OFM: An Online Fisher Market for Cloud Computing

Abhinandan S. Prasad*, Mayutan Arumaithurai*, David Koll*‡, Yuming Jiang†, Xiaoming Fu*

*University of Goettingen, Germany
{asridha,mayutan.arumaithurai,koll,fu}@cs.uni-goettingen.de

†Norwegian University of Science and Technology (NTNU), Norway
ymjiang@ieee.org

‡Continental AG, Germany
david.koll@conti.de

***Abstract*—Currently, cloud computing is a primary enabler of new paradigms such as edge and fog computing. One open issue is the pricing of services or resources. Current pricing schemes are usually *oligopolistic* and not fair. In this work, we propose OFM, an online learning based marketplace that dynamically determines the price for arbitrary resource types based on supply and demand existing at that period. Unlike state of the art solutions, OFM can handle an arbitrary number of customers and resource types at every instance of time. It further performs integral allocation of resources and thereby avoids the unbounded integrality gap. We evaluate OFM with both real and synthetic datasets to reflect varying buying interests, the number of resources sold and market volatility to demonstrate the feasibility of our solution for several realistic scenarios. We observe that (i) OFM achieves about 95% of optimal prices and maximizes the Nash social welfare (NSW); (ii) OFM converges faster and works with different data distributions; and (iii) OFM scales for a large number of resources and buyers and computational time is in the order of *microseconds*, making it applicable for real-time use cases especially in edge markets.**

I. Introduction

Cloud computing is a widely popular paradigm of offering services over the Internet. Currently, most often these services are virtualized to capitalize on the inherent scaling flexibility of virtualization. Hence, cloud computing has become a significant enabler for new paradigms such as edge computing and network function virtualization (NFV). *Cloud service or resource pricing*[1] is one of the critical challenges for any cloud service provider as *pricing affects profit and customer experience simultaneously*. Moreover, cloud customer demands are *online* and hence require *online* pricing [1]. Consequently, resource pricing has captured much attention from both academia and industry [2].

Currently, "pay-as-you-go" or *fixed pricing* model is popular among cloud service providers where resource prices are computed based on the usage (e.g., hourly billing). This pricing strategy maximizes revenue only if the customer behavior is *well defined* (e.g., more spending on weekends) and their arrivals are *temporally invariant* [3]. However, both customer demands and arrivals in the Cloud are ad-hoc [1]. Moreover, the default fixed pricing favors cloud service providers contractually [4]. Hence, fixed pricing approach neither maximizes the revenue of cloud service provider nor is fair towards the customers.

The cloud computing community is exploring alternative online pricing schemes such as *posted pricing* or *leave-it-or-take-it* and *dynamic pricing* [5]–[7] to address above-mentioned fixed pricing limitations. In posted pricing, the seller publishes prices. Customers procure resources only if the prices are acceptable. However, prices do not necessarily reflect resources' supply and demand or market demand. Hence, it cannot guarantee a complete allocation [8].

Conversely, dynamic pricing leads to efficient resource utilization and satisfaction of user demands [1]. Furthermore, prices reflect the market demand. There are some efforts from the industry towards dynamic pricing such as *spot pricing*. In spot pricing, a user can specify the maximum prices he/she is willing to pay, and the instances are allocated until the spot instance prices are within the maximum price. However, these prices do not reflect the market demand [7]. Auctions are another popular form of dynamic pricing. However, auctions are *not fair* since resources are allocated only to the winner [9].

Generally, fixed prices are profit driven. However, profit-driven approach will lead to customer attrition due to higher prices. Hence, cloud computing community is exploring social welfare maximization. In cloud computing, maximizing social welfare improves not only the overall system efficiency but also assures a superior user experience [1]. Furthermore, maximizing social welfare is apt for both public and private clouds [10]. Therefore, maximizing social welfare is beneficial for both the cloud service providers and users.

In the literature, there are three types of social welfare, namely *utilitarian* (maximizing sum of utility), *egalitarian* or *max-min fair* (maximizing minimum utility), and *Nash social welfare* (NSW)(geometric mean of utility) [8]. In utilitarian, the allocation maximizes the overall utility of customers. In egalitarian, the allocation maximizes the minimum utility of the customers, rewarding customers with lower utility. NSW is the *Pareto outcome* between utilitarian and egalitarian approaches [11]. In other words, NSW achieves a balance between *efficiency* and *fairness*.

In the area of cloud computing, most of the current dynamic pricing works [7] are *utilitarian*. Interestingly, the *Fisher* or *Eisenberg-Gale* market is widely prevalent in algorithmic game theory for computing prices that maximize NSW [12]–[16]. The online variants of Fisher markets are proposed as well [17], [18]. However, these solutions cannot be applied in the current cloud context due to the following challenges:

C1: The online algorithms are analyzed using adversarial models (input behavior models). The state-of-the-art solutions are either fully adversarial or stochastic. These adversarial models fail to capture the application-specific behavior of the input [19].

C2: Customer valuations are not independent and identically distributed [20] and can vary dynamically. Hence, the pricing decisions often need to be time-critical and online [1].

[1]In this work we use the terms *services* and *resources* interchangeably.

C3: Cloud resources are usually virtualized and therefore rendered intangible [5]. The state-of-the-art solutions, which, when dealing with integer allocation, typically round fractional allocations to the nearest integer solution, but the resulting rounding difference, also called as *integrality gap*, is unbounded for a market, i.e., the difference grows *exponentially* with the number of buyers [21]. In other words, *fairness* drops *exponentially* with the number of buyers.

We believe that there exists no solution that solves the above challenges together. To fill this gap, we introduce OFM, an *online learning* based Fisher market that supports online pricing and allocation. Furthermore, OFM adapts and integrates several techniques, making itself a novel online Fisher market solution for the Cloud and addresses the above-mentioned challenges: (i) OFM random permutation model addresses challenge C1 by modeling application-specific behavior; (ii) OFM updates are in closed-form expressions and hence computationally efficient so that OFM can be scaled for a large number of buyers or customers. Hence, addresses challenge C2; and (iii) OFM tackles challenge C3 by ensuring integer allocation.

We extend the stochastic dataset based on Google AdX data used by Bateni *et al.* [18] for evaluation. We evaluate OFM is in two scenarios, namely a fixed and varying resources. For fixed scenario, we extend the dataset by distributing buyer utilities normally and uniformly. The experimental results clearly show the convergence of OFM with a prediction accuracy of about 95% of optimal prices on datasets namely AdX, normal and uniformly distributed data.

For varying resource scenario, we generate the number of resources offered at each instant by randomly sampling CPU demand without replacement from the Google cluster data trace [22]. We employ prediction schemes to predict the resources offered at the current instant. We use time series models, mean of the previous resource offered and resource offered in the previous instant. Our evaluation shows the superiority of the ARIMA (Auto-Regressive Integrated and Moving Average) model over others regarding prediction accuracy. However, merely using information of last offered resources can reduce computational time without significantly affecting the overall performance.

