Online Job Allocation with Hard Allocation Ratio Requirement

Han Deng
Department of ECE
Texas A&M University
College Station, TX 77840, USA
Email: hdeng@email.tamu.edu

I-Hong Hou
Department of ECE
Texas A&M University
College Station, TX 77840, USA
Email: ihou@tamu.edu

***Abstract*—The problem of allocating jobs to appropriate servers in cloud computing is studied in this paper. We consider that jobs of various types arrive in some unpredictable pattern and the system is required to allocate a certain ratio of jobs. In order to meet the hard allocation ratio requirement in the presence of unknown arrival patterns, one can increase the capacity of servers by expanding the size of data centers. We then aim to find the minimum capacity needed to meet a given allocation ratio requirement. We propose two online job allocation policies with low complexity. We prove that, given a hard allocation ratio requirement, these two policies can achieve the requirement with the least capacity. We also derive a closed-from expression for the amount of capacity needed to achieve any given requirement. Two other popular policies are studied, and we demonstrate that they need at least an order higher capacity to meet the same hard allocation ratio requirement. Simulation results demonstrate that our policies remain far superior than the other two when jobs arrive according to some random process.**

I. Introduction

In this paper, we discuss the problem of online job allocation for cloud computing where each job can only be served by a subset of servers. Such a problem exists in many emerging Internet services, such as YouTube, Netflix, etc. For example, in the case of YouTube, each video is replicated only in a small number of servers, and each server can only serve a limited number of streams simultaneously. When a user accesses YouTube and makes a request to watch a video, this request needs to be allocated to one of the servers that not only stores the video but also has remaining capacity to process this request. If no server can process this request, the request is dropped, which can significantly impact user satisfaction.

There are many studies on obtaining statistics about user behaviors, video popularity, and access locality [1]–[5]. These studies provide important insights about system planning. However, they are not sufficient for job allocation. It is usually difficult to predict which video will go viral and generate most requests. Therefore, online job allocation policy that makes decisions solely based on current system state is needed.

The problem of online job allocation has attracted much attention. Most current studies study the performance of online policies by comparing the number of allocated jobs under online policies against that of the an offline policy with full knowledge about all future arrivals. For example, Karp, Vazirani and Vazirani [6] propose an online policy that is guaranteed $1 - \frac{1}{e} \approx 63\%$ of jobs, given that the offline policy allocates all jobs. They also show that no online policy can guarantee to allocate more jobs than their proposed policy.

These studies are still insufficient for many practical scenarios. In particular, one can argue that most practical applications demand a much higher ratio of allocated jobs than 63%. In order to meet this demand, current practice is to add redundancy and increase the capacity of data centers to accommodate unpredictable patterns of job arrivals. Motivated by this observation, we seek to address the following question: How much capacity is needed to guarantee the allocation of, say, 95% of jobs?

To solve this problem, we first formulate a linear programming problem for job allocation. The uncertainty in future job arrivals corresponds to the uncertainty in some of the parameters. Also, increasing the capacity of the servers corresponds to relaxing some of the constraints in the linear programming problem.

Using this formulation, we propose two simple online scheduling policies and derive closed-form expressions for their performance. Specifically, we prove that, in order to allocate at least $1-\frac{1}{\theta}$ of jobs, the two policies only need to increase the capacity by $\ln\theta$ times. We also prove that no online policy can guarantee to allocate the same ratio of jobs with less capacity, and hence our policies are optimal.

We further evaluate two popular online job allocation policies. Surprisingly, we prove that, to guarantee to allocate at least $1 - \frac{1}{\theta}$, these two policies require at least $\theta - 1$ times capacity. Therefore, they are both an order worse than our policies.

The allocation ratio guarantees stated above need to hold for every sample path. We also consider the scenario where jobs arrivals are generated by some unknown

This material is based upon work supported by, or in part by, the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-15-1-0279.

random process, and one only needs guarantees on the expected value of allocation ratio. By both theoretical analysis and numerical studies, we demonstrate that our policies remain much better than the other two policies for this scenario.

The rest of the paper is organized as follows: Section II reviews some previous work on online learning and load balancing problem. Section III describes our system model and problem formulation. Section IV introduces two online job allocation policies and studies their performance. Section V derives a performance bound for all online policies. Section VI compares our proposed policies with two widely used policies. Section VII provides simulation results of the four policies. Finally, Section VIII concludes this paper.

II. Related Work

There are many studies on YouTube videos about their statistical properties [5], [7], [8]. Studies on online learning further investigate the possibility to predict the future video requests. The problem of "learn from expert advise" was first studied by Littlestone and Warmuth [9], DeSantis, Markowsky, and Wegman [10]. Later "learn from examples" was studied. The Winnow algorithm was proposed and studied by Littlestone and Nicholas [11], [12]. The algorithm applies well to practical tasks such as on World Wide Web [13]. Other sequence prediciton research also include the studies by Nicolo and Gabor [14], Hutter [15].

The online job allocation problem is an online matching procedure which aims to make the best decision on job-server pair to maximize the number of jobs matched. The problem of online bipartite matching was studied by Karp, Vazirani, and Vazirani [6]. They use an adversary model and studied GREEDY which achieves a matching ratio of $1/2$ and RANKING which achieves $1-1/e$. They further showed that no algorithm can achieve a better ratio than $1-1/e$. Other models, which are based on further assumption on arrival pattern, have also been studied. Random arrival model has been studied by Goel and Mehta [16], and Karande, Mehta and Tripathi [17]. They show that GREEDY achieves a matching ratio of $1-1/e$ and RANKING achieves greater than $1-1/e$. Known distribution model was introduced by Feldman, Mehta, Mirrokni, and Muthukrishnan [18]. They provide a two-suggested-matching algorithm which achieves a ratio of 67%. Kalyanasundaram and Pruhs studied the online b-matching problem [19] which can be seen as the job allocation problem with server capacity b. They presented BALANCE algorithm and proved that it approaches $1-1/e$. Applications of online matching to ad-words problem, which is an allocation of bidders to key words within the budget limit of each bidder, have been studied in [16], [20].

