Lecture 13: List Scheduling Algorithms

June 16th, 2009

Last class we saw a list-scheduling algorithm for minimizing makespan in identical parallel machines with no side-constraints, and proved that it is a $(2-1/m)$-factor approximation algorithm. In this class, we look at two more applications of list scheduling to minimize makespan in identical machines with precedence constraints, and to minimize makespan in an open shop.

1 $(P|prec|C_{max})$

We saw two lower bounds on OPT last class. We need another lower bound for $(P|prec|C_{max})$.

Let $(j_1, \ldots, j_\ell)$ be a set of jobs such that $(j_1 \to j_2 \to \cdots \to j_\ell)$ are the precedence constraints. Since job ℓ can start only after jobs $(1, \ldots, \ell-1)$ finish, the completion time of job ℓ is at least $(p_1 + \cdots + p_\ell)$, and this is a lower bound on OPT.

$$OPT \geq \sum_{i=1}^{\ell} p_{j_i} \quad , \text{ for all } (j_1 \to \cdots \to j_\ell)$$

Now we are ready to give the list scheduling algorithm for $(P|prec|C_{max})$.

> Fix any ordering of the job. Whenever a machine becomes free, process the earliest *available* job in this order.

Theorem 1.1. *Let S be the schedule returned by the above list scheduling algorithm. Then, $C^S_{max} \leq 2OPT$.*

Proof. Let ℓ be the job which is finishes last. Let t_ℓ be the time when the job was started. Let $\ell-1$ be the job which precedes ℓ and finishes last, and suppose it starts at time $t_{\ell-1}$. By definition, $t_\ell \geq t_{\ell-1} + p_{\ell-1}$. Similarly, let job j, for $j < \ell$ be the job which finishes last among jobs which precede job $j+1$, and let t_j be the time at which job j starts processing.

So we get a series of jobs $(1, 2, \ldots, \ell)$ such that no job precedes job 1 and $(1 \to 2 \to \cdots \to \ell)$. Note that by the lower bound above, $OPT \geq \sum_{j=1}^{\ell} p_j$.

The crucial observation is the following: For any $1 \leq j < \ell$, between times $(t_j + p_j)$ and t_{j+1}, all the m machines must be busy. Otherwise, job $(j+1)$ could've been processed earlier. The same is true between time 0 and t_1. Since these machines can be made busy only by jobs, we get

$$m \cdot (t_1 + \sum_{j=1}^{\ell-1} (t_{j+1} - (t_j + p_j))) \leq \sum_j p_j \leq m \cdot OPT$$

Rearranging, we get

$$t_1 + \sum_{j=1}^{\ell-1}(t_{j+1} - t_j) \leq OPT + \sum_{j=1}^{\ell-1} p_j$$

giving us $t_\ell \leq OPT + \sum_{j=1}^{\ell-1} p_j$, and adding p_ℓ to both sides, we get

$$C_{max} \leq OPT + \sum_{j=1}^{\ell} p_j \leq 2OPT$$

□

2 $(O||C_{max})$

We now look at the application of list scheduling to minimize makespan in an open shop. Recall in an open shop, each job j needs to be processed in each machine i, and takes time p_{ij} on it. There are no restrictions on the processing order.

What is a lower bound on OPT in this case? We now have

$$OPT \geq \sum_{i=1}^{m} p_{ij}, \quad \forall j$$

since every job must be processed on each machine. Furthermore, each machine must process every job and so we get

$$OPT \geq \sum_{j=1}^{n} p_{ij}, \quad \forall i$$

Now consider the list-scheduling algorithm which processes any unprocessed operation of a job on a free machine. Let M be the machine which finishes last, and let ℓ be the job which it finishes last. The observation is the following: at any point of time either machine M is busy or some operation of job ℓ is being performed. If neither, then the unprocessed operation of job ℓ would've been performed on machine i.

Therefore, $C_{max} \leq \sum_{i=1}^{m} p_{i\ell} + \sum_{j=1}^{n} p_{Mj} \leq 2OPT$.