Further, we perform experiments to determine the impact of buyers on OFM. The results clearly show that OFM computation time is in the order of *microseconds* for a large number of buyers. Hence, OFM is a candidate for the edge computing market where there is a need for quick and irrevocable price computations.

The remainder of this paper is structured as follows. We review related work in Section II. In Section III, we formally describe the OFM problem and the adversarial model. We introduce the OFM algorithm in Sections IV. In Section V, we evaluate OFM and conclude the paper in Section VI.

II. Related work

Cloud resource allocation is naturally an *online* decision-making problem [1], [23]. There are several works on cloud online pricing. Zhang *et al.* [24] propose an online auction mechanism for VMs based on the primal-dual framework. Zhang *et al.* [1] compute posted prices for cloud resources by designing exponential pricing function based on the primal-dual framework for 0-1 Knapsack problem. Zhou *et al.* [25] design an online auction-based the primal-dual framework for cloud jobs with deadlines.

Xilouris *et al.* [26] propose T-Nova, a marketplace for offering VNF (virtual network function) to customers as-a-Service. D'Oro *et al.* [27] implement service chain composition based on a marketplace approach. The servers behave as buyers and network request brokers act as customers to perform service composition at a minimum price. Zhang *et al.* [28] propose an online auction-based marketplace for VNF service chains. A deterministic fraction program is derived from the online stochastic social welfare and allocation is performed using the primal-dual method. Further, prices are learned based on historical bids and arrival and departure of bidders. It is assumed that bidders are strategic. However, these approaches are utility maximizing and hence not fair as they are biased towards the customers with higher utility. All these works compute prices based on the current arrival. In contrast, OFM computes prices before the data or utilities are revealed.

Additional works have investigated online algorithms for market clearing prices. Angelopulos *et al.* [29] propose deterministic and randomized algorithms for finding an approximate market equilibrium for a linear fisher market in an online setting. Blum *et al.* [30] propose a market clearing algorithm for customer bids. The auctioneer has to decide whether to accept or decline a bid without knowledge of future bids. Bateni *et al.* [18] perform multiobjective optimization to dynamically allocate goods appearing online to budgeted buyers of a Fisher market. Azar *et al.* [17] design a primal-dual convex programming based algorithm for an online Fisher market where goods appear in each round with irrevocable decision. In OFM, the buyers and goods change every round unlike [29], and its random permutation model is stronger than the stochastic model used in [18]. Finally, unlike [17], [18], OFM guarantees integer allocation.

III. Problem statement

A. Scenario

The cloud users arrive at the cloud service providers with demand vectors based on the resource types. For instance, in case of infrastructure resources, the demand vectors are CPU and memory requirements. We assume that these demand vectors are mapped to appropriate instances trivially. We illustrate this mapping with a hypothetical numerical example. Let us assume that the customer arrives with a demand vector $< 2, 7.5 >$ of CPU and memory requirements. If Amazon EC2 instances are offered, then the demand vector is mapped to an m3.large instance. This procedure can be extended for services as well. We assume that the cloud provider offers end-to-end service to the user which is common in real scenarios. For instance, Amazon and Google offer complex services such as an analytical engine to the users which are entirely built from their infrastructure. Market-based pricing with end-to-end granularity offers the following advantages: (i) Since infrastructure and application instances are priced differently, the current cloud market is complex [31]. If the service providers offer an end-to-end service consisting of all the components, this has the potential to reduce the cloud market complexity; (ii) In the Cloud, maximizing social welfare improves not only the overall system efficiency but also assures better user experience [1]. Additionally, as discussed before, in the context of cloud computing, NSW achieves a balance between *efficiency* and *fairness* [11]. Further, NSW is *scale-free* — optimal allocation is independent of the scale of each customer's utility. Consequently, there is no incentive to inflate or deflate utility.

The goal of this work is to build an *online market* that maximizes NSW. In this section, we formalize the OFM problem and corresponding adversarial model.

B. Definitions

Let $\mathcal{N} = \{1, 2, \ldots, n\}$ be a set of n buyers indexed by i, i.e., i represents the i^{th} buyer. Let $\mathcal{R} = \{1, 2, \ldots, m\}$ be the set of resources indexed by j, i.e., j represents the j^{th} resource. Let x_{ij}

be the fraction of allocation of the j^{th} resource to the i^{th} buyer. Let u_{ij} be the utility derived by an i^{th} buyer for j^{th} resource and $u_i = \sum_{j=1}^{m} u_{ij} \cdot x_{ij}$ be the total utility derived by the i^{th} buyer. Also, let b_i be the budget of the i^{th} buyer,i.e., total endowment or money of the buyer i. We assume consistent with [14]: (i) The supply of resources are limited; (ii) There is at least one buyer for all the resources, i.e., $u_{ij} > 0, \forall j \in \mathcal{G}$. The prices are determined such that all instances are sold as long as there are enough buyers, i.e., if required, prices are fixed in such a way that even buyers with less money are satisfied if there is no other buyer with more money available for a resource.

OFM considers resources to be indivisible to address the challenge C3 which implies that the allocation of a resource is either 0 or 1. If we take the logarithm, then the maximization of NSW reduces to an Eisenberg-Gale or Fisher market [12]. The convex program is as follows:

$$\begin{aligned} \text{Maximize} & \sum_{i=1}^{n} b_i \log u_i \\ \text{s.t } u_i &= \sum_{j=1}^{m} u_{ij} \cdot x_{ij} \\ & \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \leq 1 \\ & x_{ij} \in \{0,1\}, \forall i \in \mathcal{N}, j \in \mathcal{R}. \end{aligned} \tag{1}$$

To address the challenge C2, we need to design an online algorithm. The online convex optimization algorithms require *Lipschitz continuous* (the rate of change of the function is constant) objective function. However, the objective function of Eq. (1) is not only *non-Lipschitz continuous* but also *non-convex*. Even if we circumvent *non-Lipschitz* continuity by shifting the valuations to a range $\{1, \cdots, K+1\}$, where a number $K > 0$ then we will end up as a linear factor in regret bound leading to low performance (around less 20%) [32]. Further, the above formulation does not *guarantee integer allocation*.

