J. Operation Research Soc. of Japan
Vol. 14, No. 1, June 1971.

OPTIMAL FLOWSHOP SCHEDULING WITH DUE DATES AND PENALTY COSTS

JATINDER N.D. GUPTA

Assistant Professor, Division of Engineering
The University of Alabama in Huntsville

(Received December 14, 1970 and Revised February 8, 1971)

Abstract

The classical n-job, M-machine flowshop scheduling problem is considered where jobs are to be delivered by certain pre-assigned due dates, failing which there is some penalty cost associated with the time by which a job is delivered late. Based on the concepts of lexicographic search, an algorithm is presented for the solution of the problem under the assumption that the same ordering of jobs is followed on all machines and the penalty cost for any job is a non-negative and non-decreasing function of time by which the job is late.

1. Introduction

Shops of multiple machines in which work flow is undirectional are called *flowshops*. The general scheduling problem in flowshops is one of determining the order (schedule) in which a given number of jobs should be processed on a given number of machines so as to optimize (minimize or maximize) certain specified measure of performance. This problem was first formulated by Johnson [3] as an n-job, 2-machine problem when the objective function is to minimize the throughput time (called make-span) of all jobs. Subsequent developments in scheduling theory have been extensions of Johnson's formulation, in that the number of machines is increased to the general case M. However, the formulation of flowshop scheduling problem, as an extension of Johnson's is not the general problem encountered in practice [2, 4]. Even with the same formulation, other measures of performance exist which are more important than the make-span criterion [2]. Thus, for example, the jobs may have to be delivered by specified time and a failure to meet the due dates may result in a penalty cost, depending on the time by which a job is delivered late. Because of varied nature of jobs and customers, the penalty costs for different jobs are usually different. Thus, in such cases, the minimization of total penalty cost reflects a better measure of performance than make-span [2, 4].

2. Problem Definition and Analysis

The flowshop scheduling problem considered here may be stated as follows:

"Given n jobs to be processed on M machines in the same order, the process time of job a on machine m being t_{am}, the due date of job a is d_a and the penalty cost function for job a is $g_a(s)$ where s is the time by which job a is delivered late, $(a=1, 2, \cdots, n;\ m=1, 2, \cdots M)$; it is desired to find the common order (schedule) in which these n jobs should be processed on the M machines so as to minimize the total penalty cost."

With the above definition of the problem, assume that a partial schedule (initial part of a complete schedule) σ is available and job a is being augmented to this partial schedule σ, represented by σa. Then, from the physical conditions of the problem (non-interference at machines and non-simultaneous processing of jobs), the completion time of the partial schedule σa at machine m, $T(\sigma a, m)$, is given by the following recursive relation [1]:

(1) $$T(\sigma a, m)= \max [T(\sigma a, m-1);\ T(\sigma, m)]+t_{am} \qquad m=1, 2, \cdots, M$$

where

$$T(\Phi, m)= T(\sigma a, 0)=0;\ \forall \sigma \text{ and } m .$$

Let the total penalty cost of the partial schedule σ be $c(\sigma)$. Then, the total penalty cost of the partial schedule σa can be obtained as under:

(2) $$c(\sigma a)=c(\sigma)+g_a(s_a)$$

where

(3) $$s_a= \max [T(\sigma a, M)-d_a, 0] \text{ and } c(\Phi)=0$$

Then the scheduling problem, as outlined above, is that of finding the order of jobs so as to minimize $c(\sigma a)$ where σ ranges over all the possible permutation of $(n-1)$ jobs not containing job a and job a ranges from 1 through n.

The flowshop scheduling problem, as analyzed above, is a quantitative combinatorial search problem and permits of a finite number of feasible schedules, in fact equal to $n!$. However, as the number of jobs increase, the corresponding increase in the total number of feasible schedules is so large that a direct enumeration is economically impossible. Thus, in order to make the problem tractable, suitable schemes have to be designed so that complete enumeration is not required. By exploiting the structure of the problem, certain dominance procedures can be developed which, when coupled with the lexicographic search procedure, help solve the problem in far less number of trials than complete enumeration.

