3-competitive in social welfare if $P_i - P_{i-1} \geq \frac{1}{\alpha_3}(D_i - D_{i-1})$. We next define two relationships in Definition 5.3 and Definition 5.4, respectively. We show that if both of them hold, then $P_i - P_{i-1} \geq \frac{1}{\alpha_3}(D_i - D_{i-1})$ also holds for a

certain α_3 in Lemma 5.5. We obtain the value of α_3 through the analyses in Lemma 5.7 and Lemma 5.8, and finally prove that $A_{online3}$ is α_3-competitive in Theorem 5.9.

Definition 5.3. The Allocation-Price Relationship for $A_{online3}$ with $\alpha_{p1} \geq 1$ is

$$p_{ks}^{i-1}(t)(z_{ks}^{i}(t) - z_{ks}^{i-1}(t)) \geq \frac{1}{\alpha_{p1}} c_{ks}(p_{ks}^{i}(t) - p_{ks}^{i-1}(t)),$$
$$\forall i \in [I], \forall k \in [K], \forall t \in T(l), \forall t \in T(l).$$

Definition 5.4. The Primal Objective Increment Relationship for $A_{online3}$ with $\alpha_{p2} > 0$ is

$$P_i - P_{i-1} \geq \frac{1}{\alpha_{p2}} h_{peak}(e_{max}^{i} - e_{max}^{i-1}), \forall i \in [I].$$

Lemma 5.5. *If the Allocation-Price Relationship holds for a given $\alpha_{p1} \geq 1$ and Primal Objective Increment Relationship holds for a given $\alpha_{p2} > 0$, then $A_{online3}$ guarantees*

$$P_i - P_{i-1} \geq \frac{1}{\alpha_{p1}(1+\alpha_{p2})}(D_i - D_{i-1}), \forall i \in [I].$$

Proof. Again, we assume that user i's job is accepted and allocated according to schedule l. Note that $m(t) = h_t + \gamma_t$, then the increment of the primal objective value is

$$\begin{aligned}
P_i - P_{i-1} &= b_{il} - \sum_{t \in T(l)} h_t(e^i(t) - e^{i-1}(t)) - h_{peak}(e_{max}^{i} - e_{max}^{i-1}) \\
&= u_i + \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} r_i^k \left(p_{ks}^{i-1}(t) + m^{i-1}(t) \frac{\beta_{ks}}{c_{ks}} \right) \\
&- \sum_{t \in T(l)} h_t(e^i(t) - e^{i-1}(t)) - h_{peak}(e_{max}^{i} - e_{max}^{i-1}) \\
&= u_i + \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} r_i^k \left(p_{ks}^{i-1}(t) + \gamma^{i-1}(t) \frac{\beta_{ks}}{c_{ks}} \right) \\
&- h_{peak}(e_{max}^{i} - e_{max}^{i-1}).
\end{aligned}$$

The increase of the dual objective value is

$$D_i - D_i = u_i + \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} c_{ks}(p_{ks}^{i}(t) - p_{ks}^{i-1}(t)).$$

By the Primal Objective Increment Relationship, we have

$$\begin{aligned}
&(1+\alpha_{p2})(P_i - P_{i-1}) \geq P_i - P_{i-1} + h_{peak}(e_{max}^{i} - e_{max}^{i-1}) \\
&\geq P_i - P_{i-1} + h_{peak}(e_{max}^{i} - e_{max}^{i-1}) - \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} r_i^k \gamma^{i-1}(t) \frac{\beta_{ks}}{c_{ks}} \\
&= u_i + \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} r_i^k p_{ks}^{i-1}(t) \\
&\geq u_i + \frac{1}{\alpha_{p1}} \sum_{k \in [K]} \sum_{t \in T(l)} \sum_{s \in S(l)} c_{ks}(p_{ks}^{i}(t) - p_{ks}^{i-1}(t)) \\
&\geq \frac{1}{\alpha_{p1}}(D_i - D_i).
\end{aligned}$$

Therefore, $P_i - P_{i-1} \geq \frac{1}{\alpha_{p1}(1+\alpha_{p2})}(D_i - D_{i-1})$. □

To compute the value of α_{p1}, we define a differential version of the Allocation-Price Relationship for each pair of k and s, and provide the value of $\alpha_{p1,ks}$ in Lemma 5.7; α_{p1} is just the maximum among all $\alpha_{p1,ks}$.