Cole *et al.* [16] provide an alternative convex program equivalent to Eq. (1) as follows:

$$\begin{aligned} \text{Maximize} & \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} \log u_{ij} - p_j \log p_j) \\ \text{s.t } \forall j, & \sum_{i=1}^{n} x_{ij} = p_j \\ \forall i, & \sum_{j=1}^{m} x_{ij} = b_i \\ & \forall i, j, p_j \leq 1, x_{ij} \in \{0, p_j\}. \end{aligned} \tag{2}$$

In the above formulation, p_j is the price associated with resource type j. Also, x_{ij} is the amount paid by the buyer i for the j^{th} resource. The first constraint implies that the total amount paid for a resource by all buyers never exceeds the resource price. The second constraint guarantees that the total amount paid by the buyer is within his overall budget b_i. The third constraint implies that the buyer will either pay the full price or nothing. The constraint $x_{ij} \in \{0, p_j\}$ and $\sum_{i=1}^{n} x_{ij} = p_j$ implies that there can be only one buyer among others with $x_{ij} = p_j$, while for the rest of buyers $i' \neq i$, $x_{i'j} = 0$. Substituting, we get, $\sum_{j=1}^{m} x_{ij} \log u_{ij} - x_{ij} \log x_{ij}$ and the following convex program:

$$\begin{aligned} \text{Maximize} & \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} \log u_{ij} - x_{ij} \log x_{ij}) \\ \text{s.t } \forall i, & \sum_{j=1}^{m} x_{ij} = b_i \end{aligned} \tag{3}$$

The objective function of the above convex program *guarantee integer allocation*. However, it is not only *non-Lipschitz continuous* and but also concave. We perform following steps before applying online convex optimization methods: (i) As we know, a concave function can be converted to convex function by flipping the sign of the function. i.e., $\max g(x)$ and $\min -g(x)$ are equivalent. Hence, the goal is modified to minimization. Informally, we perform optimization in the opposite direction of the objective. (ii) In a Fisher market, equilibrium prices form a unit simplex, i.e., the sum of normalized prices of all goods is equal to 1 [15]. Furthermore, the unit simplex not only reduces computational complexity but also implicitly enforces the constraint of Eq. (3) without affecting the optimality. Hence, we address the *non-Lipschitz continuous* nature of the program by restricting the input set to a unit simplex. Finally, the convex program for OFM is as follows:

$$\begin{aligned} \text{Minimize} & \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{ij} \log x_{ij} - x_{ij} \log u_{ij}) \\ \text{s.t } \forall i, & \sum_{j=1}^{m} x_{ij} = b_i \\ & \forall i \in \mathcal{N}, j \in \mathcal{R}, x_{ij} \geq 0. \end{aligned} \tag{4}$$

The above formulation is the minimization of the *Kullback-Leiber (KL) divergence* [33] between price and the buyer's utility [16]. In summary, Eq. (4) addresses challenges C2 and C3 described in section I. In the subsequent subsection, we develop an adversarial model for tackling challenge C1.

C. OFM Adversarial Model

In literature, adversarial models are proposed to analyze the performance of an online algorithm. In other words, an adversarial model describes the nature of input for the algorithm. Most of the proposed online algorithms are pessimistically designed towards a *fully adversary* model where the input data is provided for the worst case. For instance, the fully adversarial model would provide already sorted numbers to an online quicksort algorithm. Further, the fully adversarial model is *application agnostic*.

In real scenarios, inputs are not adversary always [19], [34]. Moreover, generally, input posses application-specific features and are ignored by fully adversarial models [19]. Random permutation model is widely used to model application-specific features. For instance, the random permutation model captures the tail behavior in sponsored search auction. In this model, an adversary picks the input randomly. Finally, the permutated input is present uniformly to the algorithm. Further, the random permutation model is *sampling without replacement*, while independent and identically distributed (i.i.d) data with known and unknown distributions are *sampling with replacement* [34]. In the former case, samples drawn are independent of each other while they are dependent in the latter. Hence, the random permutation model is more generic compared to (i.i.d). Importantly, any algorithm that works on a random permutation model will work for i.i.d. models [35] and provide better performance for arbitrary input [36]. Finally, it enables us to model customers and resource dynamics over a period. Additionally, it is more generic

compared to stochastic models and simultaneously less pessimistic compared to *fully adversarial* models. Hence, we use the random permutation adversarial model for online fisher market proposed in [36] to address the challenge C1.

Formally, we define our random permutation model as follows [36]: The adversary picks an input consisting of $n = |\mathcal{N}|$, $m = |\mathcal{R}|$, $m_i, \forall i \in \mathcal{N}$ and $d_{ij} \forall i \in \mathcal{N}, j \in \mathcal{R}$. A permutation π of $\mathcal{G}$ is chosen uniformly at random. In round t, the buyer's utility inputs are $u_{ij}^t = d_{i\pi(j)}$ and the budget of the buyer i is $b_i = e_{\pi(i)}$.

Hazan *et al.* [37] define *regret* as the difference between the total cost of the current decision and the best single decision with the benefit of hindsight. Informally, regret is the performance measure between an online player and a static player with hindsight information. Let ℓ_t be the instance of Eq. (4) at instant t or in other words the value of the *loss function* at instant t. Formally, We denote the cumulative regret of the objective function as follows [38]:

$$\boldsymbol{R^o} = \sum_{t=1}^{T} \ell_t(x_t) - \min_x \ell_t(x_t) \tag{5}$$

In summary, the goal of OFM is to find prices $p_j, \forall j \in \mathcal{R}$ at every time instance t for varying buyers and resources that not only *maximize Nash social welfare* but also *Minimize regret* $\boldsymbol{R^o}$.

IV. OFM Algorithm

The goal of OFM is to find *market-clearing* or *equilibrium* prices. Generally, the computation of equilibrium prices for an offline Fisher market using convex optimization solvers is not scalable [39]. Even finding approximate equilibrium prices is non-trivial [13], [40] and especially true in online scenarios with the one-time decision and varying demand and supply as well. Thus, achieving the OFM goal is non-trivial and very challenging.

At every time instant t, OFM offers resources that are either *fixed* or *varying*. In a fixed resource set, the number of offered resources are constant while it is varying every time instance t in varying resource set. In a fixed resource set, the prices depend solely on the buyer utility at that time instant t. In this case, a closed expression (formulae) based algorithm is not only computationally efficient but also scales with the number of buyers. We can obtain a closed-form expression for the OFM objective in this scenario for finding market equilibrium prices.

Theorem IV.1. *The closed form expression for computing prices of the OFM objective function with m fixed resources is given by* $x_{ij} = \frac{b_i u_{ij}}{\sum_{j=1}^{m} u_{ij}}$.

Proof. Let α_i be the dual variable associated with each constraint in equation (4). Now we derive the closed form expression for x_{ij} which minimizes the Eq. (4)

$$\mathcal{L}(x, \alpha) = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} \log x_{ij} - x_{ij} \log u_{ij} + \sum_{i=1}^{n} \alpha_i \big(b_i - \sum_{j=1}^{m} x_{ij}\big)$$

Let $g(\alpha) = \sup_{x \in \Delta} \mathcal{L}(x, \alpha)$ be the dual function. Let $\mu_{ij} = \log u_{ij} + \alpha_i$. At optimality, $\nabla_x \mathcal{L}(x, \alpha) = 0$, substituting and eliminating x, we have the dual of (3):

$$\begin{aligned} \text{Maximize} \sum_{i=1}^{n} b_i \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{m} u_{ij} e^{\mu_{ij}-1} \\ \text{s.t } \forall i \in \mathcal{N}, j \in \mathcal{R}, \mu_{ij} - \log u_{ij} \leq \alpha_i, \\ \forall i \in \mathcal{N}, \alpha_i \geq 0. \end{aligned} \tag{6}$$

At optimality,i.e., $x_{ij} = \frac{b_i u_{ij}}{\sum_{j=1}^{m} u_{ij}}$ then $\alpha_i = 1$. The value of x_{ij} is same for the maximization of Eq. (2) and readers can see at [16]. □

The closed form expression of Theorem IV.1 can be used to compute prices for fixed resource set. Conversely, the varying resource set is not only more generic but also challenging compared to the fixed set scenario. Cloud subletting is a use case for the varying set of resources. In cloud subletting, the users can monetize their allocated resources by subletting to other users [41]. Usually, the service provider acts as a broker. At every instant, interested users can submit their allocated resources to the service provider for subletting for a specified period. Hence, in cloud subletting, the resources offered vary every time instance.

