CS 598CSC: Approximation Algorithms **Lecture date: Feb. 6th, 2009**
Instructor: Chandra Chekuri **Scribe: Tae Hyun Kim**

In this lecture, multiprocessor scheduling is explored with precedence constraints. Applications such as parallel programming and instruction scheduling in multi-issue processors fall into this category.

1 Multiprocessor Scheduling with Precedence Constraints

1.1 Problem Description

In the Multiprocessor Scheduling with Precedence Constraints, we are given n-jobs denoted by $\{J_1, J_2, \cdots, J_n\}$, the associated processing time $p_i, i = 1, \cdots, n$ and m identical machines to which jobs are needed to be assigned. In addition, jobs have precedence constraints between them, and we use notation $J_j \prec J_i$ to imply that J_i cannot be done before completing J_j (the smaller runs earlier). A directed acyclic graph (DAG) can encode these constraints. The goal of this problem is to minimize the makespan. Denote the start time and completion time of J_j by s_j and c_j, respectively. Then job J_j occupies time slots from $s_j + 1$ to c_j. The goal is thus $\min \max_j c_j$. This problem is also strongly **NP**-hard as the scheduling problem without constraints.

1.2 Graham's List Scheduling

According to the precedence constraints, a DAG is created, where each vertex represents a job, and a directed edge indicates a precedence constraint. One example of such DAGs is depicted in Figure 1. In the figure, the numbers in vertices are job indices, and the associated processing times with the jobs are noted next to vertices. In this DAG, it is noticeable, for instance, that $J_1 \prec J_4$, $J_2 \prec J_6$, etc. Subsequently, priority between jobs for scheduling is also given by having a DAG.

In Graham's List Scheduling, jobs are sequentially scheduled on the least loaded machine while satisfying the precedence constraints. One example of schedules for the jobs in Figure 1 is depicted in Figure 2. Due to the precedence constraints $J_5 \prec J_7 \prec J_{10}$, J_7 and J_{10} had to be delayed to c_5 and c_7, respectively.

It turns out that this scheduling gives the following approximation.

Theorem 1 *Graham's List Scheduling with any list gives a* $(2 - \frac{1}{m})$ *approximation.*

To prove this theorem, we first need to establish two lower bounds on OPT.

Claim 2 (Lower Bound 1) $\text{OPT} \geq \frac{1}{m} \sum_j p_j$.

This claim is trivial since the total amount of work ($\sum_j p_j$) should be done on m machines.

Another lower bound for OPT is given by using the concept of a *chain.* A chain is defined as a sequence of jobs which have predecessor-successor relationship. Since no two jobs in a chain can execute simultaneously, we have the following observation, where for a chain $\mathcal{A}$, $p(\mathcal{A})$ denotes $\sum_{j \in \mathcal{A}} p_j$.

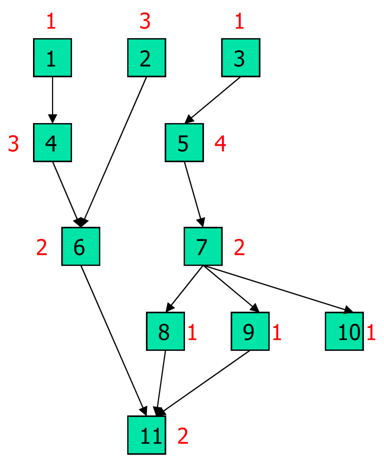

Figure 1: Directed acyclic graph (DAG) example. Numbers inside vertices are job indices, and the numbers noted by them are processing times.

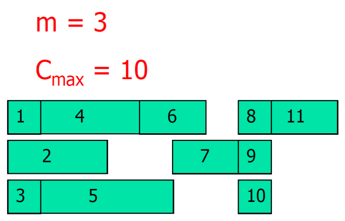

Figure 2: Output example of Graham's List Scheduling. The precedence constraints are from Figure 1.

Claim 3 (Lower Bound 2)

$$\mathrm{OPT} \geq \max_{\mathcal{A}:chain} p(\mathcal{A}), \tag{1}$$

We now define a particular chain of interest for the analysis. First, observe the example in Figure 2 which shows that not all machines are always executing, because of the precedence constraints among jobs. Let us define a full time slot to indicate a time slot during which all machines are busy. Meanwhile, a partial time slot is defined as a time slot in which some of the machines are not busy, but waiting for certain jobs to be completed. By summing up full and partial time slots, we can have a lower bound on OPT.

Define $c_{\max} := \max_j c_j$. The job i_1 is such that $c_{i_1} = c_{\max}$. Define t_2 is a maximum integer in $[0, s_{i_1}]$ such that t_2 is a partial slot. In case that $t_2 = 0$, the chain including i_1 has i_1 only. Inductively we can define the following.

- t_k : the maximum integer in $[0, s_{i_k}]$ such that t_k is a partial slot.
- i_{k+1} : the predecessor of i_k that is executing at t_k.

The index k is from 1 to k' such that $t_{k'} = 0$. It follows from the definitions that $i_1 \succ i_2 \succ \cdots \succ i_k$ and $t_1 > t_2 > \cdots > t_k > t_{k+1}$. The pictorial description of the terms is given in Figure 3.

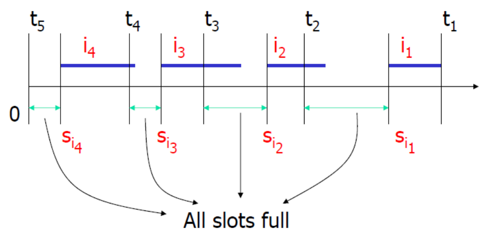

Figure 3: Illustrations on the terms i_j, t_k and s_{i_j}.

