Approximation Algorithms - Tools and Techniques **Spring 2002**

Lecture 4 : Jan 23, 2002

Lecturer: Kamesh Munagala[1] *Scribe: Siddartha Y R*[2]

0.1 The Minimum MAKESPAN Problem

Given a set of identical processors $\{P_1, P_2, \ldots, P_n\}$ and a set of jobs $\{J_1, J_2, \ldots, J_m\}$ whose corresponding processing times are $\{p_1, p_2, \ldots, p_m\}$, to find an assignment of jobs to processors which minimises the MAKESPAN - the time at which all the processors have completed all the jobs assigned to them.

For arbitrary p_i s this problem is known to be $\mathcal{NP} - Hard$. In the following, let C denote the value of the MAKESPAN.

Algorithm 1 GreedyMakespan

Input : A set of job $\{J_1, J_2, \ldots, J_m\}$, their processing times $\{p_1, p_2, \ldots, p_m\}$ and a set of processors $\{P_1, P_2, \ldots, P_n\}$.
Output : An assignment of jobs to processors which minimises the MAKESPAN.

1: order the jobs in any arbitrary way.
2: **while** there are unassigned jobs **do**
3: assign the next job to the next processor that becomes free.
4: **end while**

Analysis of GreedyMakespan

Let C^* represent the optimal MAKESPAN. Two obvious lower bounds on C^* are as follows. C^* is not less than the average load on each processor given by

$$C^* \geq \frac{\sum_i p_i}{n}$$

[1]Dept. of Computer Science, Stanford University
[2]Dept of Comp. Sci. and Automation, Indian Institute of Science, Bangalore.
siddu@csa.iisc.ernet.in

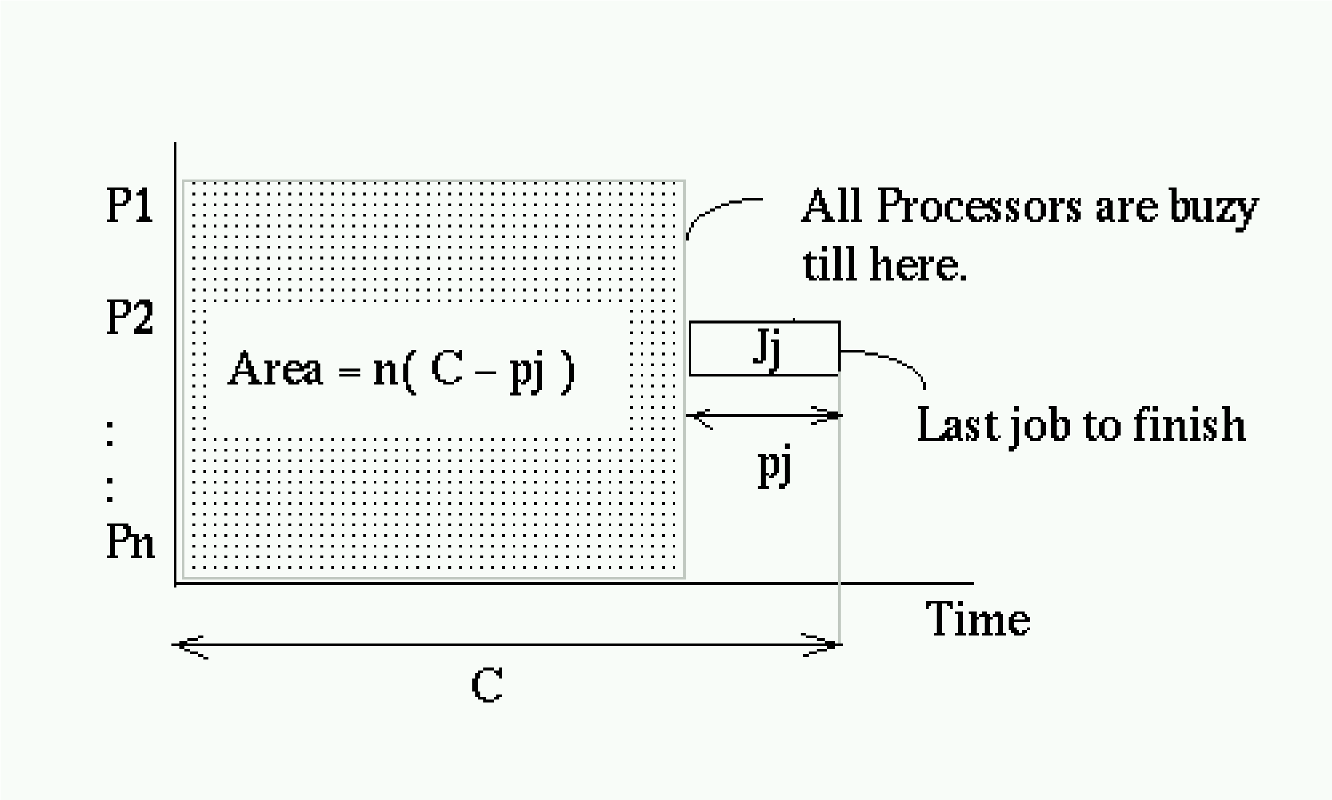

Figure 1: GreedyMakespan

and it is atleast as long as each of the jobs

$$C^* \geq p_i \quad \forall i.$$

Let the job that finishes last be J_j. All the processors are buzy till atleast the time at which J_j was started (else J_j would've been scheduled on the machine which became free earlier). The shaded area in Figure 1 represents this. Thus, the area of this shaded region gives a lower bound on the total processing time of all the jobs except the job J_j. We see that