In OFM, the input is revealed only after the current prices are computed. However, OFM has access to the past data, and there will be patterns and trends in supply and demand, e.g., periodical changes from low to high demand and vice versa. Time series analysis is widely used to prediction in such scenarios [42]. OFM updates the time series model based on the input data and performs a prediction of the number of resources offered in the next time instance. Here, OFM can employ different prediction models such as ARIMA, but also more straightforward approaches such as simple moving averages, immediate past values, etc. Once OFM has predicted the number of resources, then the scenario is similar to a fixed resource set and OFM finds the new prices near to the previous instance optimal prices. In this way, OFM achieves its goal.

We introduce the notation and definitions used by the OFM algorithm. As we know previously, the equilibrium prices form a unit simplex [15]. Let Δ_t be the unit simplex constructed from the set $\mathcal{X}$ at time t. Formally, $\Delta_t = \{x | \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} = 1\}$. The cardinality of Δ_t changes only when there is a change in the number of resources offered during previous instant i.e., $|\Delta_t| \neq |\Delta_{t-1}|$ if $m_t \neq m_{t-1}$. Let Δ be the set of all unit simplex till instant T. Formally, $\Delta = \{\Delta_t | t \leq T\}$.

Let $\boldsymbol{x^t}$ be the price matrix (column vector of size $m \times n$) which is the price buyers pay for the resources at round t, i.e., $\boldsymbol{x^t} = (x_{ij}^t, \forall x_{ij} \in \Delta_t)$. Let m_t be the number of resources offered at time t and assume that k past values are available, i.e., $m_{t-k}, m_{t-k-1}, \ldots, m_t$.

A. OFM Algorithm

Online mirror descent (OMD) is one of the widely used algorithms for online convex optimization [43]–[45]. The basic idea of OMD is to perform an update on the dual space of the regularizing function and to project on the convex decision set using appropriate distance generating functions iteratively. The regularization function not only improves the stability but also lowers the regret bounds [43]. OMD with appropriate distance generating function achieves nearly optimal regret for any convex online learning problem with a full adversarial input model. Hence, OMD is considered as *universal* [45]. Further, OMD is a first order method (involves only the slope of a function). Hence, most often updates are not only simple but also computationally efficient.

To apply OMD, we require a distance generating function based on the geometry of the objective function. The *Bregman divergence* [46] is one of the widely used distance generating functions, and it is the distance between the function and its first order Taylor expansion (tangent). Formally, let $h : \Omega \times \text{relint}(\Omega)$ be a continuously differentiable convex function and let $\boldsymbol{p}$ and $\boldsymbol{q}$ be the two points

on h with gradient $\nabla \boldsymbol{p}$ and $\nabla \boldsymbol{q}$ respectively. Then, the Bregman divergence is defined as follows:

$$B_h(\boldsymbol{p}, \boldsymbol{q}) = h(\boldsymbol{p}) - h(\boldsymbol{q}) - \langle \nabla \boldsymbol{q}, \boldsymbol{p} - \boldsymbol{q} \rangle \tag{7}$$

For a Fisher market, an unnormalized negative entropy function is used as the regularization function [47]. Hence, OFM uses unnormalized entropy as the regularization function, i.e., , $h(x) = x \log x - x$ and the corresponding Bregman divergence expression can be found in Appendix VI-A. The pseudocode of OFM is presented in Algorithm 1. OFM works in two stages. The prices are predicted in the first stage and the objective function is updated in the later stage. Let m_0 be the resources offered at time $t = 0$. Initially, OFM determines the number of resources randomly and the corresponding unit simplex and prices for all resources are initialized to $\frac{1}{m_0}$ to satisfy the unit simplex property of equilibrium prices ($\sum_{j=1}^{m_0} p_j = 1$).

For every instant t, Initially, OFM predicts the number of resources (line 5). If the number of resources offered is not the same as in the previous instant $t-1$, then either new resources are added or some existing resources are removed. The removal of a resource is straightforward and involves only updating the length of the set Δ (line 12). Moreover, it does not violate budgets. Let δ be the difference between the number of resources predicted during t and the actual number of resources offered in $t-1$, i.e., $\delta = m'_t - m_{t-1}$. Let σ_{t-1} be the sum of all prices at instance $t-1$. OFM performs the following during the addition of new resources ($\delta > 0$):

- Compute the new prices for m_t using OMD update. These resources are old resources.
- Compute the difference between the sum of the current prices of old resources and σ_{t-1}. The new prices are initialized with the value $\frac{\sum_{j=1}^{m_{t-1}} x_j - \sigma_{t-1}}{\delta \cdot t}$ and if new resources are introduced later, then they should be initialized with low prices to satisfy the unit simplex property in both the scenarios. Otherwise, will lead to a budget violation. Therefore, it is necessary to penalize the offering of new resources at a later instant.

OFM computes the difference and updates the length of the current price vector x_t(line 6). The prices are predicted before the input is revealed (line 13) using Algorithm 2. Once prices are predicted, the function ℓ_t along with the input parameters m, n and u_{ij} are revealed to OFM. These parameters are used to update the online mirror descent of the OFM algorithm. In addition, we compute the optimal prices for the current time instant using the closed-form expression provided in Theorem IV.1(line 15).

The pseudocode for online price prediction is presented in Algorithm 2. The main function of the Algorithm 2 is to predict the current prices based on the previous time instant t. In OMD, the price prediction is done in two stages. They are:

- In a first stage, the update is performed on the regularization function $h(x)$. Let y_t be the update for h at instance t. In OFM, the update rule is given by $\nabla h(y_{t+1}) = \nabla h(x_t) - \eta \nabla_t$, where ∇_t is the gradient of ℓ_t.
- Once y_{t+1} is calculated, then the prediction at time $t+1$, i.e., , x_{t+1} is the projection on Δ which minimizes Bregman divergence between points in Δ and y_{t+1} on the function h. Formally, this can be written as the following optimization problem:

$$\min_{x \in \Delta} B_h(x, y_{t+1}) \tag{8}$$

Eq. (8) is minimized if $\nabla B_h(x, y_{t+1}) = 0$. This implies that $x_{t+1} = y_{t+1}$. The proof can be found in section VI-A.

