Online Algorithms 07/12/2011

Lecture Notes 6: Load Balancing cont.

Professor: Yossi Azar *Scribe:Kalev Alpernas*

1 Introduction

In this lecture we will see two variants of the Load Balancing problem seen in the previous lecture:

- *Load Balancing on Related Machines* The load of a certain task is determined by its weight, as well as the power of the machine.
- *Restricted Assignment* Tasks can only be assigned to certain machines.

2 Load Balancing on Related Machines

Definition 2.1. *m machines. Machine j has the speed v_j. Algorithm A assigns task i to machine j. The load on machine j is then defined by:*

$$l_j \leftarrow l_j + \frac{w_i}{v_j}$$

Or, in a non recursive definition:

$$l_j = \frac{1}{v_j} \sum_{i|A(i)=j} w_i$$

The goal is to minimize the $\max_j l_j$

Possible algorithms:

1. **Pre Greedy** assign the task to the least loaded machine. That is, choose j so that l_j is minimized.
2. **Post Greedy** assign the task to the machine which will be the least loaded with the added task. That is, choose j so that $l_j + \frac{w_i}{v_j}$ is minimized.
3. assign the task to the fastest machine.

- Algorithm 3 will not work: if we have $m-1$ machines with the speed 1 and 1 machine with the speed $1+\epsilon$. If we get m unit tasks, *opt* will distribute them evenly, while on_3 will assign them all to the slightly faster machine.

  $on(\sigma) = m$

  $opt(\sigma) = 1$

  This yields a competitive ratio of m.

- Algorithm 1 fails as well, even for $m = 2$ machines: if we gave a machine with the speed 1 and a machine with the speed M, where $M \gg 1$, and we have 2 tasks with the size 1. Note that *opt* will assign them both to the faster machine, while on_1 will assign one task to each machine

  $on_1(\sigma) = 1$

  $opt(\sigma) = \frac{2}{M}$

  This yields a competitive ratio of at least $\frac{M}{2}$.

Theorem (Without Proof) 2.2. *The Post Greedy algorithm is* $\Theta(\log m)$*-competitive.*

We will now show an 8-competitive algorithm.

Step I

Lets assume that the optimal value, λ, at each step is known, or that there is a good upper bound to it. The idea is to assign the task to the slowest machine s.t. the load after the assignment is $< 2\lambda$.

w.l.o.g the machines are sorted from the slowest to the fastest by their index.

Algorithm. *We will assign the task to* $\{\min j | l_j + \frac{w_i}{v_j} \leq 2\lambda\}$
If no such machine exists, the algorithm will fail.

This is a "utilize the weak" algorithm.

Theorem 2.3. *if* $\lambda \geq opt(\sigma)$ *then our algorithm will never fail.*
In particular, the load is $\leq 2\lambda$*. We will get 2-competitiveness for* $\lambda = opt(\sigma)$

Proof. Lets assume we failed on task i. Since m is the fastest machine, and each task on the fastest machine $\leq \lambda$, we get $\frac{w_i}{v_m} \leq \lambda$. Hence $l_m > \lambda$.

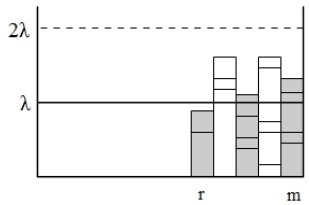

Figure 1: The state of the machines at the moment we failed

Not all machines can be loaded $\geq \lambda$ because of the Conservation of Volume rule (otherwise, the sum total work we did would be greater than that of *opt*.)

We define r as the fastest machine whose load is smaller or equal than λ. i.e. for $j > r$ we have $l_j > \lambda$

According to the Conservation of Volume rule there exists at least one task, k, which the Online Algorithm assigns to a machine j such that $r + 1 \leq j \leq m$ and *opt* assigns to a machine $\leq r$. In particular, $\frac{w_k}{v_r} \leq \lambda$ and $l_r + \frac{w_k}{v_r} \leq \lambda + \lambda$. Hence, we should have assigned task k (whose weight is w_k) to a machine $1 \leq j \leq r$. Contradiction. □

Step II

Finding λ.

1. $\lambda \leftarrow \lambda_0 \leftarrow \frac{w_1}{v_m}$
2. run A(λ)
3. if failed, $\lambda \leftarrow 2\lambda$. Back to step 2.

Once $\lambda \geq opt(\sigma)$ the algorithm must stop, by Theorem 2.3.