Let $\mathcal{A}$ be the chain defined in this manner, and note that some job in $\mathcal{A}$ is executing during every partial time slot of the schedule.

Further, between two consecutive jobs in a chain, all time slots are full. This fact suggests the following lemma.

Lemma 4 *For the chain $\mathcal{A}$ defined above:*

$$c_{\max} \leq p(\mathcal{A}) + \frac{1}{m}\sum_{j=1}^{n} p_j. \tag{2}$$

Proof: The completion time $c_{\max}$ is given by the sum of the number of partial and the number of full time slots in the schedule. The former is at most $p(\mathcal{A})$, and the total amount of work done in the full time slots cannot exceed the sum of processing times for all jobs on m machines. The number of full time slots is $1/m$ times the amount of work done in those slots, and so we obtain (2). □

Using the two lower bounds on OPT, Theorem 4 immediately gives us a 2-approximation.

This analysis can be further improved by observing that the upper bound for the sum of full time slots can be tighter as follows.

Claim 5

$$c_{\max} \leq 2 - \frac{1}{m}\text{OPT}. \tag{3}$$

Proof: Let n' denote the number of partial time slots in which a job of $\mathcal{A}$ is executing; we have $c_{\max} \leq n' + \frac{1}{m}\left(\sum_j p_j - n'\right)$; since the total amount of work done in full slots is no more than $\sum_j p_j - n'$. Thus, we have $c_{\max} \leq n'\left(1 - \frac{1}{m}\right) + \frac{1}{n}\sum_j p_j$. But $n' \leq p(\mathcal{A})$, and from the two lower bounds on OPT, we obtain the desired result. □

1.3 Hardness of the Problem

Some problems related to Multiprocessor Scheduling with Precedence Constraints remain open.

- Is the $m = 3$ case with unit processing times NP-hard?
- For fixed m, is there a $(2 - \epsilon)$ approximation where $\epsilon > 0$?

Even with unit processing times, we have the following hardness result:

Theorem 6 *Unless* **P** $=$ **NP**, *there is no approximation better than* $(\frac{4}{3} - \epsilon)$ *for any* $\epsilon > 0$.

This theorem is proved by a reduction from the clique problem. Given a graph $G = (V, E)$ and integer k, the clique problem is to find out if G has a clique of size k. This problem is known to be NP-complete.

Given $G = (V, E)$ and k, the scheduling instance is created as follows.

- There are m identical machines where $m = \frac{1}{2}n(n-1) + 1$ and $n = |V|$.
- There are $3m$ jobs with unit processing time.
- Jobs are grouped into graph jobs and dummy jobs where graph jobs are from all vertices and edges of G, and dummy jobs are additionally introduced as three sets D_1, D_2 and D_3 which respectively have

$$|D_1| = m - k, \tag{4}$$

$$|D_2| = m - \frac{1}{2}k(k-1) - n + k, \text{and} \tag{5}$$

$$|D_3| = m - |E| + \frac{1}{2}k(k-1) \tag{6}$$

  jobs.

- The precedence constraints are constructed as follows.
  1. The jobs from vertices connected by an edge are predecessors for the job from that edge. (For edge $e = uv$, jobs J_u and J_v are predecessor to J_e.)
  2. Each job in D_1 is a predecessor to each job in D_2.
  3. Each job in D_2 is a predecessor to each job in D_3.

In this scheduling instance, it is clear that $\mathrm{OPT} \geq 3$ even if there are only dummy jobs.

We have the following claims.

Claim 7 *If G has a clique of size k, then* $\mathrm{OPT} = 3$.

Proof: Consider 3 time slots. The jobs from clique vertices are scheduled in the first slot since they are all predecessors to edge jobs. In the second slot, the edges of the clique are scheduled. For the remaining vertices and edges, the vertex jobs are scheduled in the second slot in addition to the edges of the clique. This allows the remaining edge jobs to be scheduled in the last slot. Regarding dummy jobs, each job in D_i is scheduled in ith slot. They ensure that all the machines are always busy during three slots, thus giving us $\mathrm{OPT} = 3$. □

Claim 8 *If* $\mathrm{OPT} = 3$*, then G has a clique of size k.*

Proof: Suppose G does not have a clique of size k. We want to show that all jobs in this case cannot be scheduled in 3 time slots ($\mathrm{OPT} > 3$).

To use only 3 time slots, all dummy jobs in D_1, D_2 and D_3 should be scheduled in time slot 1, 2 and 3, respectively. Then, the time slots respectively have k, $\frac{1}{2}k(k-1) + (n-k)$, $|E| - \frac{1}{2}k(k-1)$ idle machines to schedule all edge and vertex jobs; since the total amount of processing time remaining is $n + |E|$, each slot must be used fully if we are to complete the jobs in 3 time slots.

Since no edge job can be scheduled in the first slot, only k vertex jobs corresponding to a set of vertices V' are scheduled. Now, the only edge jobs that can be scheduled in the second slot are those corresponding to edges that have both their endpoints in V'. But since V' is not a clique, there are strictly fewer than $\frac{1}{2}k(k-1)$ such edge jobs, and even if all the remaining $(n-k)$ vertex jobs are scheduled in the second time slot, some machine must be idle in this slot. Thus, we cannot schedule all the jobs in 3 slots. $\square$