III. System Model

We consider a system with multiple non-identical servers. Jobs arrive at the system sequentially. Jobs are of different types, and each job can only be served by a subset of the servers. For example, in the application of on-demand video streaming, a job is a request for one video, and can therefore only be served by servers that possess the video. We assume that when a job enters the system, it needs to be allocated to a server immediately. Jobs that cannot be allocated upon arrivals are discarded from the system. We also assume that jobs cannot be moved once they are allocated to servers, as moving jobs between servers cause additional costs on job migration. A similar model has been used in [21].

We use $\mathcal{J}$ to denote the set of servers, and $\mathcal{I} = \{1, 2, \dots\}$ to denote the arrival sequence of jobs. Each job i can be served by a subset $K_i \subseteq \mathcal{J}$ of servers. Upon its arrival, job i reveals its K_i, and the system either allocates it to a server or discards it. Each job takes one unit of capacity in the server to which the job is allocated to. A server j has a total amount of C_j units capacity, and can therefore at most serve C_j jobs. We use X_{ij} to denote the assignment of the jobs. If $X_{ij} = 1$, then job i is assigned to server j. If $X_{ij} = 0$, job i is not assigned to server j.

We aim to maximize the number of jobs that can be served. The problem of maximizing the number of served jobs can be formulated as the following linear programming problem:

Allocation:

$$Max \sum_{ij} X_{ij} \tag{1}$$

$$s.t. \sum_{i:j\in K_i} X_{ij} \le C_j, \forall j \in \mathcal{J}, \tag{2}$$

$$\sum_{j:j\in K_i} X_{ij} \le 1, \forall i \in \mathcal{I}, \tag{3}$$

$$X_{ij} \ge 0, \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{4}$$

Since $X_{ij} = 1$ if job i is served by j, (1) is the total number of served jobs. On the other hand, (2) states that each server j can at most serve C_j jobs, and (3) states that each job can be served by at most one server. In this formulation, we allow X_{ij} to be any real number between 0 and 1, while X_{ij} needs to be either 0 or 1 according to our model. Therefore, **Allocation** describes an upper-bound of the number of jobs that can be allocated.

Solving **Allocation** is straightforward when one has knowledge of all its parameters $\{K_i\}$ and $\{C_j\}$. However, as jobs arrive sequentially, the system needs to make allocation decisions without knowledge of future jobs. We say that an allocation policy is an *online policy* if it makes allocation decisions only based on jobs that have already arrived. On the other hand, an allocation policy is an *offline policy* if it has full knowledge about all future

job arrivals, and can therefore find the optimal solution to **Allocation**.

We consider that the service provider can increase server capacity to allocate more jobs. When the system capacity is increased by R times, a server j with C_j capacity will have RC_j capacity after the increase. We can now formulate the following linear programming problem:

Allocation(R):

$$Max \sum_{ij} X_{ij} \tag{5}$$

$$s.t. \sum_{i:j\in K_i} X_{ij} \le RC_j, \forall j \in \mathcal{J}, \tag{6}$$

$$\sum_{j:j\in K_i} X_{ij} \le 1, \forall i \in \mathcal{I}, \tag{7}$$

$$X_{ij} \ge 0, \forall i \in \mathcal{I}, j \in \mathcal{J}. \tag{8}$$

We evaluate the performance of online policies by comparing the number of allocated jobs under online policies with R times capacity against that under offline policy with unit capacity. Specifically, given $\{K_i\}$ and $\{C_j\}$, let Γ_{opt} be the optimal value of $\sum_{ij} X_{ij}$ in **Allocation**, and $\Gamma_\eta(R)$ be the value of $\sum_{ij} X_{ij}$ in **Allocation(R)** under policy η. Since a typical server can host a large number of jobs, we consider the case when $\{C_j\}$ is large, and define the *competitive ratio per sample path* as follows:

Definition 1: An online policy η is said to be (R,θ)-*competitive-per-sample-path* if there exists a function $\Theta(C_{min})$ such that, for all $\{K_i\}$ and $\{C_j\}$ with $C_j \ge C_{min}$, $\Gamma_{opt}/\Gamma_\eta(R) \le \Theta(C_{min})$, and $\Theta(C_{min}) \to \theta$ as $C_{min} \to \infty$.

Definition 1 defines competitive ratio based on the worst-case sample path. This definition may ignore effects of statistic multiplexing. In practice, jobs may arrive according to some random process. Therefore, the arrivals of different types of jobs are likely to be interwined.

We can expand our model to accommodate the random nature of job arrivals. Given $\{K_i\}$, we can consider the case where the actual arrival sequence is a random permutation of $\mathcal{I} = \{1, 2, \dots\}$. Let $E[\Gamma_\eta(R)]$ be the expected number of allocated jobs under η with R times capacity when the arrival sequence is a random permutation. We then define the *expected competitive ratio* as follows

Definition 2: An online policy η is said to be (R,θ)-*competitive-in-expectation* if there exists a function $\Theta(C_{min})$ such that, for all systems with $C_j \ge C_{min}$, $\Gamma_{opt}/E[\Gamma_\eta(R)] \le \Theta(C_{min})$, and $\Theta(C_{min}) \to \theta$ as $C_{min} \to \infty$.

It is obvious that the competitive ratio per sample path cannot be better than the expected competitive ratio.

Lemma 1: A (R,θ)-competitive-per-sample-path policy is (R,θ)-competitive-in-expectation.

IV. Two Online Allocation Policies and Their Competitive Ratios

In this section, we propose online policies and analyze their competitive ratios. Our analysis is based on the Weak Duality Theorem of linear programming [22]. The dual problem of **Allocation** can be written as:

Dual:

$$Min \sum_j C_j\alpha_j + \sum_i \beta_i, \tag{9}$$

$$s.t.\ \alpha_j + \beta_i \ge 1, \forall i \in \mathcal{I}, j \in K_i \tag{10}$$

$$\alpha_j \ge 0, \forall j, \tag{11}$$

$$\beta_i \ge 0, \forall i, \tag{12}$$

where each α_j corresponds to a constraint in (2), and each β_i corresponds to a constraint in (3). The following lemma is then a direct result of the Weak Duality Theorem.

Lemma 2: Given any vectors of $\{\alpha_j\}$ and $\{\beta_i\}$ that satisfy the constraints (10)–(12), we have $\sum_j C_j\alpha_j + \sum_i \beta_i \ge \Gamma_{opt}$.