In each iteration of the algorithm with a new λ it continues running as though there is no load on the machines. That is because the algorithm assumes empty load on the machines. Otherwise the algorithm will not work.

Note that $\lambda_{final} \leq 2opt(\sigma)$. Hence, the real load will be:

$$2\lambda_0 + 2(2\lambda_0) + 2(4\lambda_0) + \ldots + 2\lambda_{final} \leq 2\lambda_0 + \ldots + 2 \cdot 2opt(\sigma) \leq 8opt(\sigma)$$

Theorem 2.4. *If there exists an algorithm for the Load Balancing problem, which is c-competitive given the value of opt(σ), then there exists a 4c-competitive algorithm without knowing the value of opt(σ) (the theorem requires the assumption that we know $\lambda_0 < opt(0)$)*

Proof.

$$c\lambda_0 + c(2\lambda_0) + \ldots + c\lambda_{final} \leq c\lambda_0 + \ldots + c \cdot 2opt(\sigma) \leq 4c \cdot opt(\sigma)$$

□

Note. *The choice of multiplying λ by 2 at each iteration is exactly the path search problem we've seen in the first lecture. Recall that the optimal common ratio of the geometric progression was 2.*

3 Restricted Assignment

Definition 3.1. *We have m identical machines. Task i has a weight w_i and $\phi \neq M(i) \subset M$ a subset of all the machines. Task i can only be assigned to some of the machines, i.e. only to the machines $j \in M(i)$. The goal is to minimize the* $\max_j l_j$

Note. *This model is represented by a bi-partite graph.*

Algorithm. *Greedy - assign the task to the machine with the smallest load, among $M(i)$.*

Theorem (Proof will be given next week) 3.2. *The greedy algorithm is* $\log m + 1$*-competitive.*

Theorem 3.3. *Every algorithm for the Restricted Assignment model is at least* $\lfloor \log m \rfloor + 1$*-competitive.*

Proof. We will build a sequence where $\forall_i w_i = 1$ and $opt = 1$.

Assume w.l.o.g that m is a power of 2. Otherwise we will take the largest power of 2 smaller than m, and use only those machines.

The first $\frac{m}{2}$ tasks with $M(i) = \{i, \frac{m}{2} + i\}, 1 \leq i \leq \frac{m}{2}$. w.l.o.g Alg will assign these tasks to i (otherwise we switch names.) opt will assign them to $\frac{m}{2} + i$.

The next $\frac{m}{4}$ tasks with $M(i) = \{i, \frac{m}{4} + i\}, 1 \leq i \leq \frac{m}{4}$ w.l.o.g Alg will assign them to i. opt will assign them to $\frac{m}{4} + i$.

We repeat this $\log m$ times.

Figure 2: Stage 1

Figure 3: Stage 2

We get one task with $M = \{1, 2\}$. Alg assigns it to 1, while opt assigns it to 2.

At last we get one last task with $M = \{1\}$, and both algorithms assign it to 1.

Why is $opt = 1$?

At each stage, opt assigns the tasks to an empty machine, while Alg assigns it tot the other machine.

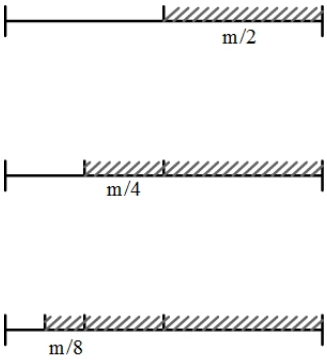

Figure 4: Load on opt

Eventually, $opt(\sigma) = 1$ and $Alg(\sigma) = \log m + 1$. If m is not power of 2 then it will be $\lfloor \log m \rfloor + 1$

□

Figure 5: Load on Alg

Definition 3.4. *A Fractional model for the Load Balancing problem is a model in which the online algorithm can split the tasks between different machines. Moreover, opt cannot split the tasks.*

Theorem 3.5. *The lower bound for a Fractional model is* $\frac{1}{2}\log m$.

Proof. We repeat the same proof, with the difference that in every step, the algorithm assigns the bigger part of the task to i and the smaller one to $i + \frac{m}{2^k}$. We lose a factor of 2 compared with the previous lower bound. □

Remark. *When looking at the expected value, a lower bound for the Fractional model (where optimum is non-fractional) is a lower bound for the random algorithm.*

Explanation of the remark: The fractional load on a machine corresponds to the expected load of the online algorithm on that machine. In particular, the lower bound is on $\max_j E(l_j)$. Since

$$E(\max_j l_j) \geq \max_j E(l_j)$$

the lower bound also holds for randomized algorithms.