Scheduling Theory **CO 454, Spring 2009**

Homework 4

Due: June 30th, 2009

Solutions

1. **(6)**
Give an optimal algorithm to solve $(P|pmtn|C_{max})$, that is, minimizing makespan on identical parallel machines when preemption is allowed. Justify optimality. *(Hint: Think of the lower bounds done in class)*

Solution: Note that we know $OPT \geq \max\{\frac{1}{m}\sum p_j, p_{max}\}$. Call the second term D. We now show that if preemption is allowed, then there exists a schedule with $C_{max} = D$ and thus will be an optimal algorithm.

This can be achieved by what is called *McNaughton's wrap-around rule.*

> Order the jobs arbitrarily. Place the jobs on the machines in order, filling up machine i until time D has been reached. Then split job j onto machine i and $i+1$.

Since $D \geq p_j\ \forall j$, job j can be split onto at most two machines i and i_1, and will not be processed on machines i and $i+1$ at the same time. Since $\sum p_j$ units of processing time are needed and a schedule of length $D \geq \frac{1}{m}\sum p_j$ allows for $\sum p_j$ processing time units, every job gets scheduled.

Theorem 0.1. *McNaughton's wrap-around rule gives an optimal schedule for* $Pm|pmtn|C_{\max}$.

Proof. The schedule is feasible: no job is scheduled at the same time on different machines.

The schedule is optimal: the schedule has length $C_{\max} = D = \max\left\{\frac{1}{m}\sum p_j, \max\{p_j\}\right\}$ which is in turn at most both our lower bounds. □

2. **(6+6)**
We saw that List-Scheduling returned a schedule with $C^S_{max} \leq (2 - 1/m)OPT$. Find an example of an ordering of jobs such that the above inequality holds with equality. Recall we did for $m = 2$ in class.

Solution: Let there be $m(m-1)$ jobs of processing time 1 each, and one job of processing time m, in that order. OPT will process the m-job on one machine, and m jobs of processing time 1 on each of the $m-1$ machines. So $OPT = m$. List scheduling, on the other hand, will process $m-1$ jobs of processing time 1 on each of the m machines, and therefore, will take time $C_{max} = 2m - 1$.

We saw that LPT returned a schedule with $C^S_{max} \leq (4/3 - 1/3m)OPT$. Find an example of an ordering of jobs such that the above inequality holds with equality. Recall we did $m = 2$ in class.

Solution: Consider three jobs with times m, and two jobs each of times $m+1, m+2, \ldots, 2m-1$. There are $2m+1$ jobs in all. The optimum is $3m$. Machine 1 processes the three m jobs, machine i for $i > 1$ processes jobs with processing times $m+i-1$ and $2m-i+1$. Check that all jobs are processed.

What does LPT do? It processes the two jobs of length $2m-1$ on the first two machines, the two jobs of length $2m-2$ on second two, and then if m is even and $m=2k$, processes jobs of length $2m-k$ on machine $m-1$ and m. Following that, it goes down this order putting in the jobs. Thus, when the first $2m$ jobs are filled, each machine will have exactly $3m-1$ processing time. The last m will make $C_{max} = 4m-1$.

3. Consider the LPT algorithm for $(P||C_{max})$. If the machine i which processes the job which finishes last, processes k other jobs, then LPT returns a $\frac{k+1}{k}$ algorithm.

   Solution: Let ℓ be the job that finishes last and let i process it. Since there are k other jobs scheduled on machine i before ℓ, the total load on machine i at the time job ℓ starts is at least kp_ℓ. Since machine i has the least load at this time, the total load on all machines is at least mkp_ℓ. Thus, $OPT \geq kp_\ell$, implying $p_\ell \leq OPT/k$.

   The analysis of list scheduling gives us that

$$C^S_{max} \leq OPT + (1-1/m)p_\ell \leq \frac{k+1}{k} OPT$$

.

4. In class, we saw that list scheduling (using any order of the jobs) is a 2-approximation for $P||C_{\max}$. Consider the problem with l release dates, $P|r_j|C_{\max}$.

   (a) Generalize the lower bound $OPT \geq p_j \ \forall j$ to take into account the release times r_j for the jobs.

   (b) Consider the following list scheduling rule:

   Whenever a machine becomes available, schedule an *available*, unprocessed job.

   Show that this rule results in a 2-approximation for $P|r_j|C_{\max}$.

   Solution: a) $OPT \geq p_j + r_j$.
   b) Consider the machines between time r_ℓ and t_ℓ. None of the machines can be idle since otherwise ℓ would've started processing earlier. Therefore,

$$m(t_\ell - r_\ell) \leq \sum_{j=1}^{n} p_j$$

   and so,

$$C^S_{max} = p_\ell + t_\ell \leq p_l + r_l + \frac{1}{m}\sum_{j=1}^{n} p_j \leq 2OPT$$