We now introduce an online policy using the duality of linear programming. This policy maintains a variable α_j for each server j. When the system starts, it sets $\alpha_j \equiv 0$. When a job i arrives, the policy checks the values of α_j for all $j \in K_i$, and selects j^* as the one with the minimum value of α_j. If $\alpha_{j^*} < 1$, job i is assigned to server j^*, and therefore $X_{ij^*} = 1$. The value of α_{j^*} is updated to be $\alpha_{j^*}(1 + \frac{1}{C_{j^*}}) + \frac{1}{(d_{j^*}-1)C_{j^*}}$, where we set $d_j = (1+1/C_j)^{RC_j}$, for all j. The value of d_j is chosen to achieve the optimal competitive ratio, as will be shown in the proof of Lemma 3. On the other hand, if $\alpha_{j^*} \ge 1$, job i is discarded. The complete policy is described in Algorithm 1 and we call it Primal-Dual (PD) Algorithm.

Algorithm 1 PD Algorithm

```
Initially, α_j = 0, β_i = 0, X_ij = 0.
d_j ← (1 + 1/C_j)^{RC_j}, ∀j.
for each arriving job i do
    j* ← argmin_{j∈K_i} α_j.
    if α_{j*} < 1 then
        β_i ← 1 − α_{j*}.
        α_{j*} ← α_{j*}(1 + 1/C_{j*}) + 1/((d_{j*} − 1)C_{j*}).
        X_{ij*} ← 1.
        Job i is assigned to server j*.
    else
        Discard job i.
```

We first need to show that the vector $\{X_{ij}\}$ produced by this policy satisfies all constraints of **Allocation(R)**, so that the policy never assigns a job to a server that is already fully utilized.

Lemma 3: Let $\alpha_j[n]$ be the value of α_j after n jobs have been allocated to server j. Then,

$$\alpha_j[n] = (\frac{1}{d_j - 1})(d_j^{n/RC_j} - 1). \tag{13}$$

Proof: Here we prove (13) by induction.

Initially, when $n = 0$, $\alpha_j[0] = 0 = (\frac{1}{d_j-1})(d_j^0 - 1)$ and (13) holds.

Suppose (13) holds when $n = k$. When the $(k+1)$-th job is allocated server j, we have

$$\begin{aligned}\alpha_j[k+1] =& \alpha_j[k](1 + \frac{1}{C_j}) + \frac{1}{(d_j - 1)C_j} \\ =& \frac{1}{(d_j - 1)}(d_j^{k/RC_j} - 1)(1 + \frac{1}{C_j}) + \frac{1}{(d_j - 1)C_j} \\ =& \frac{1}{(d_j - 1)}[d_j^{(k+1)/RC_j} - 1],\end{aligned}$$

and (13) still holds for $n = k+1$. By induction, (13) holds for all n. ∎

With Lemma 3, $\alpha_j = 1$ when RC_j jobs have been allocated to server j. Since Algorithm 1 only allocates jobs to servers with $\alpha_j < 1$, our policy does not violate any constraints in **Allocation(R)**.

Next, we study the competitive ratio of Algorithm 1.

Theorem 1: Algorithm 1 is $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path.

Proof: We prove Theorem 1 by three steps:

First, we show that solutions $\{\alpha_j\}$ and $\{\beta_i\}$ satisfy all constraints in **Dual**. Initially, α_j and β_i are set to be 0. By step 7 in Algorithm 1, α_j is non-decreasing throughout the execution of the policy, and hence (11) holds. Also, by Lemma 3, $\alpha_j \leq 1$, for all j. When a job i arrives, our policy sets $j^* \leftarrow \underset{j \in K_i}{\operatorname{argmin}}\{\alpha_j\}$. If $\alpha_{j^*} = 1$, we have $\alpha_j = 1$ for all $j \in K_i$ and $\beta_i = 0$. Hence, both constraints (10) and (12) hold. On the other hand, if $\alpha_{j^*} < 1$, $\beta_i = 1 - \alpha_{j^*} \geq 1 - \alpha_j$, for all $j \in K_i$. Both constraints (10) and (12) still hold.

Next, we derive the ratio between $\sum_{ij} X_{ij}$ and $\sum_j C_j\alpha_j + \sum_i \beta_i$. Both formulas are initially 0. We now consider the amounts of change of these two formulas when a job i arrives. We use $\Delta P(R)$ to denote the change of $\sum_{ij} X_{ij}$, and ΔD to denote the change of $\sum_j C_j\alpha_j + \sum_i \beta_i$.

If job i is discarded, then $\{X_{ij}\}$, $\{\alpha_j\}$ and $\{\beta_i\}$ remain unchanged, and therefore $\Delta P(R) = \Delta D = 0$.

On the other hand, consider the case when job i is assigned to server j. We have $X_{ij} = 1$ and $\Delta P(R) = 1$. We also have

$$\begin{aligned}\frac{\Delta D}{\Delta P(R)} = \Delta D =& C_j(\frac{\alpha_j}{C_j} + \frac{1}{(d_j - 1)C_j}) + 1 - \alpha_j \\ =& 1 + \frac{1}{d_j - 1} = \frac{d_j}{d_j - 1} \\ =& \frac{(1 + 1/C_j)^{RC_j}}{(1 + 1/C_j)^{RC_j} - 1}.\end{aligned}$$

When we impose a lower bound on C_j by requiring $C_j \geq C_{min}$, for all j, and let $C_{min} \to \infty$, we have $\frac{\Delta D}{\Delta P(R)} \to \frac{e^R}{e^R-1}$, whenever a job i is allocated to some server. Therefore, we have, under Algorithm 1,

$$\frac{\sum_j C_j\alpha_j + \sum_i \beta_i}{\sum_{ij} X_{ij}} = \frac{e^R}{e^R - 1}. \tag{14}$$

Finally, by Lemma 2, we establish that Algorithm 1 is $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path. ∎

Algorithm 1 relies on the usage of artificial variables $\{\alpha_j\}$ and $\{d_j\}$. Below, we introduce a second online policy that not only is simpler, but also conveys better intuition. The policy is called Join-Least-Utilization (JLU) policy. When a job arrives, JLU simply allocates the job to the server with the smallest utilization ratio, which is the number of allocated jobs at a server divided by its capacity. Specifically, let n_j be the number of jobs that have already been allocated to server j. When job i arrives, it is allocated to $\operatorname{argmin}_{j \in K_i} \frac{n_j}{RC_j}$.

