Scheduling: Open Shop Problems

R. Chandrasekaran

0.0.1 O2|$C_{\max}$|Non-preemptive:

This section deals with *open shop* problems. Here we have a set of jobs and a set of machines. Each job requires processing by each machine (if some jobs do not require processing by a machine, then we set the appropriate time to zero). Let $p_{j,i}$ denote the duration job i requires on machine $j; 1 \leq i \leq n; 1 \leq j \leq m$. In this section we consider $m = 2$, and when no preemption is allowed. We wish to minimize the makespan.

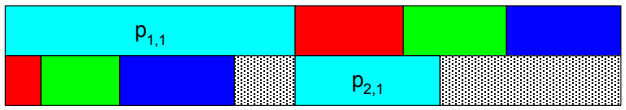
Let

$$\alpha = \max[\sum_{i=1}^{n} p_{1,i}; \sum_{j=1}^{n} p_{2,i}; \max_{i=1}^{n}\{p_{1,i} + p_{2,i}\}]$$

Clearly, $C_{\max}(S) \geq \alpha \ \forall \ S$. (Actually a similar result is true for all m). We show equality for $m = 2$. Without loss of generality, let us suppose that $\sum_{i=1}^{n} p_{1,i} \geq \sum_{j=1}^{n} p_{2,i}; p_{1,1} + p_{2,1} = \max_{1 \leq i \leq n}\{p_{1,i} + p_{2,i}\}$.

Case (i): $p_{1,1} + p_{2,1} \geq \min[\sum_{i=1}^{n} p_{1,i}, \sum_{i=1}^{n} p_{2,i}] = \sum_{i=1}^{n} p_{2,i}$

This implies that $p_{1,1} \geq \sum_{i \neq 1} p_{2,i}$. Consider the schedule shown in the figure below:



This schedule has length equal to $\max[p_{1,1} + p_{2,1}; \sum_{i=1}^{n} p_{1,i}] = \alpha$.

Case (ii): $p_{1,1} + p_{2,1} < \min[\sum_{i=1}^{n} p_{1,i}, \sum_{i=1}^{n} p_{2,i}] = \sum_{i=1}^{n} p_{2,i}$

Let $\Delta = \sum_{i=1}^{n} p_{1,i} - \sum_{i=1}^{n} p_{2,i} \geq 0; p'_{2,1} = p_{2,1} + \Delta; \ p'_{j,i} = p'_{j,i}$ for $i \neq 1; \forall \ j; p'_{1,1} = p_{1,1}$.

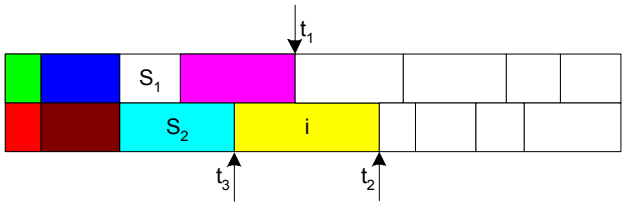
$$\sum_{i=1}^{n} p'_{2,i} = \sum_{i=1}^{n} p'_{1,i} = \alpha \geq p'_{1,1} + p'_{2,1} = \max_{i=1}^{n}\{p'_{1,i} + p'_{2,i}\}$$

Thus, if we show the result for this case, it follows for other cases. Note that the value of α has *not* changed.

Now we use the LAPT — Largest Alternate Processing Time – Rule for obtaining the schedule. (Actually this works for the previous case as well; it was easier to prove this way).

Whenever a machine is free, load the job which requires the largest processing time on the OTHER machine from among all jobs that still have not been processed by either machine (if they are available). Then process the remaining jobs in the same order that they were processed on the other machine.

Let S_j be the set of jobs which are processed first by machine $j; 1 \leq j \leq 2$. Let t_j be the instant of time when all of the jobs in the set S_j finish on machine $j; 1 \leq j \leq 2$. See figure below:



If $|S_2| = 1$, then we have a case similar to (i). So, we suppose that this is not the case. At instant t_1, there can be no jobs that require processing on both machines (else, the algorithm would load such a job on machine 1, contradicting the definition of the set S_1). Hence all jobs in the set S_2 *except possibly* the last one, can be processed on machine 1 at t_1. Clearly, all jobs in S_1 can be processed on machine 2 at t_2. Let t_3 be the last instant a job in S_2 gets loaded on machine 2 ($t_3 < t_1$); and let this job be i. To show that the last job in S_2 can also be processed on machine 1 without an idle period, we need to show that

$$\sum_{k \in S_2; k \neq i} p_{1,k} \geq t_2 - t_1$$

Recall

$$\sum_{k \in S_2; k \neq i} p_{1,k} + p_{1,i} = \alpha - \sum_{k \in S_1} p_{1,k} = \alpha - t_1$$

