Scheduling: Flowshops

R. Chandrasekaran

February 4, 2020

0.1 Problem Statement:

Given n jobs each of which requires m operations that are to be done on m different machines (one on each), with a specified order which is the same for all jobs is called a *flowshop*. Let $t_{i,j}$ be the time required for processing of operation j of job i.

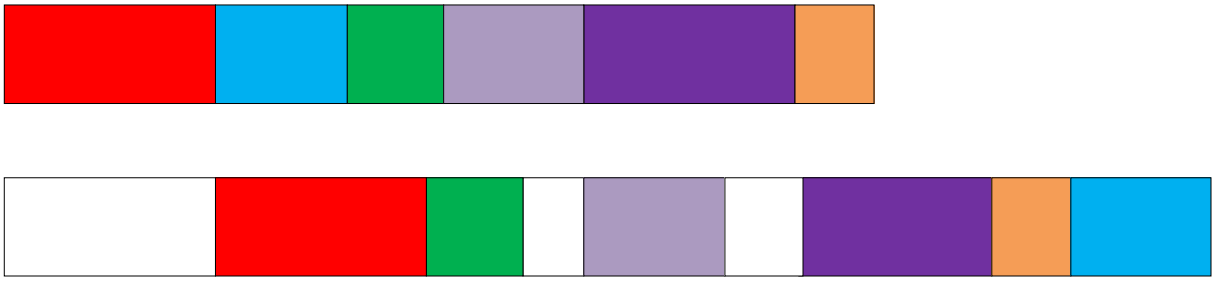
We consider the problem of minimizing makespan, i.e $F_{\max}$. We assume that all jobs are available at time 0. We also assume that there is unlimited storage at intermediate facilities. It is not at all clear that there is an optimal schedule in which the order of processing jobs on each machine is the same. Schedules that restrict so that this is true are called *permutation schedules*. For $m = 2$ there indeed do exist optimal schedules that are permutation schedules. This is shown as follows:

Lemma 1 *There exist an optimal schedule for a flowshop with $m = 2$, that minimizes makespan in which there is no preemption or inserted idle time.*
Proof. *Suppose an optimal schedule has premptions on first amchine such as in the first of the figures below, we can consolidate all pieces of the same job to its last pience and slide all pieces of all jobs in between to the elft without affecting the validity of the schedule or its makespan. Hence there is an optimal schedule with no preemptions on the first machine.*

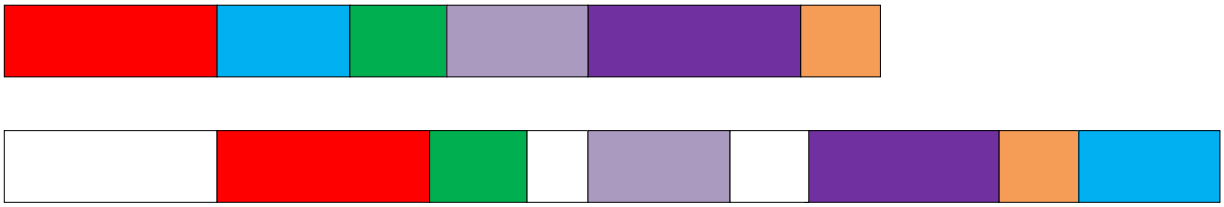
Simliar consolidation to the first piece as shown can be done on the last machine.

Hence the result follows for $m = 2$. ■

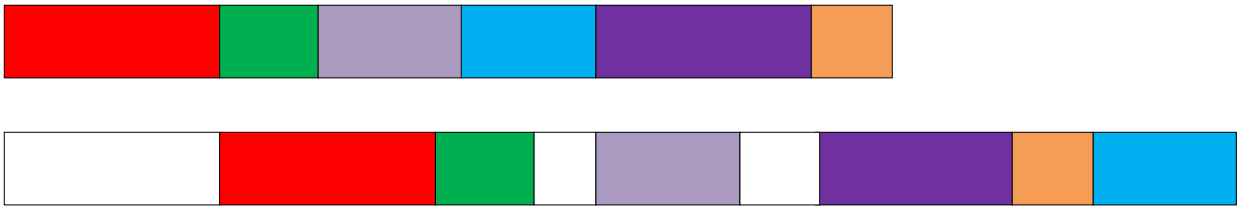
Lemma 2 *Considering only nonpreemptive schedules, there exists an optimal schedule for flowshop problems to minimize makespan in which jobs are processed in the same order on the first two machines (and on the last two machines).*
Proof. *Suppose we have an optimal schedule in which the order is not the same on the first two machines as shon below:*



Change the schedule on the first machine so that the order is the same as follows and compress on machine 2 if that is possible. :



Interchange these two jobs as shown below:



Similar arguments show the process for the lst wo machines. Hence, for $m = 2, 3$ we get a permutation schedule for an optimal solution if no preemption is allowed. For $m = 2$ we know there is optimal schedule that has no preemption. Hence we can restrict our attention to permutation schedules for $m = 2$. ■

[S. Johnson, 1954]: $m = 2, F_{\max}$:

1. $S_1 \leftarrow \{i : p_{i,1} = A[i] \leq p_{i,2} = B[i]\}; S_2 \leftarrow \{i : p_{i,1} = A[i] > p_{i,2} = B[i]\}//S_1 \cup S_2 = \{1, 2, ..., n\}; S_1 \cap S_2 = \phi$

2. Sort elements of S_1 in nondecreasing order of values of $A[i]$. Let this order be indeicated by T_1//"ties" are broken abribitrarily

3. Sort elements of S_2 in nonincreasing order of $B[i]$. Let this order be indeicated by T_2//"ties" are broken abribitrarily

4. Process the jobs on both machines in the order $[T_1, T_2]$

   Complexity of the algorithm: $\Theta(n \lg n)$.