Definition 5.6. The Differential Allocation-Price Relationship for $A_{online3}$ with $\alpha_{p1,ks}$ is

$$p_{ks}(t)dz_{ks}(t) \geq \frac{c_{ks}}{\alpha_{p1,ks}} dp_{ks}(t),$$

$\forall i \in [I], \forall k \in [K], \forall t \in T(l), \forall s \in S(l)$.

LEMMA 5.7. *$\alpha_{p1,ks} = \ln \frac{U_k - m(t)}{L_k - m(t)}$ and the marginal price defined at line 7 in Algorithm 5 satisfies the Differential Allocation-Price Relationship for $A_{online3}$.*

PROOF. We observe that the Differential Allocation-Price Relationship for $A_{online3}$ is same as that for $A_{online1}$. Thus, the detailed proof can be found in Lemma 4.8. □

LEMMA 5.8. *$\alpha_{p2} \geq \frac{h_{peak} \sum_{k\in[K]} \sum_{s\in S(l)} r_i^k \beta_{ks}/c_{ks}}{b_i - (w_i+1) h_{peak} \sum_{k\in[K]} \sum_{s\in S(l)} r_i^k \beta_{ks}/c_{ks}}$ satisfies the Primal Objective Increment Relationship.*

PROOF. If user i's job is not allocated to the slot with the final maximum power consumption, $e^i_{max} - e^{i-1}_{max} = 0$, α_{p2} can be any value. Otherwise, $h_{peak}(e^i_{max} - e^{i-1}_{max}) = h_{peak} \sum_{s\in S(l)} \sum_{k\in[K]} r_i^k \beta_{ks}/c_{ks}$, and $P_i - P_{i-1} = b_i - \sum_{t\in T(l)} \sum_{k\in[K]} \sum_{s\in S(l)} h_t r_i^k \beta_{ks}/c_{ks} - h_{peak} \sum_{s\in S(l)} \sum_{k\in[K]} r_i^k \beta_{ks}/c_{ks} \geq b_i - (w_i+1) h_{peak} \sum_{k\in[K]} \sum_{s\in S(l)} r_i^k \beta_{ks}/c_{ks}$, thus, $\frac{h_{peak}(e^i_{max} - e^{i-1}_{max})}{P_i - P_{i-1}} \leq \frac{h_{peak} \sum_{k\in[K]} \sum_{s\in S(l)} r_i^k \beta_{ks}/c_{ks}}{b_i - (w_i+1) h_{peak} \sum_{k\in[K]} \sum_{s\in S(l)} r_i^k \beta_{ks}/c_{ks}}$. And the value of α_{p2} satisfies the Primal Objective Increment Relationship. □

THEOREM 5.9. *The online auction $A_{online3}$ in Algorithm 4 is α_3-competitive in social welfare with $\alpha_3 = \alpha_{p1}(1 + \alpha_2)$, where*

$$\alpha_{p1} = \max_k \left\{ \ln \frac{U_k - h_{max} - h_{peak}}{L_k - h_{max} - h_{peak}} \right\} \text{with } h_{max} = \max_t h_t, \text{ and}$$

$$\alpha_2 = \max_i \frac{e_c}{b_i - (w_i + 1)e_c} \text{ with } e_c = \max_{k,s} \left\{ h_{peak} \sum_{k\in[K]} r_i^k \frac{\beta_{ks}}{c_{ks}} \right\}.$$

