Competitive Analysis using Resource Augmentation

Referred paper

1 Problem Model

Jobs arrive over time at the data center, Job i arrives at time r_i, with know work/size w_i and known income function $I_i(t)$. The function $I_i(t)$, defined for all $t > 0$, gives the income earned if job i is completed at time t. We assume that the income function $I_i(t)$ is non-negative and non-increasing. We assume that the income goes to 0 at the completion time approaches infinity, that is $\lim_{t\to\infty} I_i(t) = 0$.

The objective is to maximize the income.

$$\sum_i I_i(C_i) - E_i$$

Notations used in this analysis is summarized in Table 1.

notation	meaning
r_i	job i's release time
w_i	job i's size
$I_i(t)$	income of completing job i at time t
C_i	the completion time of job i
$P(s_i)$	the power consumption of running job i at speed s_i
E_i	energy consumption of job i, $E_i = P(s_i)w_i/s_i$
n	total number of jobs

2 Competitive Ratio Analysis

Lemma 1. *The competitive ratio of any deterministic algorithm can not be bounded by any function of n (total number of jobs), even if jobs have unit work.*

Proof. We consider each processor has only two possible speeds, 1 and $1/\epsilon$.

We consider two job instances and a power function for which $P(1) = 1$ and $P(1/\epsilon) = L/\epsilon$, where $\epsilon > 0$ is a small number and $L > 1$. (For intuition, think of L as large).

- job 1 has $r_1 = 0, w_1 = 1$ and $I_i(t) = 1 + \epsilon$ if $t \leq 1$ and $I_1(t) = 0$ for $t > 1$.
- job 2 has $r_2 = 1 - \epsilon, w_2 = 1$ and $I_2(t) = L + 1 - \epsilon$ if $t \leq 1$ and $I_2(t) = 0$ for $t > 1$.

In order to gain positive income by running job 1, we need to run it at time 0 and with speed 1. If we run at speed $1/\epsilon$, then the income would be $1 - L < 0$. The profit of execute job 1 at time 0 is $1 + \epsilon - 1 = \epsilon$. The profit of executing job 2 at time $1 - \epsilon$ is $L + 1 - \epsilon - L = 1 - \epsilon$.

Denote the deterministic algorithm and adversary as ALG and ADV respectively.

Case 1: If ALG does not schedule job 1, then ADV schedules job 1 and does not release job 2. Thus the competitive ratio is $\frac{ADV}{ALG} = \frac{\epsilon}{0} = \infty$.

Case 2: If ALG schedule job 1, then ADV release job 2. Assume ALG does not switch to job 2, then the profit it gains is ϵ. Otherwise, the profit it gains is the income of job 2 subtract the energy cost on job

1, i.e., $1-\epsilon-(1-\epsilon)=0$. ADV will only execute job 2 and gains income $1-\epsilon$. Thus the competitive ratio is $\frac{ADV}{ALG}=\frac{1-\epsilon}{\epsilon}$, by making ϵ small, we can make this ratio as large as we like.

□

3 Competitive Analysis with Resource Augmentation

Lemma 2. *For any $\epsilon > 0$, there is an online algorithm A that is $1+\epsilon$-speed $O(1/\epsilon^3)$-competitive for profit maximization.*

$(1+\epsilon)$-speed: If ADV can run at speed s with power P, then ALG can run at speed $(1+\epsilon)*s$ with power P.

3.1 Critical Speed

The maximum speed that job i can be run without non-negative profit. Define the speed at $\hat{s_i}$, then we have

$$I_i(t) - P(\hat{s_i}(t))\frac{w_i}{\hat{s_i}(t)} = 0 \tag{1}$$

Divide Equation 1 by $(1+\epsilon)$ and regrouping terms, we have

$$I_i(t) - P(\hat{s_i}(t))\frac{w_i}{(1+\epsilon)\hat{s_i}(t)} = \frac{\epsilon}{1+\epsilon}I_i(t) \tag{2}$$

3.2 Online Algorithm

High-level description

- When a job arrives, we use the *deadline setting* and *job assignment policy* to set a deadline and assign the job to various time intervals on machines. Note, this assignment is not a schedule, as we many assign multiple jobs to the same machine at the same time.
- We also set a speed via the speed scaling policy.
- We then use *job selection* policy to take some jobs from the intervals and machines on which they are assigned and actually run them on the machines.

Throughout this section, we use $\delta = \frac{\epsilon}{2}$.

