UTAH STATE UNIVERSITY, COMPUTER SCIENCE DEPARTMENT

CS 7910 Computational Complexity Assignment 9

Gopal Menon

April 10, 2016

1. This exercise is for helping you to understand better the 2-approximation algorithm for the load balancing problem we studied in class.

   You are asked to consult for a business where clients bring in jobs each day for processing. Each job i has a processing time t_i that is known when the job arrives. The company has a set of ten machines, and each job can be processed on any of these ten machines.

   At the moment the business is running the simple 2-approximation greedy algorithm we discussed in class (i.e., the one without sorting the processing times; indeed, this is an "online" problem and you do not know the processing time of the next job until it arrives). They have been told that this may not be the best approximation algorithm possible, and they are wondering if they should be afraid of bad performance. However, they are reluctant to change the scheduling as they really like the simplicity of the current algorithm: jobs can be assigned to machines as soon as they arrive, without having to defer the decision until later jobs arrive.

   In particular, they have heard that this algorithm can produce solutions with makespan as much as twice the minimum possible; but their experience with the algorithm has been quite good: They have been running it each day for the last month, and they have not observed it to produce a makespan more than 20 percent above the average load, i.e., $\frac{1}{10}\sum_i t_i$.

   To try understanding why they do not seem to be encountering this factor-of-two behavior, you ask a bit about the kind of jobs and loads they see. You find out that the sizes of jobs range between 1 and 50, that is, $1 \le t_i \le 50$ for all jobs i; and the total load $\sum_i t_i$ is quite high each day: it is always at least 3000.

   Prove that on the type of inputs the company sees, the greedy algorithm will always find a solution whose makespan is at most 20 percent above the average load (this implies that the approximation ratio of the algorithm on this special input pattern is no more than 1.2).

Consider the machine M_i that had the maximum load T_i after all the jobs were assigned. Let job t_j be the last one that is assigned to M_i. The load of the machine before the job was assigned would have been $T_i - t_j$. And every machine would need to have a load that is at least $T_i - t_j$. So the the sum of the loads of all the machines ($\sum_k T_k$) would be at least $m(T_i - t_j)$, where m is the number of machines.

$$\sum_k T_k \geq m(T_i - t_j)$$

$$T_i - t_j \leq \frac{1}{m}\sum_k T_k \tag{1}$$

Since the sum of all the jobs is at least 3000 and there are 10 machines, from inequality 1 we get

$$T_i - t_j \leq \frac{3000}{10}$$

$$T_i - t_j \leq 300 \tag{2}$$

We know that the largest job size is 50, and so

$$t_j \leq 50 \tag{3}$$

Adding inequalities 2 and 3, we get

$$T_i - t_j + t_j \leq 350 + 50$$

$$T_i \leq 350 \tag{4}$$

For calculating the average load T_{avg}

$$T_{avg} = \frac{3000}{10} = 300$$

$$\frac{1}{T_{avg}} = \frac{1}{300} \tag{5}$$

Using 4 and 5, we get

$$\frac{T_i}{T_{avg}} \leq \frac{350}{300} = 1.2$$
$$T_i \leq 1.2 T_{avg}$$

Since T_i is the load of the machine with the maximum load, it is the makespan of the greedy algorithm. So the makespan with the greedy algorithm is at most 20% above the average load.