PROOF. α_{p1} satisfies the Allocation-Price Relationship for $A_{online3}$ as $\alpha_{p1} = \max_{k,s}\{\alpha_{p1,ks}\}$. α_2 also satisfies the Primal Objective Increment Relationship for $A_{online3}$ as α_2 is the maximum value among all possible α_{p2}. Therefore, $A_{online3}$ is α_3-competitive in social welfare. The value of α_{p1} and α_2 depend on the system configuration, and our trace-driven simulation studies show that $\alpha_3 < 2$ with $\alpha_{p1} \approx 1.4$ and $\alpha_2 \approx 0.3$. □

6 PERFORMANCE EVALUATION

We evaluate the performance of our online auctions $A_{online1}$, $A_{online2}$, and $A_{online3}$ through large-scale simulation studies based on real-world traces. We first briefly introduce the simulation setup. Trace Version 1 in Google Cluster Data (Google 2016) contains information about jobs running on Google compute cells, including start time, execution, duration, and normalized job demand (CPU and RAM). We translate each job into a bid, requesting two types of resources at demands extracted from the traces. We assume each time slot is 5 minutes (Google 2016) and each job consumes $[1, 12]$ slots, arriving sequentially in 18 hours. Each job's deadline is randomly generated between its arrival time and the system end time. We set the bidding price of each job by multiplying the overall resource demands by unit prices randomly picked within the range $[L_k, U_k]$. The default

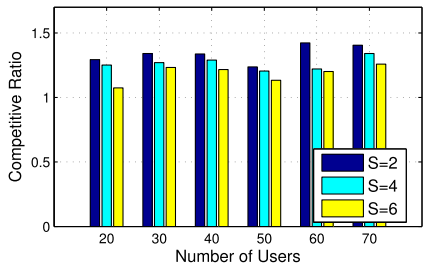

Fig. 1. Competitive ratio of $A_{online1}$ under different numbers of users and servers.

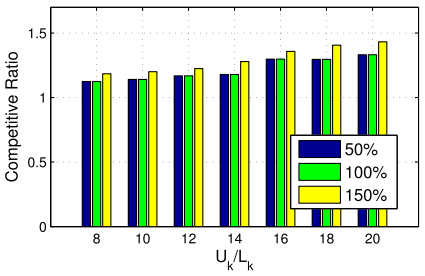

Fig. 2. Competitive ratio of $A_{online1}$ under different U_k/L_k and estimated U_k.

value of L_k is 5 and U_k is 50 for $A_{online1}$. We let $L_k = 905$ and $U_k = 2000$ in $A_{online2}$ and $A_{online3}$. The capacity of type-k resource in server s, c_{ks}, is set to 1 as the resource demand is in normalized units.

For the power consumption of a server, parameter β_{ks} is set within $[20, 60]$ for CPU and within $[0.2, 2]$ or RAM (Tian and Zhao 2014; Kansal et al. 2010). We assume that the data center is powered by BC Hydro with a peak charge of \$9.95 per kWh and a volume charge of ¢4.86 per kWh (Zhang et al. 2015a; Hydro 2016). The value of h_t is generated by adding randomness to the volume charge of ¢4.86 per kWh. The available power at each time slot E_t is set to within the range of [20,100] kW based on a report of data center server power usage and required demand response power reduction (ZDnet 2013).

