Approximation Algorithms **Dept. of CS, Aarhus University**

Homework 3

Lecturer: Nguyen Kim Thang *Student: Morten Schaumburg (20033306)*

Question 1

1. We note that the greedy algorithm from the assignment is the same as algorithm 10.2 from the book. Theorem 10.3 proves that the algorithm is a factor 2 approximation, and we will just modify the proof slightly in order to prove $2 - \frac{1}{m}$ approximation.

   Let L be the makespan of schedule that the algorithm outputs. Let M_i be a machine with load L in the schedule, and let p_j be the last job scheduled on M_i. Since M_i is the machine that is scheduled, it must have the lowest load, which must be less than the average of the sum of all jobs scheduled until now. The observation here is that since we are scheduling p_j it has not already been scheduled, and so we can lower the upper bound on the starting time a bit.

   $$L = start_j + p_j \le \frac{1}{m}\left(\sum_{i=1}^{n} p_i - p_j\right) + p_j = \frac{1}{m}\sum_{i=1}^{n} p_i + (1 - \tfrac{1}{m}) \cdot p_j$$
   $$\le \text{OPT} + (1 - \tfrac{1}{m}) \cdot \text{OPT} = (2 - \tfrac{1}{m}) \cdot \text{OPT}$$

   As a tight example we can still use example 10.4. The algorithm gives a makespan of $2m$ while OPT $= m + 1$, and the approximiation factor promises

   $$L \le (2 - \tfrac{1}{m})(m + 1) = 2m + 1 - \tfrac{1}{m}$$

2. We now sort the jobs in decreasing order of processing times instead of random order before scheduling them. If $n \le m$ then we schedule at most one job for each machine, which is optimal. If $n \ge m + 1$, then at least one machine must get at least two jobs in all schedules. Since the jobs are ordered by decreasing size, this gives us that $p_{m+1} \le \frac{1}{2} \cdot \text{OPT}$.

   There are now two possibilities, either M_i gets a single job scheduled or it gets more than one. If it only gets a single job and still finishes last, p_j must be the longest job and it's completion time is a lower bound so our schedule is optimal. If M_i gets multiple jobs assigned we know that p_j is the last of them which gives us that $j \ge m + 1$, so $p_j \le p_{m+1}$. This gives

   $$L = start_j + p_j \le \frac{1}{m}\left(\sum_{i=1}^{n} p_i\right) + p_j \le \text{OPT} + \tfrac{1}{2} \cdot \text{OPT} = \tfrac{3}{2} \cdot \text{OPT}$$

Question 2

This is the same as bin packing with fixed number of object sizes from section 10.2.1 in the book, except that our bins have size T instead of 1. We use the same method to compute first the different groupings of processing times that can be executed on one machine in time T, and then recursively compute the number of machines needed for any set of n tasks while keeping subresults in a table. If our given tasks can be executed on m or fewer machines we report YES, otherwise we return NO.