Algorithm 1 OFM algorithm

Require: ℓ_t, m_t, n_t
1: $m_0 \leftarrow random()$
2: $\forall j \in m_0, x_0 = \frac{1}{m_0}$
3: $\Delta = \Delta_{m_0}$
4: **for** $t \leftarrow 1, T$ **do**
5: $\quad m'_t \leftarrow predictResources(predict, t)$
6: $\quad$ **if** $m'_t \neq m_{t-1}$ **then**
7: $\quad\quad \delta \leftarrow m'_t - m_{t-1}$
8: $\quad\quad$ **if** $\delta > 0$ **then**
9: $\quad\quad\quad \forall k \in \delta, x_t[k] \leftarrow \frac{\sum_{j=1}^{m_{t-1}} x_j - \sigma_{t-1}}{\delta \cdot t}$
10: $\quad\quad\quad$ Set $|x_t| \leftarrow \max\{m_{t-1}, m'_t\}$
11: $\quad$ **else**
12: $\quad\quad$ Set $|x_t| \leftarrow \min\{m_{t-1}, m'_t\}$
13: $\quad x_t \leftarrow OFM - MD(predict, t)$
14: $\quad$ Observe ℓ_t
15: $\quad x_t \leftarrow \arg \ell_t(x^*)$
16: $\quad$ **if** $m_{t-1} \neq m_t$ **then** $\Delta \leftarrow \Delta \cup \{\Delta_{m_t}\}$
17: $\quad$ onlinePrice($update, \ell_t, x_t$)
18: $\quad$ predictResources($update, m_t$)

Algorithm 2 onlinePrice algorithm

1: **procedure** ONLINEPRICE($state, f', x'$)
2: $\quad x_0 \leftarrow \frac{1}{e}$
3: $\quad$ **for** $t \leftarrow 1, T$ **do**
4: $\quad\quad$ **if** $state == predict$ **then return** x_t
5: $\quad\quad$ **if** $state == update$ **then**
6: $\quad\quad\quad \ell_t \leftarrow \ell'$
7: $\quad\quad\quad x_t \leftarrow x'$
8: $\quad\quad\quad \nabla_t = \nabla \ell_t(x_t)$
9: $\quad\quad\quad x_{t+1} = e^{(\log x_t - \eta \nabla_t)}$

The structure of OFM objective function f is time invariant. Hence, we get a closed-form expression for updates and can be used to predict the prices for the next instance instantly as soon as the input is revealed in Algorithm 2(line= 5). The details can be found in the Section VI-B.

In summary, OFM performs the computation of market equilibrium prices for the resources and prediction of resources offered at every instant. The computation of optimal prices can be done in $m \cdot n$ steps. Also, the minimum prices can be found in n steps. Computing the slope using previous prices and current price prediction require n steps each. Hence, the time complexity for every round is $\mathcal{O}(m \cdot n)$. The total time complexity for T instants is given by $\mathcal{O}(T \cdot (\boldsymbol{m} \cdot \boldsymbol{n}))$ where $\boldsymbol{m} = \max m_t, t < T$ and $\boldsymbol{n} = \max n_t, t < T$.

Theorem IV.2. *The regret bound of OFM is* $\boldsymbol{R^o} \leq 2\sqrt{2T \log \boldsymbol{n}}$.

Proof. In OFM, Δ is a simplex and the sum of offline optimal prices (equilibrium prices) never exceeds 1 [15]. Hence, the slope of OFM is bounded even though it is logarithmic. Let $\|\nabla_t\|$ be the dual norm of the slope at instant t and for Δ, $\|\nabla_t\| \leq 1$. Let $\boldsymbol{n} = \max n_t, t < T$ be the maximum number of resources offered in OFM. By substituting these values in [43, section 5.4], we get the regret bound $\boldsymbol{R^o} \leq 2\sqrt{2T \log \boldsymbol{n}}$. □

V. OFM Evaluation

A. Experimental setup

Resource pricing is of strategic importance for any service provider. Hence, the internal pricing mechanisms are not made public. We believe that there is no openly available cloud pricing data of service providers. Balserio *et al.* [48] generate stochastic dataset based on Google AdX real data to evaluate their AdX placement algorithms. Bateni *et al.* [18] extend this dataset by augmenting volatile information to model the sensitivity of shocks due to social and news trends such as negative publicity in the news about the resources.

Each dataset consists of varying advertisers (6 to 101) and impression types (7 to 406). The number of advertisers, impressions and advertisers' utilities are different in each dataset. Arrivals are assumed to be an Ornstein-Uhlenbeck process. An Ornstein-Uhlenbeck is a diffusion process for modeling the velocity of a particle in Brownian motion and used extensively in mathematical finance to model market prices and volatility. The authors estimate the parameters for the dataset presented in [48] and generate synthetic arrival data without affecting the statistical properties of the real dataset. The estimation methodology can be found in [18]. We evaluate both *fixed* and *varying* set of resources. We measure the regret of OFM, i.e., the distance between our OFM online algorithm objective without hindsight and an optimal algorithm with hindsight.

1) Fixed resource set: It is evident from the Theorem IV.1 that the prices depend only on utility and budget of buyers. However, buyer utilities in modified AdX dataset [18] are limited to an Ornstein-Uhlenbeck process. Hence, we perform the following steps on [18] dataset for evaluating OFM: (i) Let Λ_j^t be the Ornstein-Uhlenbeck arrival rate of the resource j at time interval t. We treat mean values of impression type j and advertiser i as a base utility u_{ij}^* and generate a new utility u_{ij} for a resource and a buyer at every instant as a product of base utility and Ornstein-Uhlenbeck arrival rate of the resource, i.e., $u_{ij}^t = u_{ij}^* \Lambda_j^t$. In this way, we ensure that the volatility of buyer utilities in every instant t. In the OFM random permutation model, the expectation of the data varies at each time interval. Hence, maintaining volatility captures the random permutation scenario in the evaluation; (ii) In a real marketplace, buyers arrive with different budgets and it is part of our formulation. We calculate the budget along the similar lines of [18]. In our case the budget of buyer i is calculate as $b_i = \frac{\sum_j u_{ij}}{|\{j : u_{ij} > 0\}|}$; (iii) The challenge C2 implies customers with varying utility distribution. Hence, we use different distributions for generating utilities, namely uniform and normal distributions. A uniform distribution is a simple and widely used distribution, and the utility is generated in the interval $[0, 1]$ uniformly. According to the central limit theorem, non-heavy tailed distribution over a period will converge to a normal distribution [49]. Hence, evaluation on a normal distribution guarantees similar behavior as in other non-heavy-tailed distributions. In summary, the AdX dataset is modified to evaluate OFM for buyers with different probability distributions for a fixed set of resources scenario.

2) Varying resource set: In the case of varying resource set, the OFM solution set is also modified frequently. The current input for OFM is revealed after the price prediction. Hence, evaluating varying resource set is non-trivial. As we discussed before, the varying resource set scenario is typical in cloud subletting. The AdX dataset [18] is not suitable due to fixed buyers and resources. Therefore, we perform the trace-based simulation for OFM evaluation in this scenario.