6.1 Performance of $A_{online1}$

We first examine the competitive ratio achieved by $A_{online1}$. The optimal social welfare of the convex problem Equation (3) is computed by CVX with the Gurobi Optimizer. Figure 1 shows the competitive ratio of $A_{online1}$ under different numbers of users and servers. We observe that $A_{online1}$ always performs well with a lower competitive ratio (<1.5), which is noticeably better than the theoretically proven bounds. The competitive ratio decreases as the number of servers increases, and fluctuates when the number of users grows. Our algorithm $A_{online1}$ allocates a user's job on the cheapest server to maximize its utility. Therefore, the algorithm has a larger solution space to explore when the number of servers is large, leading to a better competitive ratio. The number of users doesn't influence the value of ratio, as confirmed by the analysis in Theorem 4.9. Recall that U_k and L_k are the maximum and minimum unit price of type-k resource, respectively, defined in the price function Equation (8). Figure 2 illustrates that $A_{online1}$ still achieves a good competitive ratio when we vary the value of U_k/L_k and use the estimated values of U_k as the input of $A_{online1}$. We notice that the competitive ratio becomes larger with the increment of U_k/L_k, while both underestimation and overestimation have minor impact on the performance, as compared to that achieved by the real U_k (labelled by 100%). Theorem 4.9 reveals that U_k/L_k determines the competitive ratio, which is consistent with the downward trend in Figure 2. Moreover, underestimation is slightly better than overestimation, due to the reason that overestimation leads to a higher price, filtering out jobs that are supposed to be accepted.

We next investigate the social welfare and the cloud service provider's revenue achieved by $A_{online1}$. Figure 3 compares the social welfare achieved by $A_{online1}$ to the optimal social welfare under different lengths of job execution time (w_i). $A_{online1}$ always achieves close-to-optimal performance regardless the value of w_i. When $w_i \leq 18$, the social welfare increases when the user requests more slots for its job, which is reasonable because the social welfare is mostly contributed

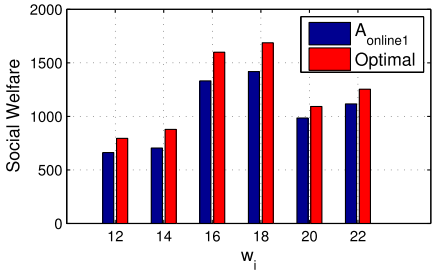

Fig. 3. Social welfare of $A_{online1}$ under different values of w_i.

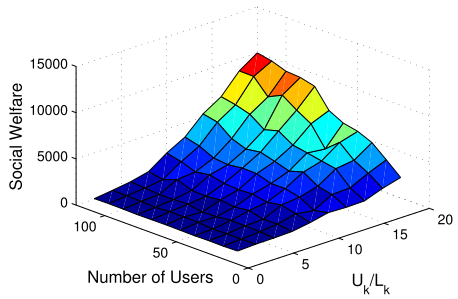

Fig. 4. Social welfare of $A_{online1}$ under different numbers of users and U_k/L_k.

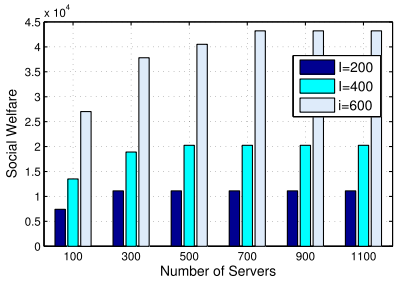

Fig. 5. Social welfare of $A_{online1}$ under different values of I and S.

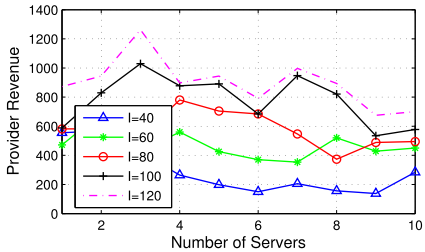

Fig. 6. Cloud service provider's revenue of $A_{online1}$ with different numbers of servers and users.

by the bidding price, and the user will raise its bidding price when its job needs a long execution time. The gap between the social welfare returned by $A_{online1}$ and the optimal social welfare becomes larger with the increment of w_i, as long execution time brings more computation difficulties for $A_{online1}$ to approach the optimal solution. Another interesting observation is that the social welfare drops sharply when $w_i > 20$. This is because the competition for resources in each time slot is fiercer with a larger w_i, then the number of winners would decrease when the number of users is fixed, leading to a smaller overall social welfare.