Invariants The online algorithm maintains a pool Q of *admitted* jobs. A job i remains in Q until it is completed or its deadline passes. Each job i in Q has several associated attributes.

- A deadline d_i assigned to job i when it was released.
- The critical speed $\hat{s_i}^A = \hat{s_i}^A(d_i)$ derived from the deadline d_i , and defined by Equation 1. The online algorithm will run job at speed $(1+2\delta)\hat{s_i}^A$.
- A collection of time intervals $J(i) = \{[t_1,t_1'],[t_2,t_2'],\cdots,[t_h,t_h']\}$, where $r_i \le t_1 \le t_1' \le t_2 \le \cdots \le t_h \le t_h' = d_i$. This collection $J(i)$ is fixed when job i is released (but depends on previously scheduled jobs). The total length of the time intervals in $J(i)$ will be $(1+\delta)\frac{w_i}{(1+2\delta)\hat{s_i}^A}$. Question: why define the length in this way?
- A processor $m_{i,k}$ associated with job i and each time interval $[j_k,j_k'] \in J(i)$ that was fixed at time r_i. Intuitively, at the time that job i is released, the online algorithm is tentatively planning on running job i on processor $m_{i,k}$ during the time period $[t_k,t_k']$. We say that job i is assigned to run on m_k during times $[t_k,t_k']$.

Deadline setting Given a fixed deadline d_i, we can obtain the critical speed $\hat{s_i}^A = \hat{s}_i(d_i)$, the profit $p_i^A = I_i(d_i) - \frac{P(\hat{s}_i^A)w_i}{\hat{s}_i^A(1+2\delta)}$, and the online density $u_i^A = \frac{p_i}{t_i} = \frac{p_i\hat{s}_i^A}{w_i}$.

Then let $c = 1 + \frac{2}{\delta}$, let $X(\frac{u_i^A}{c})$ be the set of jobs in Q with density at least $\frac{u_i^A}{c}$. Let A be the maximum subintervals of $[r_i, t]$of times such that for each $[a, a'] \in A$, there is a processor m_k for which no job in $X(\frac{u_i^A}{c}$ is assigned to run on m_k during any time in $[a, a']$. Adjust d_i to make A be exactly the value $(1+\delta)\frac{w_i}{(1+2\delta)\hat{s_i}^A}$. If there is no d_i that satisfy this condition, then the job will not be admitted.

Speed scaling policy Every job is run at its critical speed for its set deadline.

Job selection policy At any time t, on any processor m_k, run the job i, assigned to m_k at time t, of maximum density.

3.3 Construction of a structurally-nice near-optimal schedule OPT'

We assume the optimal schedule is non-migratory. Question: why make this assumption?

To facilitate the comparison of online schedule to the optimal schedule, we modify the per-job power functions as follows, after modification, it will be more profitable.

$$P_i'(s) = \begin{cases} P(s/(1+\epsilon)) & \text{if } s \in [s_i^{min}(C_i), \hat{s}_i(C_i^O)] \\ P(s) & \text{otherwise .} \end{cases} \tag{3}$$

Lemma 3. *There exist a schedule that in OPT' each job i that runs does so at speed $(1+\epsilon)\hat{s}_i(C_i^O)$, and the total profit obtained using the modified power is at least $\frac{\epsilon}{1+\epsilon}$ times of the profit that OPT achieves using the power function P.*

Proof. C_i^O is the completion time of job i by OPT. For OPT, it will run the job i at speed $s \in [s_i^{min}, \hat{s}_i]$ where $s_i^{min} = s_i^{min}(C_i^O)$, and $\hat{s}_i = \hat{s}_i(C_i^O)$. Let OPT' run job at speed $(1+\epsilon)\hat{s}_i$, as we are speeding the job up, each job still completed by its completion time C_i^O. Now the net profit associated with job i is

$$\begin{aligned} I_i(C_i^O) - P_i'(\hat{s}_i(1+\epsilon))\frac{w_i}{\hat{s}_i(1+\epsilon)} = I_i(C_i^O) = P(\hat{s}_i)\frac{w_i}{\hat{s}_i(1+\epsilon)} \\ = \frac{\epsilon}{1+\epsilon} I_i(C_i^O) \\ \geq \frac{\epsilon}{1+\epsilon} p_i^O \end{aligned}$$

□

The first equality follows from the definition of P', and the second equality follows from Equation 3, and the final equality follows because the income must be greater than the profit.