We use a Google cluster data trace [22] of around $12.5k$ machines for 29 days in a Google data center. Each job has different CPU requirement and hence, different VMs are allocated which eventually leads to different CPU usage, and CPU demand. In other words, the CPU demand is not uniform for all time intervals and depends on the demands of the incoming jobs. We use this demand information to simulate the demand behavior of OFM resources. We generate the number of resources offered at each time instance by randomly sampling CPU demand without replacement from the $41GB$ data set as random permutation model is a sampling without replacement. Furthermore, we assume that service providers introduce more resources during high demand time periods. We perform time series analysis and use Box Jenkin's method to build an ARIMA model to predict the CPU usage for future prediction. OFM uses this prediction information to set the prices for the current time instance. The results of OFM evaluation for both the scenarios are presented in the subsequent subsection.

B. Results

1) Predicting the number of offered resources: We used three time series models namely AR (auto-regressive), MA (moving-average) and ARIMA (auto-regressive moving average). In the AR model, the output is regressed from the previous values. Similarly, MR model regress output from the residual of the previous values. ARIMA combines both AR and MR. In other words, ARIMA forecasts the current output by taking previous values and residuals into account. Apart from time series models, we perform alternative prediction schemes which are computationally straightforward. In the first method, the current prediction is the mean of all the previous values. In the second method, the current prediction is the immediate past value. We call this approach as *immediate previous*. The result of time series modeling and additional approaches of the randomly sampled CPU demand is presented in the figure 1. We tested the series for non-stationary of CPU usage using Dickey-Fuller test (test for finding stochastic process affecting time series statistical properties). The sampled series is stationary with 99% confidence level. We determined the order (number of past data in time series) and moving average statistically using autocorrelation plots. The mean absolute errors of the prediction approaches are presented in the Table I.

It is evident from the Table I that ARIMA outperforms other approaches. However, the immediate previous approach is not only computationally simpler than the rest of the approaches but also closer to ARIMA forecast. For time-sensitive applications, the immediate previous approach is an ideal candidate for predicting the number of offered resources.

2) Fixed Resource set: The experiment is performed for a total instant $T = 500$ on both AdX dataset (Ornstein-Uhlenbeck process), and AdX augmented with both uniform and normally distributed utilities. We performed normal distribution fit on the collected CPU demand trace, and the estimated parameters are used to generate normally distributed utility. The regret for AdX dataset can be found in figure 2a, while Figure 2b and 2c represent regret for uniformly distributed and normally distributed data respectively. The convergence of OFM over a period is easily evident from all the figures. Hence, the regret is reduced with time. The prediction is about 95% of optimal prices on all three datasets AdX, normal and uniformly distributed data.

There are some cases (for instance in Figure 2a AdX 4) where the regret is negative. In such cases, the value of the predicted objective of OFM is greater than the optimal objection due to constraint violation of Eq. (4). In other words, the predicted prices would result in a

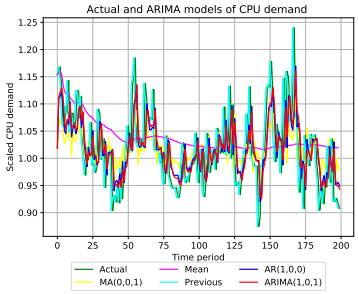
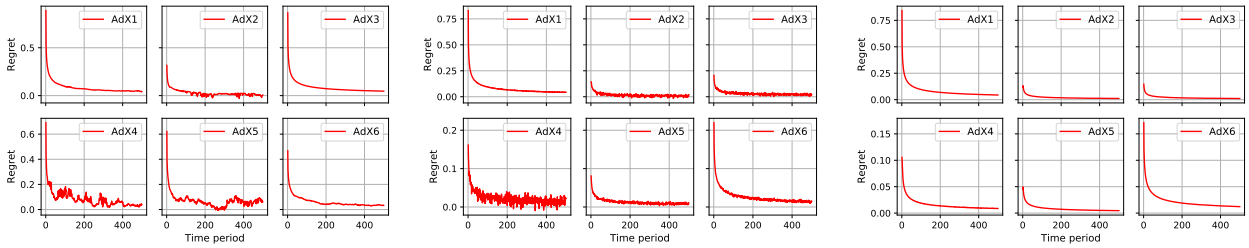


Fig. 1: Comparison of prediction based on ARIMA, mean and previous values of sampled CPU usage of Google cluster trace

Approach	MAE
ARIMA(1,0,1)	0.038524
AR(1,0)	0.038602
MA(0,1)	0.044700
Mean of previous	0.053589
Immediate previous	0.039389

TABLE I: Mean absolute error (MAE) for ARIMA, mean and previous values of sampled CPU usage of Google cluster trace.



(a) Real Adx Dataset (b) Uniformly distributed (c) Normally distributed

Fig. 2: Regret for fixed resource set

budget violation. As soon as the input is revealed, OFM corrects itself in the next instant as evident in the figures due to the absence of successive violations. The number of violations is negligible on both AdX data set and the normally distributed set. The maximum number of violations observed is 7.8% for the uniformly distributed data. In the uniform distribution, all kind of input data (best case, average case, worst case) appear with equal probability. Hence, we find the higher violation when the data is uniformly distributed.

3) Varying Resource set: The Figure 3 shows the regret for uniform and normally distributed data for three prediction approaches namely ARIMA, immediate previous and mean. In this scenario, regret is not smooth and vary unlike the fixed set of resources. This is due to the difference between the actual and predicted resource offered. We see a sudden decrease in regret, when the actual values are decreased suddenly, i.e, a smaller value in an increasing sequence.

4) Buyer Scalability: We evaluate the OFM scalability to handle a large number of buyers. We measure the time taken by OFM to compute prices using the *timeit* function in Matlab. *timeit* calls a function multiple times and return the median of actual time taken. As we know, most of the functions are vectorized for performance improvement. Hence, we fix the number of resources offered to large number 1000 for the entire period. Initially, we start with 1000 buyers and at every period, we increase the buyers by 1000. At the end of the period $t = 50$, the number of buyers is 50000. We repeat the experiment for resources $2000, 3000, 4000$, and 5000. The Figure 4 shows the time required by OFM to compute price. It is evident from the figure that the computational time is in the order of *microseconds* which implies that OFM is an apt candidate for real-time deployment.

VI. Conclusion

In this work, we proposed OFM for computing prices for Fisher marketplace with integer allocation for varying buyers and resources at every time instance. OFM is an online algorithm and based on stricter adversarial model compared to state of the art solutions. In other words, the prices once computed cannot be altered. Further, prices for next time instant is predicted even before all the inputs are revealed. The experimental evaluation on both real world and emulated dataset demonstrate the low regret bound and faster convergence over the period achieving around 95% of optimal prices. The updates in OFM are closed-form expressions and are computationally efficient as demonstrated in our evaluations. OFM computation time is in the order of *microseconds* even for a large number of buyers. Hence, OFM could be an ideal choice for deploying online marketplace for cloud resources, especially in edge computing.