The 3D figure in Figure 4 shows that a large social welfare comes with a large number of users and a high value of U_k/L_k. The underlying reason is that $A_{online1}$ can select more high-value bids when there is a large set of users participating in the auction. Furthermore, the bidding price rises when the value of U_k/L_k increases, and hence a higher social welfare is achieved by high-value bids. In Figure 5, we consider a large input scale, and vary the number of users and the number of servers. Again, we observe an upward trend in the social welfare with the increases of the number of users and the number of servers. It remains steady when the number of servers is larger than the number of users, as all users' jobs are accepted, achieving the same social welfare. In Figure 6, we plot the revenue of the cloud service provider under different numbers of users and servers. The change of the number of servers doesn't have major influence on the revenue. The cloud service provider is able to gain higher profit with a larger set of users, as more jobs with high bidding prices would be accepted to contribute to the revenue.

The performance of $A_{online1}$ in terms of winner satisfaction, as measured by the percentage of winning users, is demonstrated in Figure 7. We observe that more jobs are successfully allocated when the number of participating users is small and U_k/L_k is large. The reason can be explained

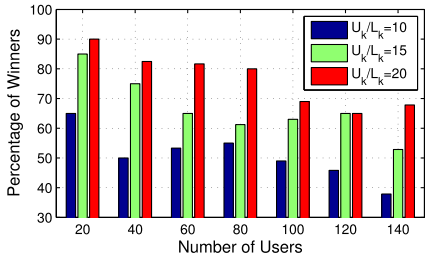

Fig. 7. Percentage of winners of $A_{online1}$ with different numbers of users and U_k/L_k.

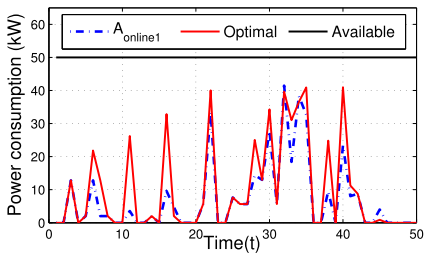

Fig. 8. Power consumption over the system time for $A_{online1}$ with $Et = 50$.

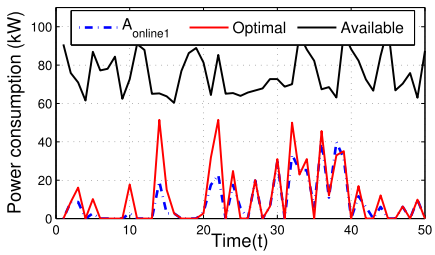

Fig. 9. Power consumption over the system time for $A_{online1}$ with $Et \in [60, 100]$.

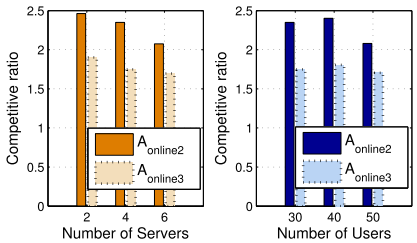

Fig. 10. Competitive ratio of $A_{online2}$ and $A_{online3}$.

as the following: the number of winners is almost fixed and limited by the resource capacity and the available power. The users face stiff competition when a large number of users submits bids to the cloud. Furthermore, the winner is determined by the current price of the resource, which rises from L_k to U_k. When L_k is close to U_k, the difference between the bidding prices is small. As a result, more bids with similar bidding prices are rejected as the price is increased during each round.

Finally, we vary the distribution of the amount of power reduction required by the EDR (and hence the amount of available power) in each time slot, and plot the power consumption over the system time in Figure 8 and Figure 9. There are only minor differences in the competitive ratio under these two distributions: 1.1991 and 1.2055. We also observe that the power consumption in both the optimal solution and $A_{online1}$ fluctuates and doesn't follow the distribution of E_t.