The complete policy is described in Algorithm 2. While the algorithm still involves $\{\alpha_j\}$, $\{d_j\}$, and $\{\beta_i\}$, these variables are introduced solely for the purpose of the analysis of competitive ratio. They can be omitted in actual implementation.

Algorithm 2 JLU

1: Initially, $\alpha_j = 0$, $\beta_i = 0$, $X_{ij} = 0$, $n_j = 0$.
2: $d_j \leftarrow (1 + 1/C_j)^{RC_j}, \forall j$.
3: **for** each arriving job i **do**
4: $\quad j^* \leftarrow \operatorname{argmin}_{j \in K_i}\{\frac{n_j}{RC_j}\}$.
5: $\quad$ **if** $n_{j^*} < RC_{j^*}$ **then**
6: $\quad\quad X_{ij^*} \leftarrow 1$.
7: $\quad\quad \alpha_{j^*} \leftarrow \alpha_{j^*}(1 + \frac{1}{C_{j^*}}) + \frac{1}{(d_{j^*} - 1)C_{j^*}}$.
8: $\quad\quad n_{j^*} \leftarrow n_{j^*} + 1$.
9: $\quad\quad$ Job i is assigned to server j^*.
10: $\quad\quad \beta_i \leftarrow 1 - \min_{j \in K_i} \alpha_j$.
11: $\quad$ **else**
12: $\quad\quad$ Discard job i.

Lemma 4: For any $\delta > 0$, there exists a finite C_{min} such that, by requiring $C_j > C_{min}$, for all j, we have

$$\alpha_{j_1} - \alpha_{j_2} > \delta \Longrightarrow \frac{n_{j_1}}{RC_{j_1}} > \frac{n_{j_2}}{RC_{j_2}},$$

for all $j_1, j_2 \in \mathcal{J}$, throughout the execution of JLU.

Proof: The equation for updating α_j in JLU is the same as that in Algorithm 1. By Lemma 3, at any point of time, we have

$$\begin{aligned}\alpha_j =& (\frac{1}{d_j - 1})(d_j^{n_j/RC_j} - 1) \\ =& \frac{1}{(1 + 1/C_j)^{RC_j} - 1}[(1 + 1/C_j)^{n_j} - 1],\end{aligned}$$

as $d_j = (1 + 1/C_j)^{RC_j}$.

Note that $\alpha_j \to \frac{e^{n_j/C_j}-1}{e^R-1}$ for a fixed R and all $\frac{n_j}{C_j} \le R$, as $C_j \to \infty$. Thus, for any $\delta > 0$, there exists a finite C_{min} such that, by requiring $C_j > C_{min}$ for all j, we have $|\alpha_j - \frac{e^{n_j/C_j}-1}{e^R-1}| < \delta/2$ for all j. Therefore, for any two servers j_1 and j_2, we have

$$\begin{aligned}&|(\alpha_{j_1} - \alpha_{j_2}) - (\frac{e^{n_{j_1}/C_{j_1}}-1}{e^R-1} - \frac{e^{n_{j_2}/C_{j_2}}-1}{e^R-1})|\\ <&|\alpha_{j_1} - \frac{e^{n_{j_1}/C_{j_1}}-1}{e^R-1}| + |\alpha_{j_2} - \frac{e^{n_{j_2}/C_{j_2}}-1}{e^R-1}| < \delta.\end{aligned}$$

This implies that

$$\begin{aligned}\alpha_{j_1} - \alpha_{j_2} > \delta \Longrightarrow &\frac{e^{n_{j_1}/C_{j_1}}-1}{e^R-1} > \frac{e^{n_{j_2}/C_{j_2}}-1}{e^R-1}\\ \Longrightarrow &\frac{n_{j_1}}{RC_{j_1}} > \frac{n_{j_2}}{RC_{j_2}},\end{aligned}$$

and the proof is complete. ■

Theorem 2: JLU is $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path.

Proof: The proof is very similar to that of Theorem 1. By Lemma 3, $\frac{n_j}{RC_j} < 1 \Leftrightarrow \alpha_j < 1$. Therefore, under JLU, an arriving job i is allocated if and only if $\min_{j\in K_i} \alpha_j < 1$. For any $\delta > 0$, we pick a sufficiently large C_{min} so that (4) holds.

We can establish that the solutions $\{X_{ij}\}$, $\{\alpha_j\}$, and $\{\beta_i\}$ produced by JLU satisfy all constraints in **Allocation(R)** and **Dual** using an argument that is virtually the same as that in the proof of Theorem 1.

Next, we derive the ratio between $\sum_{ij} X_{ij}$ and $\sum_j C_j\alpha_j + \sum_i \beta_i$. Both formulas are initially 0. We now consider the amounts of change of these two formulas when a job i arrives. We use $\Delta P(R)$ to denote the change of $\sum_{ij} X_{ij}$, and ΔD to denote the change of $\sum_j C_j\alpha_j + \sum_i \beta_i$.

If job i is discarded, then $\Delta P(R) = \Delta D = 0$.

On the other hand, consider the case when job i is assigned to server $j^* = \text{argmin}_{j\in K_i} \frac{n_j}{RC_j}$. By Lemma 4, $\alpha_j \ge \alpha_{j^*} - \delta$, for all $j \in K_i$.

We now have

$$\begin{aligned}&\frac{\Delta D}{\Delta P(R)} = \Delta D\\ =&C_{j^*}(\frac{\alpha_{j^*}}{C_{j^*}} + \frac{1}{(d_{j^*}-1)C_{j^*}}) + 1 - \min_{j\in K_i} \alpha_j\\ \le&C_{j^*}(\frac{\alpha_{j^*}}{C_{j^*}} + \frac{1}{(d_{j^*}-1)C_{j^*}}) + 1 - \alpha_{j^*} + \delta\\ =&\frac{(1+1/C_{j^*})^{RC_{j^*}}}{(1+1/C_{j^*})^{RC_{j^*}}-1} + \delta.\end{aligned}$$

Let $C_{min} \to \infty$, and we have under JLU,

$$\frac{\sum_j C_j\alpha_j + \sum_i \beta_i}{\sum_{ij} X_{ij}} \le \frac{e^R}{e^R-1} + \delta,$$

for any $\delta > 0$

Finally, by Lemma 2, we establish that JLU is $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path. ■

By Lemma 1, we also have the following theorem.