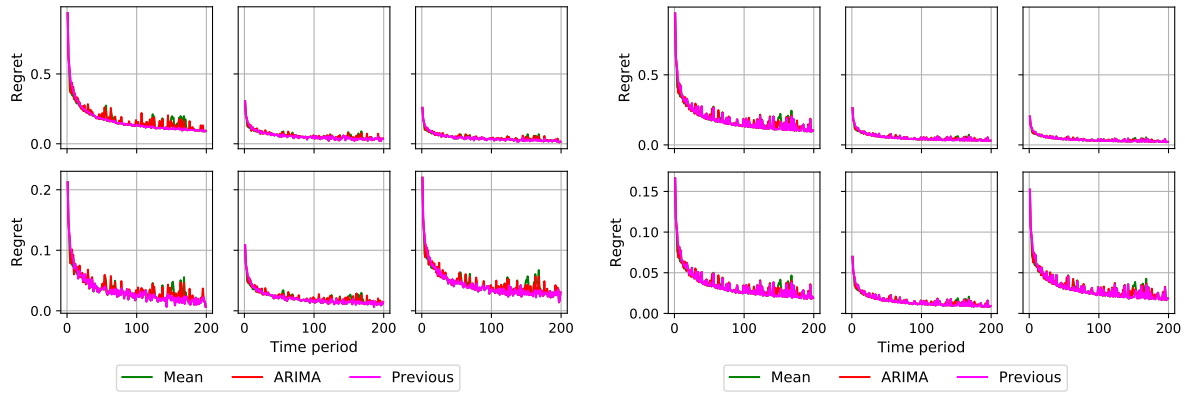
Appendix

A. Bregman divergence

In our case, $h(x) = x \log x - x$. Hence, $\nabla h(y) = \log y$. Bregman divergence $B_h(x, y)$ is given by $B_h(x, y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$. The Bregman divergence is minimized when $\nabla B_h(x, y) = 0$ i.e., $\nabla B_h(x, y) = 1 + \log x - \log y - 1 = 0$ which implies $x = y$. Therefore, Bregman divergence $B_h(x, y)$ of unnormalized entropy function $h(x) = x \log x - x$ is minimized when $x = y$.

B. Online Mirror Descent(OMD)

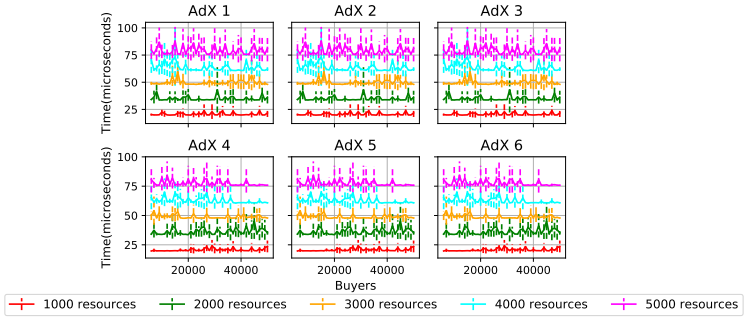
Let $h(x) = x \log x - x$ be the regularization function. We have, $\nabla_{x_{ij}} f(x) = 1 + \log x_{ij} - \log u_{ij}$ and $\nabla_{x_{ij}} h(x) = \log x_{ij}$. In online mirrored descent method, the update is first performed on $h(x)$ and projected back such that projected point minimizes Bregman divergence between objective and regularization function [43]. First we have to find the initial point x_1 such that $x_1 = \arg min B_h(x, y1)$ where $\nabla h(y1) = 0$. If $y1 = \frac{1}{e}$, then $\nabla h(y1) = 0$. Hence, $x1 = y1 = 1$. As per [43], agile version of OMD update is given by $\nabla h(y_{t+1}) = \nabla h(x_t) - \eta \nabla_{f_t}$.



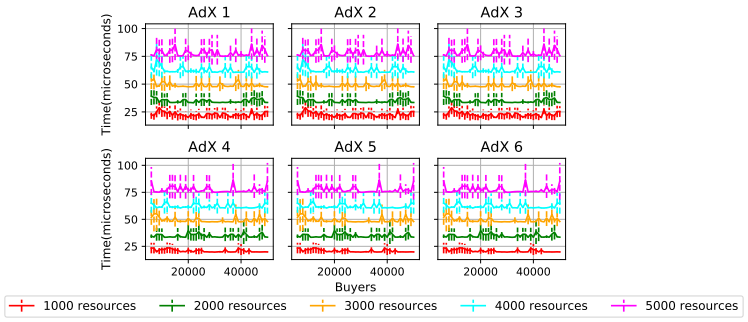
(a) Uniformly distributed

(b) Normally distributed

Fig. 3: Regret for varying resources for ARIMA, immediate previous and mean model



(a) Uniformly distributed



(b) Normally distributed

Fig. 4: Measured time for varying buyers for fixed resources

ACKNOWLEDGMENT

The research leading to these results has received funding from the EU FP7 Marie Curie Actions by the EC Seventh Framework Programme (FP7/2007-2013) Grant Agreement No. 607584 (the Cleansky project). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the CleanSky project or the European Commission.

REFERENCES

[1] Z. Zhang, Z. Li, and C. Wu, "Optimal Posted Prices for Online Cloud Resource Allocation," *Proceedings of the ACM on Measurement and Analysis of Computing Systems* , vol. 1, no. 1, pp. 23:1–23:26, Jun. 2017.

[2] N. C. Luong, P. Wang, D. Niyato, Y. Wen, and Z. Han, "Resource Management in Cloud Networking Using Economic Analysis and Pricing Models: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 954–1001, 2017.

[3] S. Jagannathan and K. C. Almeroth, "Price Issues in Delivering E-content On-demand," *ACM SIGecom Exchanges*, vol. 3, no. 2, pp. 18–27, Mar. 2002.

[4] F. Ridder and A. Bona, "Four Risky Issues When Contracting for Cloud Services," 2011. [Online]. Available: https://www.gartner.com/doc/1543314/risky-issues-contracting-cloud-services

[5] J. Zhao, H. Li, C. Wu, Z. Li, Z. Zhang, and F. C. M. Lau, "Dynamic Pricing and Profit Maximization for the Cloud with Geo-distributed Data Centers," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Apr. 2014, pp. 118–126.

[6] A. S. Prasad and S. Rao, "A Mechanism Design Approach to Resource

Procurement in Cloud Computing," *IEEE Transactions on Computers*, vol. 63, no. 1, pp. 17–30, Jan. 2014.

[7] H. Xu and B. Li, "Dynamic Cloud Pricing for Revenue Maximization," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 158–171, Jul. 2013.

[8] A. Mas-Colell, M. D. Whinston, and J. R. Green, *Microeconomic Theory*. Oxford University Press, 1995.

[9] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[10] I. Menache, A. Ozdaglar, and N. Shimkin, "Socially Optimal Pricing of Cloud Computing Resources," in *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), May 2011, pp. 322–331.

[11] H. Moulin, *Fair Division and Collective Welfare* . The MIT Press, Jan. 2003.

[12] K. Jain and V. V. Vazirani, "Eisenberg–Gale markets: Algorithms and game-theoretic properties," *Games and Economic Behavior*, vol. 70, no. 1, pp. 84–106, Sep. 2010.