6.2 Performance of $A_{online2}$ and $A_{online3}$

In this subsection, we evaluate the performance of our online auctions under the peak charge model. We first compare the social welfare achieved by $A_{online2}$ and $A_{online3}$ with the offline optimum. The left figure in Figure 10 shows the competitive ratio of $A_{online2}$ and $A_{online3}$ when we vary the number of servers, and the right figure plots the competitive ratio with different numbers of users. Figure 10 confirms that $A_{online3}$ is a more intelligent auction with a lower competitive ratio (<2). Both $A_{online2}$ and $A_{online3}$ perform well with a small loss in social welfare when we involve the peak charge term. The increment of the competitive ratio is smaller than 0.5 for $A_{online3}$. In addition, the downward trend observed in Figure 10 is similar to that in Figure 1. Figure 11 presents the detailed comparison among $A_{online2}$, $A_{online3}$, and offline optimum in terms of provider's profit, volume charge, peak charge, and social welfare. Compared to $A_{online2}$, $A_{online3}$ generates a higher profit for the cloud service provider with a lower operation cost. We also

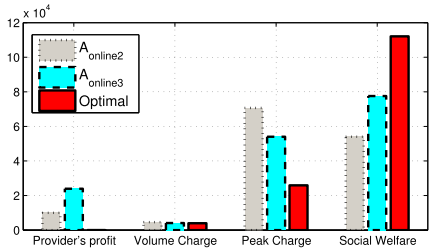

Fig. 11. Comparison between $A_{online2}$ and $A_{online3}$.

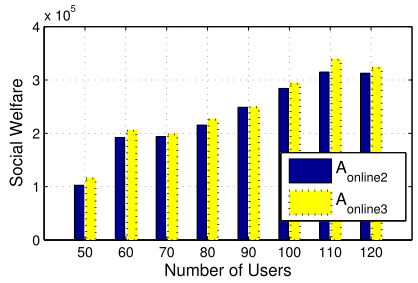

Fig. 12. Social welfare of $A_{online2}$ and $A_{online3}$ with different numbers of users.

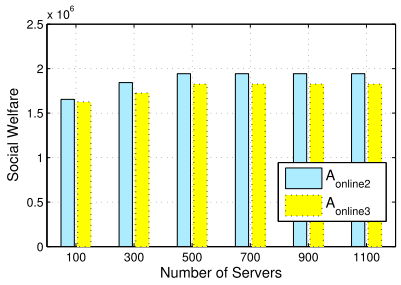

Fig. 13. Social welfare of $A_{online2}$ and $A_{online3}$ under different values of S when $I = 400$.

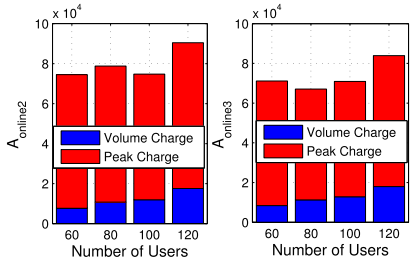

Fig. 14. Electricity charge of $A_{online2}$ and $A_{online3}$ with different numbers of users.

observe that the peak charge contributes a major part of the operation cost. The optimal solution achieves a higher social welfare because it spreads out the workload evenly to reduce both the volume charge and the peak charge.

Figure 12 demonstrates that the social welfare achieved by $A_{online2}$ and $A_{online3}$ increases when the number of users grows. The underlying reason is similar to the explanation for Figure 4. Furthermore, $A_{online3}$ always brings a higher social welfare, as it tracks the current peak to avoid accumulating power consumption on the same slot, which contributes to cutting down the eventual peak consumption. In Figure 13, we fix the number of users to 400 and change the number of servers. We obtain a similar observation as compared to Figure 5. The increase of the number of servers has a positive impact on the social welfare. Figure 14 illustrates the operation costs of the two online auctions with different numbers of users. We observe that although the volume charge of the two schemes are similar, the peak charge generated by $A_{online3}$ is smaller than that of $A_{online2}$. The volume charge increases gradually when there are more users participating in the auction, while the peak charge fluctuates. This is because the volume charge is the electricity cost over all slots, which increases when more jobs are executing, while peak charge is the cost that occurs in one slot only.