Theorem 3: Both Algorithm 1 and JLU are $(R, \frac{e^R}{e^R-1})$-competitive-in-expectation.

V. Lower Bounds of Competitive Ratios

In Section IV, we show that our online policies are both $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path. In this section, we will study the lower bound for the competitive ratio per sample path. We focus on a special class of systems described below:

A system in this class has N servers with capacity C each, where C is chosen to be a multiple of $N!$. A total number of NC jobs arrive in sequence, and they are separated into N groups, where the k-th group contains jobs $\{(k-1)C+1, (k-1)C+2, \ldots, kC\}$. Jobs in the same group can be served by the same subset of servers. The subset of servers that can serve a job i, i.e., K_i is constructed as follows: Jobs in the first group $\{1, 2, \ldots, C\}$ can be served by all servers, i.e., $K_i = \mathcal{J}$. For each job in the $(k+1)$-th group, its K_i is obtained by removing one element from that for jobs in the k-th group. More specifically, for a job i_1 in the k-th group and a job i_2 in the $(k+1)$-th group, we have $K_{i_2} \subsetneq K_{i_1}$ and $|K_{i_2}| = |K_{i_1}| - 1$. It is easy to verify that an offline policy can allocate all NC jobs for all systems in this class.

We consider a policy, namely, EVEN, that evenly distributes jobs in the same group among all available servers. We first establish the number of jobs that can be allocated by EVEN, and then show that no policy can guarantee to allocate more jobs than EVEN within this class of systems.

Lemma 5: When the capacity is increased by R times, EVEN serves at most $(N - \frac{N+1}{e^R} + 2)C$ jobs.

Proof: Since EVEN allocates jobs evenly on all available servers, each server gets $\frac{C}{N}$ jobs in the first group of C jobs. Similarly, as jobs in the k-th group can be served by $N-k+1$ servers, each server that can serve this group gets $\frac{C}{N-k+1}$ jobs in this group, unless the server is already fully utilized.

Consider the case when each server has RC capacity. Suppose the system can only serve up to the $(k+1)$-th group, that is, servers are still not fully utilized after serving the k-th group. We then have

$$\begin{aligned}&C(\frac{1}{N} + \frac{1}{N-1} + \frac{1}{N-2} + \ldots + \frac{1}{N-k+1}) < RC\\ \Rightarrow &\int_{N-k+1}^{N+1} \frac{1}{x}dx < (\frac{1}{N} + \frac{1}{N-1} + \ldots + \frac{1}{N-k+1}) < R\\ \Rightarrow &\log(N+1) - \log(N-k+1) = \log\frac{N+1}{N-k+1} < R\\ \Rightarrow &k < N+1 - \frac{N+1}{e^R}\end{aligned}$$

Since servers can serve up to the $(k+1)$-th group and become fully utilized after the arrival of this group, the number of jobs served in the system is then at most $(k+1)C < (N - \frac{N+1}{e^R} + 2)C$ ■

Lemma 6: When the parameters R, N, and C are fixed, no online policy can guarantee to allocate more jobs than EVEN.

Proof: We consider an alternative policy ALT and show that it cannot allocate more jobs than EVEN. Given R, N, and C, we construct K_i iteratively as follows: The first group can be served by all servers. Let j_1 be the server with the least jobs. We then choose $K_i = \mathcal{J}\backslash\{j_1\}$ for the second group. Similarly, let j_k be the server with the least jobs among severs that can serve the k-th group. We choose $K_i = \mathcal{J}\backslash\{j_1, j_2 \ldots, j_k\}$ for the $(k+1)$-th group. Under this arrival sequence, the total amount of unused capacity in all servers is at least $(RC - \frac{C}{N}) + (RC - \frac{C}{N} - \frac{C}{N-1}) + \ldots$, which is the total amount of unused capacity by EVEN. Therefore, ALT cannot allocate more jobs than EVEN. ■

Theorem 4: Any online policy cannot be better than $(R, \frac{e^R}{e^R - 1})$-competitive-per-sample-path.

Proof: This is a direct result of Lemmas 5 and 6. ■

VI. Competitive Ratios of Other Widely Policies

In this section, we study the competitive ratios of two widely used polices.

A. Join the Shortest Queue

The first policy is the join the shortest queue (JSQ) policy, which allocates jobs to servers with the smallest number of jobs. Specifically, let n_j be the number of jobs that have already been allocated to server j. When a new job i arrives, it is allocated to $\text{argmin}_{j \in K_i}\{n_j | n_j < RC_j\}$, if there exists a server $j \in K_i$ with $n_j < RC_j$.

Theorem 5: JSQ cannot be better than $(R, 1 + \frac{1}{R})$-competitive-per-sample-path.

Proof: Given R, we construct a system with two types of servers, $J1$ and $J2$, and two types of jobs, $I1$ and $I2$. Type $I1$ jobs can be served by all servers, while type $I2$ jobs can only be served by type $J2$ severs.

The system is described as Fig 1. It has one type $J1$ server with capacity MC and K type $J2$ servers with capacity C. The job arrival sequence is as follows: first MC jobs of type $I1$ arrive; then KC jobs of type $I2$ arrive. The values of M and K are chosen such that $R = M/(K+1)$. The jobs (or servers) of same type are in the same square box. An arrow line indicates that the job can be allocated to the server.

The optimal offline policy is to allocate all type $I1$ jobs to the server of type $J1$, and allocate all type $I2$ jobs to type $J2$ servers. This allocation can allocate all $MC+KC$ jobs, and $\Gamma_{opt} = MC + KC$.