The rule LAPT selects jobs so that the processing time on the other machine is the largest. Since $|S_2| > 1$,and all jobs in S_2 other than i were selected before i by LAPT, we have

$$\sum_{k \in S_2; k \neq i} p_{1,k} \geq p_{1,i}$$

Therefore,

$$\sum_{k \in S_2; k \neq i} p_{1,k} \geq \frac{\alpha - t_1}{2}$$

Since jobs not in S_2 [which are in S_1] have larger processing time on machine 2 than those in S_2, with the possible exception of job i, because we are using LAPT rule,

$$\alpha - t_2 \geq t_2 - t_3 = \sum_{k \in S_2 - \{i\}} p_{2,k}$$

Hence

$$t_1 + \sum_{k \in S_2; k \neq i} p_{1,k} \geq \frac{\alpha + t_1}{2} \geq \frac{\alpha + t_3}{2} \geq t_2$$

Hence, none of the jobs in S_2 have idle times on machine 1. Since none of the jobs in S_1 have idle times on machine 2, the schedule length equals $\max[\sum_{i=1}^{n} p_{1,i}; \sum_{i=1}^{n} p_{2,i}] = \alpha$. Hence the schedule is optimal.

Now we turn to preemptive schedules.

0.0.2 O2|$C_{\max}$|Preemptive:

Clearly $C_{\max} \geq \max[\max_{j=1}^{m} \sum_{i=1}^{n} p_{j,i}; \max_{i=1}^{n} \sum_{j=1}^{m} p_{j,i}] = \alpha$. We show that this value can always be achieved. We demonstrate this by an example which clearly illustrates the proof of its optimality as well.

Example 1 $m = 3; n = 4$:

15		30*	15			
15			40*		5	
15	40*		5			
15*				45		
	20			15*	25	
		30			30*	
						60*

15*		15	15			
15			25*		5	
15	25*		5			
				45*		
	20				25*	
		30*			15	
						45*

		15*	15			
15			10*		5	
15*	10		5			
				30*		
	20*				10	
		15			15*	
						30*

		5	15*			
15*					5	
5	10*		5			
				20*		
	10				10*	
		15*			5	
						20*

		5*	5			
5*					5	
5			5*			
				10*		
	10*					
		5			5*	
						10*

			5*			
					5*	
5*						
				5*		
	5*					
		5*				
						5*

J3	J1	J3	J4	J3	J4
J4	J4	J4	J1	J1	J6
J2	J2	J1	J2	J4	J1
J1	J5	J5	J5	J5	J5
J5	J6	J2	J6	J2	J2
J6	J3	J6	J3	J6	J3
J7	J7	J7	J7	J7	J7

Clearly, this schedule is optimal. Some improvements can be made as shown by the following example;

Example 2 $m = 6; n = 6$ *But this can be reduced to the previous example with lower values for* m *and* n *by condensing jobs and machines as shown below:*

$$\begin{bmatrix} 5 & 10 & & & & 15 \\ & & & & 30 & \\ 10 & 5 & & & & 40 \\ & & 10^* & 5^* & & \\ & & 10^* & 15^* & & \\ 10 & 5 & & & & 5 \end{bmatrix}$$

By combining machines $\{1,2\},\{3\},\{4,5,6\}$ into three super machines and jobs $\{1,2\},[3,4\},\{5\},\{6\}$ into four super jobs, we get the previous example. (Note that there are better combinations possible; however, finding the best is an NP-hard problem.) The durations are the total of these respective jobs on these respective machines. Finally we unscramble the schedule as shown below: For example the jobs $\{3,4\}$ on machines $\{4,5\}$ have a total of 40 units of processing. This corresponds to super job 2 on super machine 3. The 40 units are arbitrarily broken down to suit the original jobs on the original machines with no overlapping.

Now we can use this model to also solve the parallel machine case (both identical and unrelated) with non-overlapping preemption. We show the more general case of unrelated machines.

Recall the LP:

$$\begin{gathered} \min z \\ \textstyle\sum_{i=1}^{n} t_{j,i} x_{j,i} \leq z; 1 \leq j \leq m \\ \textstyle\sum_{j=1}^{m} t_{j,i} x_{j,i} \leq z; 1 \leq i \leq n \\ \textstyle\sum_{j=1}^{m} x_{j,i} = 1; 1 \leq i \leq n \\ x_{j,i} \geq 0 \end{gathered}$$

Here the variables $x_{j,i}$ denote the fraction of the job i done on machine j. Once we get the values of the variables in an optimal solution, we can calculate the duration $t_{j,i}x^*_{j,i}$ required for each job on each machine. Now if we let $p_{j,i} = t_{j,i}x^*_{j,i}$ and find an open shop preemptive schedule, we get the required answer to the original problem.