We next investigate user satisfaction in Figure 15. The percentage of winners decreases in both schemes when the number of users increases, and $A_{online3}$ accepts more jobs than $A_{online2}$ does, leading to a higher social welfare. This is due to the setting of the electricity price function $m(t)$. The value of $m(t)$ in $A_{online3}$ is lower than that in $A_{online2}$ at the beginning, improving the probability of winning. The last figure in Figure 14 depicts the power consumption over the system time. The peak consumption is marked by a circle. An interesting observation is that the optimal

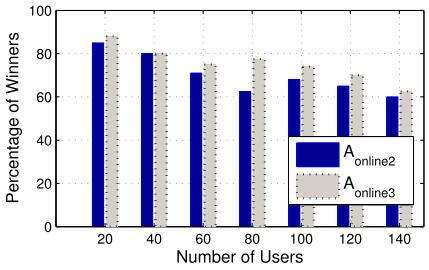

Fig. 15. Percentage of winners for $A_{online2}$ and $A_{online3}$ with different numbers of users.

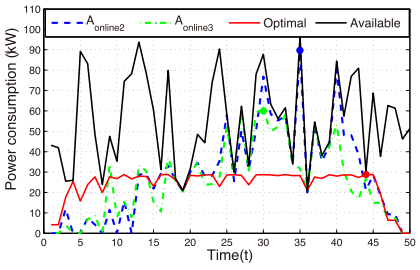

Fig. 16. Power consumption over the system time for $A_{online2}$ and $A_{online3}$.

approach averages the workload to cut down the peak consumption, which is quite different from the observation in $A_{online1}$. The peak in $A_{online3}$ occurs in the middle, between that of $A_{online2}$ and the optimal one.

7 CONCLUSION

We studied data center EDR where (i) the power grid dictates an upper bound in power consumption in each time slot during the EDR period and (ii) cloud jobs with soft deadlines arrive in an online fashion. We adapt the classic primal-dual framework for efficient approximation algorithm design, and employ a posted-pricing framework for truthful online mechanism design, to derive a truthful online auction that runs efficiently and approaches optimal social welfare. However, it turns out that the above techniques alone are not sufficient. A salient feature of this work lies in the new compact exponential optimization technique we introduce, which works in concert with a dual oracle to handle the job-completion time constraints imposed by their soft deadlines. Our compact exponential method may further shed light on other algorithm and mechanism design scenarios where the optimization problem contains both conventional and non-conventional constraints, such as delay-constrained optimization in cyber physical systems.

REFERENCES

Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. 2014. A dynamic near-optimal algorithm for online linear programming. *Operations Res.* 62, 4 (2014), 876–890.

H. Bauschke and Yves Lucet. 2012. What is a fenchel conjugate? *Not. AMS* 59, 1 (2012), 1–3.

S. Boyd and L. Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press.

N. R. Devanur, K. Jain, T. Mai, V. V. Vazirani, and S. Yazdanbod. 2016. New convex programs for fisher's market model and its generalizations. *arXiv preprint, arXiv:1603.01257.*

ENERNOC. 2016. *Ensuring U.S. Grid Security and Reliability: U.S. EPA's Proposed Emergency Backup Generator Rule.* Retrieved June 2016, from https://goo.gl/XyJQVg.

Google Cluster Data. 2016. TraceVersion1. Retrieved from https://code.google.com/p/googleclusterdata/wiki/TraceVersion1.

Z. Huang and A. Kim. 2015. Welfare maximization with production costs: A primal dual approach. In *Proceedings of ACM-SIAM SODA.*

BC Hydro. 2016. *Power Smart.* Retrieved from https://www.bchydro.com/index.html.