Now, consider the performance of JSQ when the server capacity is increased by R times. After the increase, the type $J1$ server has RMC capacity, and all other servers have RC capacity. The first MC arrivals are all type $I1$ jobs, who can be served by all servers. Therefore, JSQ evenly distribute these jobs to all servers, and each server gets $MC/(K+1) = RC$ jobs. Next, type $I2$ jobs arrive,

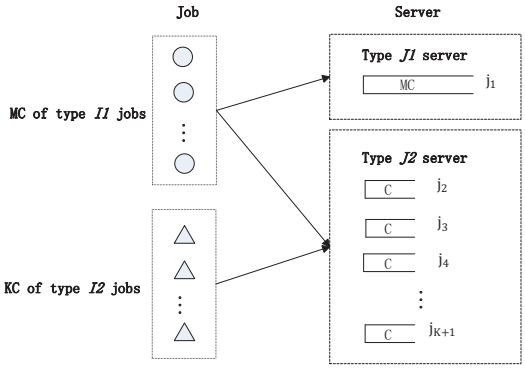

Fig. 1. System illustration for the analysis of JSQ.

and they can only be served by type $J2$ servers. However, at this point, all type $J2$ servers are fully utilized, and no type $I2$ job can be served. The total number of served jobs under JSQ is $\Gamma_{JSQ}(R) = MC$.

We then have

$$\frac{\Gamma_{opt}}{\Gamma_{JSQ}(R)} = \frac{MC + KC}{MC} = \frac{M+K}{M} \to 1 + \frac{1}{R},$$

as $K \to \infty$, and $M = R(K+1)$. ■

Theorem 6: JSQ cannot be better than $(R, 1 + \frac{1}{R^2+2R})$-competitive-in-expectation.

Proof: Given R, we use the same system as that in the proof of Theorem 5, but consider that the actual arrival sequence is a random permutation of all jobs. We still have $\Gamma_{opt} = MC + KC$. In this proof, we choose M and K such that $R = \frac{M}{K}$.

Now, consider the performance of JSQ when the server capacity is increased by R times. Type $J2$ servers can serve all jobs, while the type $J1$ server can only serve jobs of type $I1$. Under JSQ, whenever a type $J2$ server has the least jobs among all servers, the next job will be allocated to this server, regardless of the type of the job. Therefore, under JSQ, all type $J2$ servers will have at most one less job than the number of jobs at the $J1$ server before all type $J2$ servers are fully utilized. Hence, after $(K+1)RC$ arrivals, all type $J2$ servers are fully utilized. From then on, only type $I1$ jobs can be served. Further, there are MC type $I1$ jobs and KC type $I2$ jobs. Under a random permutation, the average number of type $I1$ jobs that arrive after the first $(K+1)RC$ arrivals is $\frac{MC}{MC+KC}[MC + KC - (K+1)RC] = MC - \frac{M(K+1)RC}{M+K}$. The expected number of jobs served by JSQ is then

$$\begin{aligned} E[\Gamma_{JSQ}(R)] \leq & (K+1)RC + MC - \frac{M(K+1)RC}{M+K} \\ = & MC + (K+1)RC\frac{K}{M+K}, \end{aligned}$$

and hence

$$\begin{aligned}\frac{\Gamma_{opt}}{E[\Gamma_{JSQ}(R)]} &\geq \frac{MC+KC}{MC+(K+1)RC\frac{K}{M+K}}\\ &= \frac{RK+K}{RK+(K+1)RK/(RK+K)}\\ &\to \frac{(R+1)^2}{R(R+1)+R}\\ &= 1+\frac{1}{R^2+2R},\end{aligned}$$

as $K\to\infty$ and $M=KR$. ■

B. Join the Most Residue Queue

Join the most residue queue (JMQ) allocates jobs to servers with most remaining space, which is the server capacity minus the number of allocated jobs in this server. Let RC_j be the capacity of server j, n_j be the number of jobs that have already been allocated to j. The arriving job i is allocated to server $\text{argmax}_{j\in K_i}\{RC_j-n_j|n_j<RC_j\}$, if there exists a server $j\in K_i$ with $n_j<RC_j$.

Theorem 7: Join the most residue queue policy cannot be better than $(R,1+\frac{1}{R})$-competitive-per-sample-path.

Proof: Given R, we construct a system with two types of servers, $J1$ and $J2$, and two types of jobs, $I1$ and $I2$. Type $I1$ jobs can only be served by type $J1$ servers, while type $I2$ jobs can be served by all servers.

The system has one type $J1$ server with capacity $(M+1)C$, and K type $J2$ servers with capacity C. The job arrival sequence is as follows: first KC jobs of type $I2$ arrive; then $(M+1)C$ jobs of type $I1$ arrive. The value of M and K are chose such that $R=K/M$.

The optimal offline policy is to allocate all type $I2$ jobs to servers of type $J2$ and all type $I1$ jobs to the type $J1$ server. The total number of jobs allocated by this policy is $(M+1)C+KC$, and $\Gamma_{opt}=(M+1)C+KC$.

When the server capacity is increased by R times, type $J1$ server has $R(M+1)C$ capacity, and type $J2$ servers have RC capacity. The first arriving $KC=MRC$ jobs are of type $I2$. They can be served by both type $J1$ and $J2$ servers. JMQ allocates all these jobs to type $J1$ server. Type $J1$ server can therefore serve only RC jobs of type $I1$. The total number of jobs served is $R(M+1)C$.

We then have

$$\begin{aligned}\frac{\Gamma_{opt}}{\Gamma_{JMQ}(R)} &= \frac{(M+1)C+KC}{R(M+1)C}\\ &= \frac{M}{M+1}+\frac{1}{R}\to 1+\frac{1}{R},\end{aligned}$$

as $M\to\infty$, and $K=MR$. ■

Theorem 8: JMQ cannot be better than $(R,1+\frac{1}{R^2+2R})$-competitive-in-expectation.

Proof: We use the system in the proof of Theorem 7 but consider that the actual arrival sequence is a random permutation of all jobs. First, we have $\Gamma_{opt}=(M+1)C+KC$. Now we study $E[\Gamma_{JMQ}(R)]$. In this proof, we choose M and K such that $K=MR-1$.