Before proving correctness of the algorithm, we can also state the algorithm in a slightly different manner so as to show that it si an example of "greedy algorithm".

At each step, find the lest processing time (on either machine) of the jobs yet to be processed; if that value corresponds to machine 1 place the jbow in the next position from front; if it happens to be on machine 2 place the job in the next position from the rear. Again "ties" are broken arbitrarily.

Now to prove correctness of the algorithm.

Let permutation schedule be denoted by $[1], [2], ..., [n]$. Here is how the "Gantt" chart might look like for this permutation schedule:

Here $X[i]$ indicates the idle time on machine 2 before processing job i. The following expressions for these values are easily checked:

$$\begin{aligned} X[1] &= A[1] \\ X[2] &= \max[A[1] + A[2] - B[1] - X[1], 0] \\ X[j] &= \max[\sum_{i=1}^{j} A[i] - \sum_{i=1}^{j-1} B[i] - \sum_{i=1}^{j-1} X[i], 0] \end{aligned}$$

From this we can obtain;

$$\begin{aligned} \sum_{i=1}^{j} X[i] &= \max_{k=1}^{j}[\sum_{i=1}^{k} A[i] - \sum_{i=1}^{k-1} B[i]] \\ &= \max_{k=1}^{j} Y[k] \end{aligned}$$

where $Y[k] = \sum_{i=1}^{k} A[i] - \sum_{i=1}^{k-1} B[i]$ and $B[0] = 0$. For any permutation schedule S

$$F_{\max}(S) = \sum_{i=1}^{n} B[i] + \sum_{i=1}^{n} X[i]$$

$$= \sum_{i=1}^{n} B_i + \max_{k=1}^{n} Y[k]$$

However, $\sum_{i=1}^{n} B_i$ is a constant independent of the schedule. Hence $\min_S F_{\max}(S)$ is equivalent to minimizing $\max_{k=1}^{n} Y[k]$. Please note that $[k]$ depends on S.

Approximation to m-Machine Flow Shops

1. Vector Summation in Banach Space and polynomial Algorithms for Flow Shops and Open Shops, by S. Sevest'janov, Mathematics of Operations Research, 20, (1995), pp. 90-103

   See the reference to I. Barany 1981 paper in the above.

Theorem 3 (Barany) *For a finite set of vectors* $\mathbf{V} = \{v_1, v_2, ..., v_n\} \subseteq R^d$ *with* $\sum_{i=1}^{n} v_i = 0$ *and* $\|v_i\| \leq 1$ *for* $i = 1, 2, ..., n$ *there exists a permutation* $i_1, i_2, ..., i_n$ *of* $\{1, 2, ..., n\}$ *such that*

$$\max_{1 \leq k \leq n} \left\| \sum_{j=1}^{k} v_{i_j} \right\| \leq \frac{3}{2} d$$

Moreover, this permutation can be found in $O(n^2 d^3 + nd^4)$ *steps.*

The original form of this theorem is known as *Compact Vector Summation* (CVS) thoerem due to Steinitz (1913) where instead of $\frac{3}{2}d$, was a function $\phi(d)$. Sevestyanov (1978) has d instead of $\frac{3}{2}d$ with slightly different complexity.

First we will show the application of this reult to get approximate solution to flowshops. Consider a m-machine n-job flow shop. Let $t_{i,j}$ represent the processing time of job i on machine j. Suppose $t_{i,j} \leq K$ for all i and j. Let $M_j = \sum_{i=1}^{n} t_{i,j} \leq nK$ for all j; and $M = \max_{j=1}^{m} M_j \leq nK$.

Theorem 4 *There exists a permutation schedule for which the finish time* T *satisfies the inequalties:*

$$M \leq T \leq M + (m-1)[\frac{3m-1}{2}]K$$

Moreover this schedule can be obtained in $O(n^2 m^3 + nm^4)$ *steps.*

Proof. *(Using Theorem above): First we construct a fictitious problem with same number of jobs and machines for which the processing times are gien by* $t'_{i,j}$ *where*

$$\begin{aligned} t_{i,j} &\leq t'_{i,j} \leq K \text{ for all } i, j \\ \sum_{i=1}^{n} t'_{i,j} &= M \text{ for all } j \end{aligned}$$

This is easy to in $O(nm)$ steps. Given a permutation $i_1, i_2, ..., i_n$ of jobs, the finish time for this schedule is given by

$$T = \max_{1=k_0<k_1<k_2<...<k_m=n} \left[\sum_{j=1}^{m-1} \sum_{s=k_j}^{k_j+1} t_{i_s,j+1} \right]$$

Hence

$$T' = \max_{1=k_0<k_1<k_2<...<k_m=n} \left[\sum_{j=1}^{m-1} \sum_{s=k_j}^{k_j+1} t'_{i_s,j+1} \right]$$

Clearly $M \leq T \leq T'$. ■