Patrick Jaillet and Xin Lu. 2012. Near-optimal online algorithms for dynamic resource allocation problems. *Arxiv:1208.2596.*

A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. 2010. Virtual machine power metering and provisioning. In *Proceedings of ACM SoCC.*

Z. Liu, I. Liu, S. Low, and A. Wierman. 2014. Pricing data center demand response. In *Proceedings of ACM SIGMETRICS.*

A. Misra. 2016. *Responding Before Electric Emergencies.* Retrieved from http://goo.gl/eNyquJ.

NRDC. 2016. *America's Data Centers Consuming and Wasting Growing Amounts of Energy.* Retrieved from http://www.nrdc.org/energy/data-center-efficiency-assessment.asp.

PJM. 2016. *Retail Electricity Consumer Opportunities for Demand Response in PJM?s Wholesale Markets*. Retrieved from https://www.pjm.com/~/media/markets-ops/dsr/end-use-customer-fact-sheet.ashx/.

PJM. 2014. *Emergency Demand Response (Load Management) Performance Report 2013/2014*.

P. Samadi, H. Mohsenian-Rad, R. Schober, and V. Wong. 2012. Advanced demand side management for the future smart grid using mechanism design. *IEEE Trans. Smart Grid* 3, 3 (2012), 1170–1180.

W. Shi, L. Zhang, C. Wu, Z. Li, and F. Lau. 2014. An online auction framework for dynamic resource provisioning in cloud computing. In *Proceedings of ACM SIGMETRICS*.

W. D. Tian and Y. D. Zhao. 2014. *Optimized Cloud Resource Management and Scheduling: Theories and Practices*. Elsevier Science.

ZDnet. 2013. *Toolkit: Calculate datacenter server power usage*. Retrieved from http://goo.gl/iB9hZv.

Q. Wang, K. Ren, and X. Meng. 2012. When cloud meets ebay: Towards effective pricing for cloud computing. In *Proceedings of IEEE INFOCOM*.

A. Wierman, Z. Liu, I. Liu, and H. Mohsenian-Rad. 2014. Opportunities and challenges for data center demand response. In *Proceedings of IEEE IGCC*.

S. Zaman and D. Grosu. 2013. A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds. *IEEE Trans. Cloud Comput.* 1, 2 (2013), 129–141.

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

L. Zhang, Z. Li, C. Wu, and S. Ren. 2015a. Online electricity cost saving algorithms for co-location data centers. In *Proceedings of ACM SIGMETRICS*.

L. Zhang, S. Ren, C. Wu, and Z. Li. 2015b. A truthful incentive mechanism for emergency demand response in colocation data centers. In *Proceedings of IEEE INFOCOM*.

X. Zhang, Z. Huang, C. Wu, Z. Li, and F Lau. 2015. Online auctions in IaaS clouds: Welfare and profit maximization with server costs. In *Proceedings of ACM SIGMETRICS*.

X. Zhang, C. Wu, Z. Li, and F. Lau. 2015c. A truthful (1-ε)-optimal mechanism for on-demand cloud resource provisioning. In *Proceedings of IEEE INFOCOM*.

R. Zhou, Z. Li, C. Wu, and M. Chen. 2015. Demand response in smart grids: A randomized auction approach. *IEEE J. Sel. Areas Commun.* 33, 12 (2015), 2540–2553.

Ruiting Zhou, Zongpeng Li, Chuan Wu, and Zhiyi Huang. 2017. An efficient cloud market mechanism for computing jobs with soft deadlines. *IEEE/ACM Trans. Netw.* 25, 2 (2017), 793–805.

Z. Zhou, F. Liu, and Z. Li. 2015. Pricing bilateral electricity trade between smart grids and hybrid green datacenters. In *Proccedings of ACM SIGMETRICS*.

Received June 2016; revised September 2017; accepted December 2017