The type $J1$ server can serve all jobs and has MRC more capacity than others. Thus, the first MRC jobs will be allocated to the type $J1$ server, regardless of job types. After the first MRC arrivals, the type $J1$ server has RC capacity left and hence at most RC more type $I1$ jobs can be served. Since there are $(M+1)C$ type $I1$ jobs and KC type $I2$ jobs, the average number of type $I1$ jobs among the first MRC arrivals is $MRC\cdot\frac{M+1}{M+1+K}$. The expected number of allocated jobs is then no more than $RC+MRC\frac{M+1}{M+1+K}+KC$. Therefore we have:

$$\begin{aligned}&\frac{\Gamma_{opt}}{E[\Gamma_{JMQ}(R)]}\\ &\geq \frac{MC+C+KC}{RC+MRC\frac{M+1}{M+1+K}+KC}\\ &= \frac{M+MR}{MR-1+R+MR\frac{M+1}{M+MR}}\\ &= \frac{M^2(R+1)^2}{(M+1)RM(R+1)-M(R+1)+MR(M+1)}\\ &\to \frac{(R+1)^2}{R(R+1)+R}\\ &= 1+\frac{1}{R^2+2R}\end{aligned}$$

as $M\to\infty$ and $K=MR-1$. ■

C. Discussions

We have shown that our policies are $(R,\frac{e^R}{e^R-1})$-competitive-per-sample-path, while JSQ and JMR are no better than $(R,1+\frac{1}{R})$-competitive-per-sample-path, and no better than $(R,1+\frac{1}{R^2+2R})$-competitive-in-expectation. Suppose we are given a system where the offline policy can allocate all jobs. In order to guarantee that at least $1-\frac{1}{\theta}$ of the jobs are allocated, our policy only needs to increase the capacity by R times so that $\frac{e^R}{e^R-1}\leq 1/(1-\frac{1}{\theta})$. Therefore, choosing $R=\ln\theta$ is sufficient. In contrast, the two commonly used policies, JSQ and JMQ, require to increase the server capacity by at least $(\theta-1)$ times. Even when we consider that the arrival sequence is a random permutation of all jobs, and only require JSQ and JMQ to allocate $1-\frac{1}{\theta}$ of the jobs *on average*, they still need to increase the capacity by at least $\sqrt{\theta}-1$ times.

Fig. 2 plots the capacity requirement for different allocation ratios. From the figure we can observe that as the allocation ratio approaches 1, the capacity requirement using JSQ or JMQ increases much faster than that using PD or JLU. For example, if we need to allocate at least 95% of the jobs, i.e., $\theta=20$, our policies only require $R=3$, while JSQ and JMQ both require $R\geq 19$. Even when the arrival sequence is a random permutation of all jobs, JSQ and JMQ still need $R\geq 3.5$. Further, one may notice that JSQ and JMQ seem to outperform our policies when the arrival sequence is a random permutation, and when the allocation ratio is low. However, the comparison is not fair. When the arrival sequence is a random permutation,

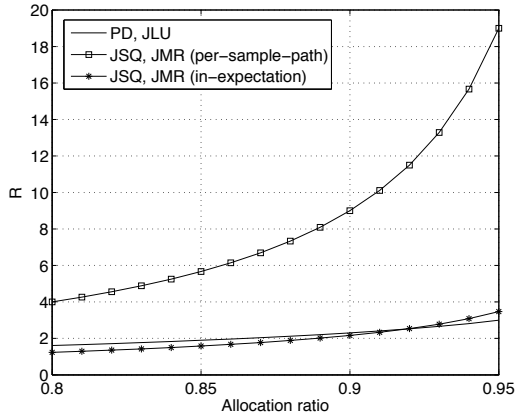

Fig. 2. Capacity requirements of different policies.

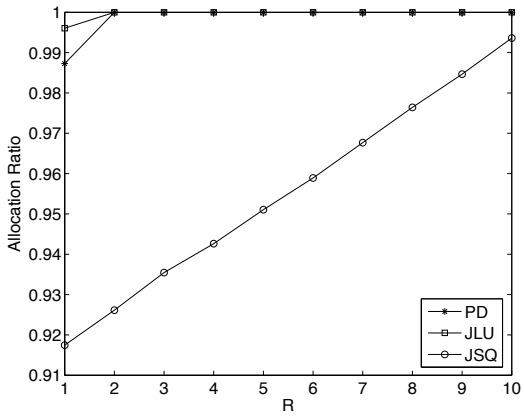

Fig. 3. Simulation results for the first scenario.

TABLE I
SYSTEM SETTING FOR THE FIRST SCENARIO.

Server Type	Number of Servers	Capacity	Job Type	Number of Jobs	K_i
$J1$	1	25000	$I1$	2500	$J2$
$J2$	50	50	$I2$	25000	$J1,\ J2$

our policies are likely to achieve better performance. In the next section, we will demonstrate this by simulations.

VII. SIMULATION

In this section, we evaluate the performance of the four policies, including PD, JLU, JSQ, and JMQ, discussed in this paper by simulations. We consider three different scenarios:

First, we compare our polices with JSQ. We construct a system with two types of servers $J1,\ J2$; and two types of jobs $I1,\ I2$. Type $I1$ job can be served by type $J2$ server, type $I2$ job can be served by both $J1$ and $J2$ server. The number of jobs and capacities of servers are shown in Table I. The arrival sequence is a random permutation of all jobs. Simulation results are the average of 10 runs. Under different R, we compute the allocation ratio by the number of jobs allocated with online policy dividing that with optimal offline policy. The simulation result is shown in Fig. 3. We can observe that PD and JLU policies outperform JSQ.

Next, we compare our polices with JMQ. We construct a system with two types of servers: $J1,\ J2$; and two types of jobs: $I1,\ I2$. The setting for jobs and servers are shown in Table II. The result is shown in Fig. 4. We can observe that PD and JLU policies outperform JMQ.

Last, we construct a system with four types of servers: $J1$, $J2$, $J3$, and $J4$; and four types of jobs: $I1,\ I2,\ I3,\ I4$. The detailed setting for servers and jobs are listed in Table III, which is a combination of the two previous settings. Simulation results are shown in Fig. 5. We can observe that PD and JLU outperform both JSQ and JMQ.

TABLE II
SYSTEM SETTING FOR THE SECOND SCENARIO.

Server Type	Number of Servers	Capacity	Job Type	Number of Jobs	K_i
$J1$	1	2550	$I1$	2550	$J1$
$J2$	500	50	$I2$	25000	$J1,\ J2$

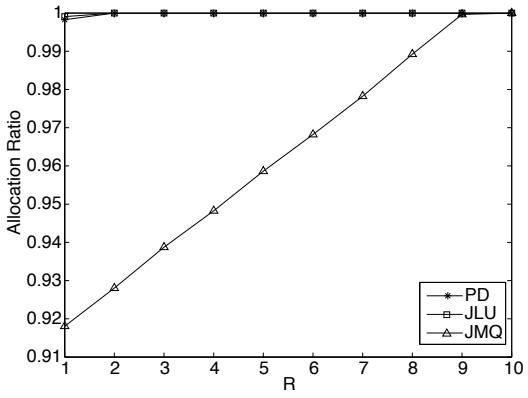

Fig. 4. Simulation results for the second scenario.