$$\begin{aligned} n(C - p_j) \leq & \sum_i p_i - p_j \\ \Rightarrow \quad (C - p_j) \leq & \frac{\sum_i p_i}{n} - \frac{p_j}{n} \\ \Rightarrow \quad C \quad \leq & \frac{\sum_i p_i}{n} + p_j\left(1 - \frac{1}{n}\right) \end{aligned}$$

Then, using the two lower bounds on C^* from Equations. 0.1 and 0.1 above, we have

$$\begin{aligned} C \leq & \ C^* + C^*\left(1 - \frac{1}{n}\right) \\ C \leq & \ C^*\left(2 - \frac{1}{n}\right) \end{aligned}$$

giving an essentially constant factor approximation ratio of $\left(2 - \frac{1}{n}\right)$. See from the figure that a bad instance of this problem would have a large(st) job as J_j. This observation leads us to an improved algorithm, discussed next.

An Improved Greedy Algorithm

Since a long job occuring at the end of the list of jobs can lead to a bad assignment, the greedy algorithm can be improved by sorting the job list in decreasing order of processing time before running GreedyMakespan. LPT stands for Longest Process First

Algorithm 2 LPTMakespan

Input : A set of job $\{J_1, J_2, \ldots, J_m\}$, their processing times $\{p_1, p_2, \ldots, p_m\}$ and a set of processors $\{P_1, P_2, \ldots, P_n\}$.
Output : An assignment of jobs to processors which minimises the MAKESPAN.

1: sort the jobs in decreasing order of processing times.
2: **while** there are unassigned jobs **do**
3: assign the next job to the next processor that becomes free.
4: **end while**.

Theorem 1 LPTMakespan *is a $\frac{4}{3}$ aprroximation alogrithm.*

Proof: Assume that the job that finishes last, say J_j, is also the job which starts last, in the assignment returned by LPTMakespan (see box below). The proof considers two cases depending on the value of p_j.

If, in the assignment returned by LPTMakespan, the above property is not satisfied. Then there exists job(s) which start after J_j but end before J_j. Throw away all such jobs and establish the above theorem. By throwing away such jobs, the MAKESPAN of LPTMakespan is unchanged (the optimum solution on the modified problem can only become smaller). Thus, any approximation ratio derived for this special case will also hold for (is only greater than that of) the general case

Case 1: $p_j \leq \frac{C^*}{3}$.

From the analysis of GreedyMakespan, we have the inequality

$$C \leq \frac{\sum_i p_i}{n} + p_j \left(1 - \frac{1}{n}\right).$$

From the lower bound established on C^* before,

$$C \leq C^* + \frac{C^*}{3}\left(1 - \frac{1}{n}\right),$$

which gives the required result

$$C \leq \frac{4}{3}C^*.$$

Case 2: $p_j > \frac{C^*}{3}$

From our assumption, J_j is the last job to start. Hence J_j is also the (last job on the list and therfore the) smallest job. Hence $p_i > \frac{C^*}{3}$ (note strict inequality) for all jobs. Hence, in any schedule, each processor is assigned no more than two jobs. With this restriction, it can be shown that the optimal schedule and the schedule returned by LPTMakespan are identical.

Exercise 1 *Show that, for the restricted case of schedules with no more than two jobs per processors, the optimal schedule is obtained by the following algorithm.*

Algorithm 3 TwoJobOptSched

Input : A set of job $\{J_1, J_2, \ldots, J_m\}$, their processing times $\{p_1, p_2, \ldots, p_m\}$ and a set of processors $\{P_1, P_2, \ldots, P_n\}$. $m \leq 2n$

Output : An assignment of jobs to processors which minimises the MAKESPAN.

sort list of jobs in decreasing order of processing times.
if *number of jobs* $\neq 2\times$ *number of processors* ***then***
    pad list with jobs of zero processing time to obtain $2n$ *jobs.*
end if
pair the extreme (first and last) jobs in the list successively (by removing the pair from the list) to obtain a schedule.
done.

Exercise 2 *Show that, for the above restricted case,* LPTMakespan *always returns a schedule identical to the schedule returned by the above algorithm.*

The above exercises show that LPTMakespan returns an optimal schedule for Case 2. This along with the approx ratio shown for Case 1, establishes the theorem.

■

Exercise 3 *Give an instance of the* MAKESPAN *problem for which the approximation ratio* $\frac{4}{3}$ *is exact.*