[13] N. R. Devanur, C. H. Papadimitriou, A. Saberi, and V. V. Vazirani, "Market Equilibrium via a Primal–Dual Algorithm for a Convex Program," *Journal of the ACM*, vol. 55, no. 5, pp. 22:1–22:18, Nov. 2008.

[14] G. Goel and V. V. Vazirani, "A Perfect Price Discrimination Market Model with Production, and a Rational Convex Program for It," *Mathematics of Operations Research*, vol. 36, no. 4, pp. 762–782, Nov. 2011.

[15] V. I. Shmyrev, "An Algorithm for Finding Equilibrium in the Linear Exchange Model with Fixed Budgets," *Journal of Applied and Industrial Mathematics*, vol. 3, no. 4, pp. 505–518, Dec. 2009.

[16] R. Cole, N. Devanur, V. Gkatzelis, K. Jain, T. Mai, V. V. Vazirani, and S. Yazdanbod, "Convex Program Duality, Fisher Markets, and Nash Social Welfare," in *Proceedings of the 2017 ACM Conference on Economics and Computation*. ACM, Jun. 2017, pp. 459–460.

[17] Y. Azar, N. Buchbinder, and K. Jain, "How to Allocate Goods in an Online Market?" *Algorithmica*, vol. 74, no. 2, pp. 589–601, Feb. 2016.

[18] M. H. Bateni, Y. Chen, D. F. Ciocan, and V. Mirrokni, "Fair Resource Allocation in A Volatile Marketplace," in *Proceedings of the 2016 ACM Conference on Economics and Computation*. ACM Press, Jul. 2016, pp. 819–819.

[19] Gupta, R. and Roughgarden, T., "A PAC Approach to Application-Specific Algorithm Selection," *SIAM Journal on Computing*, vol. 46, no. 3, pp. 992–1017, 2017.

[20] A. N. Toosi, K. Vanmechelen, F. Khodadadi, and R. Buyya, "An Auction Mechanism for Cloud Spot Markets," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11, no. 1, pp. 2:1–2:33, Feb. 2016.

[21] R. Cole and V. Gkatzelis, "Approximating the Nash Social Welfare with Indivisible Items," in *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, Jun. 2015, pp. 371–380.

[22] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[23] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. M. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2060–2073, Aug. 2016.

[24] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online Auctions in IaaS Clouds: Welfare and Profit Maximization With Server Costs," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1034–1047, Apr. 2017.

[25] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An Efficient Cloud Market Mechanism for Computing Jobs With Soft Deadlines," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 793–805, Apr. 2017.

[26] G. Xilouris, E. Trouva, F. Lobillo, J. M. Soares, J. Carapinha, M. J. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro, Y. Rebahi, and A. Kourtis, "T-NOVA: A Marketplace for Virtualized Network Functions," in *2014 European Conference on Networks and Communications (EuCNC)*. IEEE, Jun. 2014, pp. 1–5.

[27] S. D'Oro, S. Palazzo, and G. Schembra, "Orchestrating Softwarized Networks with a Marketplace Approach," *Procedia Computer Science*, vol. 110, pp. 352–360, 2017.

[28] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. M. Lau, "Online Stochastic Buy-Sell Mechanism for VNF Chains in the NFV Market," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 392–406, Feb. 2017.

[29] S. Angelopoulos, A. D. Sarma, A. Magen, and A. Viglas, "On-Line Algorithms for Market Equilibria," in *11th Annual International Conference Computing and Combinatorics (COCOON)*. Springer Berlin Heidelberg, Aug. 2005, pp. 596–607.

[30] A. Blum, T. Sandholm, and M. Zinkevich, "Online Algorithms for Market Clearing," *Journal of the ACM*, vol. 53, no. 5, pp. 845–879, Sep. 2006.

[31] J. Anselmi, D. Ardagna, J. C. S. Lui, A. Wierman, Y. Xu, and Z. Yang, "The Economics of the Cloud," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 2, no. 4, pp. 18:1–18:23, Aug. 2017.

[32] R. Freeman, S. M. Zahedi, and V. Conitzer, "Fair and Efficient Social Choice in Dynamic Settings," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. AAAI Press, Aug. 2017, pp. 4580–4587.

[33] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[34] N. R. Devenur and T. P. Hayes, "The Adwords Problem: Online Keyword Matching with Budgeted Bidders Under Random Permutations," in *Proceedings of the 10th ACM Conference on Electronic Commerce*. ACM, Jul. 2009, pp. 71–78.

[35] S. Agrawal and N. R. Devanur, "Fast Algorithms for Online Stochastic Convex Programming," in *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Jan. 2015, pp. 1405–1424.

[36] N. R. Devanur, "Online Algorithms with Stochastic Input," *ACM SIGecom Exchanges*, vol. 10, no. 2, pp. 40–49, Jun. 2011.

[37] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic Regret Algorithms for Online Convex Optimization," *Machine Learning*, vol. 69, no. 2-3, pp. 169–192, Dec. 2007.

[38] H. Wang and A. Banerjee, "Online alternating direction method (longer version)," *CoRR*, vol. abs/1306.3721, 2013.

[39] B. Codenotti, B. Mccune, S. Pemmaraju, R. Raman, and K. Varadarajan, "An Experimental Study of Different Approaches to Solve the Market Equilibrium Problem," *Journal of Experimental Algorithmics*, vol. 12, pp. 3.3:1–3.3:21, Aug. 2008.

[40] J. B. Orlin, "Improved Algorithms for Computing Fisher's Market Clearing Prices: Computing Fisher's Market Clearing Prices," in *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, Jun. 2010, pp. 291–300.

[41] Y. Zhu, S. Fu, J. Liu, and Y. Cui, "Truthful Online Auction for Cloud Instance Subletting," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Jun. 2017, pp. 2466–2471.

[42] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control (Revised Edition)*. Holden-Day, 1976.

[43] E. Hazan, "Introduction to Online Convex Optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, Aug. 2016.

[44] S. Shalev-Shwartz, "Online Learning and Online Convex Optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, Feb. 2011.

[45] N. Srebro, K. Sridharan, and A. Tewari, "On the Universality of Online Mirror Descent," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*. Neural Information Processing Systems Foundation, Dec. 2011, pp. 2645–2653.

[46] L. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 3, pp. 200–217, 1967.

[47] B. Birnbaum, N. R. Devanur, and L. Xiao, "Distributed Algorithms via Gradient Descent for Fisher Markets," in *Proceedings of the 12th ACM Conference on Electronic Commerce*. ACM, Jun. 2011, pp. 127–136.

[48] S. R. Balseiro, J. Feldman, V. Mirrokni, and S. Muthukrishnan, "Yield Optimization of Display Advertising with Ad Exchange," *Management Science*, vol. 60, no. 12, pp. 2886–2907, Oct. 2014.

[49] C. Bandi, D. Bertsimas, and N. Youssef, "Robust Queueing Theory," *Operations Research*, vol. 63, no. 3, pp. 676–700, Apr. 2015.