TABLE III
SYSTEM SETTING FOR THE SECOND SCENARIO.

Server Type	Number of Servers	Capacity	Job Type	Number of Jobs	K_i
$J1$	1	10000	$I1$	2500	$J2$
$J2$	50	50	$I2$	10000	$J1$,$J2$
$J3$	1	2550	$I3$	2550	$J3$
$J4$	200	50	$I4$	10000	$J3$,$J4$

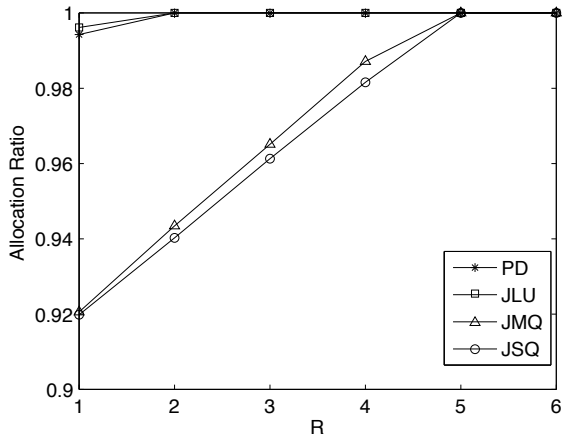

Fig. 5. Simulation results for the third scenario.

From Fig. 3 to 5, we notice that our proposed PD and JLU policies almost have identical performance. Also, we notice that the allocation ratios of all four policies converge to 1 as R increasing. However, JSQ and JMQ converges much slower than PD and JLU.

In Section IV we prove that PD and JLU are $(R, \frac{e^R}{e^R-1})$-competitive-in-expectation. In Secion VI we prove that both JSQ and JMQ are no better than $(R, 1+\frac{1}{R^2+2R})$-competitive-in-expectation. Although it looks like that the competitive ratio in expectation of our policies is not as good as that of the other two policies when R is small, the simulation results show that our policies have better allocation ratio even with small R. Thus with random permutation of the jobs arrival sequence, the performance of our policies are better than of JSQ and JMR policies.

VIII. Conclusion

In this paper, we study the job allocation problem with unknown job arriving pattern under hard allocation ratio requirement. Given the capacity of current data center which serves all jobs offline, we aim to find how much capacity we need to expand to meet the allocation ratio requirement for any unknown job arrival sequence.

We propose two online policies PD and JLU which are both $(R, \frac{e^R}{e^R-1})$-competitive-per-sample-path. We also prove that our policies can achieve any allocation ratio requirement with the least capacity. Next we study the performance of two widely used policies, JSQ and JMQ. We prove that both policies are no better than $(R, 1+\frac{1}{R})$-competitive-per-sample-path. Therefore, they need an order higher capacity to achieve the same allocation ratio requirement than our policies. We further prove that JSQ and JMQ are no better than $(R, 1+\frac{1}{R^2+2R})$-competitive-in-expectation by taking random permutation of the jobs in the arrival sequence. The simulation results show that our policies are still better than JSQ and JMQ. Thus our policies are much preferable than the two widely used JSQ and JMQ policies.

References

[1] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," in *Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems 2006*, EuroSys '06, (New York, NY, USA), pp. 333–344, ACM, 2006.

[2] S. Acharya and B. Smith, "Characterizing user access to videos on the world wide web," in *In Proceedings of MMCN*, 2000.

[3] C. Huang, J. Li, and K. W. Ross, "Can internet video-on-demand be profitable," in *in Proceedings of ACM SIGCOMM*, 2007.

[4] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 229–238, June 2008.

[5] G. Chatzopoulou, C. Sheng, and M. Faloutsos, "A first step towards understanding popularity in youtube," in *INFOCOM IEEE Conference on Computer Communications Workshops , 2010*, pp. 1–6, March 2010.

[6] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, (New York, NY, USA), pp. 352–358, ACM, 1990.

[7] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 229–238, June 2008.

[8] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *Networking, IEEE/ACM Transactions on*, vol. 17, pp. 1357–1370, Oct 2009.

[9] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," 1992.

[10] A. DeSantis, G. Markowsky, and M. Wegman, "Learning probabilistic prediction functions," in *Foundations of Computer Science, 1988., 29th Annual Symposium on*, pp. 110–119, Oct 1988.

[11] N. Littlestone, "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm," *Mach. Learn.*, vol. 2, pp. 285–318, Apr. 1988.

[12] N. Littlestone, "Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow," in *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, COLT '91, (San Francisco, CA, USA), pp. 147–156, Morgan Kaufmann Publishers Inc., 1991.

[13] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, "Webwatcher: A learning apprentice for the world wide web," pp. 6–12, AAAI Press, 1995.

[14] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press, 2006.

[15] M. Hutter, "On the foundations of universal sequence prediction," *CoRR*, vol. abs/cs/0605009, 2006.

[16] G. Goel and A. Mehta, "Online budgeted matching in random input models with applications to adwords," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, (Philadelphia, PA, USA), pp. 982–991, Society for Industrial and Applied Mathematics, 2008.

[17] C. Karande, A. Mehta, and P. Tripathi, "Online bipartite matching with unknown distributions," in *In STOC*, 2011.

[18] J. Feldman, A. Mehta, V. S. Mirrokni, and S. Muthukrishnan, "Online stochastic matching: Beating 1-1/e," *CoRR*, vol. abs/0905.4100, 2009.

[19] B. Kalyanasundaram and K. R. Pruhs, "An optimal deterministic algorithm for online b-matching," *Theoretical Computer Science*, vol. 233, p. 2000, 2000.

[20] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "Adwords and generalized on-line matching," in *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pp. 264–273, Oct 2005.

[21] S. Moharir and S. Sanghavi, "Online load balancing and correlated randomness," in *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pp. 746–753, Oct 2012.

[22] D. Bertsimas and J. N. Tsitsiklis, *Introduction to linear optimization*. Athena Scientific series in optimization and neural computation, Belmont, Mass. : Athena Scientific, c1997., 1997.