3. Mathematical Developments

The lexicographic search approach to be proposed here is based on the possibility of listing (at least conceptually) the solutions or related configurations in a structural hierarchy which also reflects a hierarchical ordering of the corresponding values of these configurations [1]. Based on certain dominance procedures, the search over this conceptual list can easily select the solution with the requisite value. Thus, if a partial schedule σ is being considered and a complete schedule S is available with a penalty cost $c(S)$ then the penalty cost of the partial schedule σ, $c(\sigma)$, can be used to skip blocks of solutions in the conceptual list of solutions and obtain the optimal schedule at an early stage of search. The following mathematical developments outline the search methodology and establish the proof of optimality of the algorithm proposed in the subsequent sections.

Lemma 1:

Consider a partial schedule σ and a complete schedule S. If:

(4) $c(\sigma) \geqq c(S)$

then:

(5) $c(\sigma\pi) \geqq c(S)\,; \ \forall \pi \notin \sigma$

where π is a post partial sequence of jobs not contained in σ.

Proof:

Since σ does not form a complete schedule, some jobs are still to be augmented to form a complete schedule. But the penalty cost associated with any job a is a non-decreasing function of time and is non-negative. Thus, it follows that:

(6) $c(\sigma\pi) \geqq c(\sigma), \ \forall \pi \notin \sigma$

The results of the lemmal, therefore, follow from relations (4) and (6).

Based on lemmal, the following corollary follows:

Corollary 1: If $c(S)=0$, then, there is no other schedule for which the penalty cost is less than $c(S)$.

Theorem 1: Consider a partial schedule σ and a complete schedule S. If:

(7) $c(\sigma a) \geqq c(S)$

then:

(8) $c(\sigma\pi) \geqq cS) \ \forall \pi \notin \sigma$

where π is the post partial sequence of jobs not contained in σ.

Proof:

Let $\pi = b\pi' a\pi''$, which implies that job b is augmented to σ instead of job a. Considering the partial schedule σ, it is seen that either (i) $c(\sigma b) < c(S)$ or (ii) $c(\sigma b) \geqq c(S)$.

Consider case (i) first. Augmentation of job b to σ implies that job a has to occupy some later sequence position to complete a schedule. However, as can be seen from equation (1) above,

(9) $T(\sigma b\pi' a, M) \geqq T(\sigma a, M)$

Hence relation (8) follows from equations (2), (3), (7), (9), and lemma 1.

Similarly, for case (ii), relation (8) follows from lemma 1. Since job b is arbitrary, the theorem is true for all π.

As a result of the above theorem, it is seen that if $\sigma = \Phi$, then S is optimal. This can be stated as a corollary to theorem 1:

Corollary 2: *Let $\sigma = \Phi$ (empty) and $c(\sigma a) \geqslant c(S)$. Then for any Schedule S';*

(10) $c(S') \geqq c(S)$.

4. The Search Algorithm

The search procedure described here generates complete schedules in accordance with Theorem 1 and corollaries 1 and 2 above. This algorithm may be considered to be an iterative procedure where, starting from a trial solution, search is continued—like words in a dictionary—to improve the value of the solution. Use of the dominance established above avoids the search through all the $n!$ possible permutations though the search is exhaustive enough to generate an optimal schedule.

Let a complete schedule S be available with penalty cost $c(S)$. Consider a partial schedule σ and the possibility of augmenting job a to σ. According to theorem 1, if $c(\sigma a) \geqq c(S)$, then, there is no schedule with σ as its leader (initial partial schedule) which is better than S. Hence, in order to find a solution better than S, the last job of σ must be removed and search be continued lexicographically with the partial schedule so obtained. Thus if $\sigma = \sigma' b$, then the above condition implies that the search be continued with σ'. Further, since the order of search is lexicographic, only jobs greater than b should be considered for immediate augmentation to σ'. (see steps 5 and 6 below). However, if $c(\sigma a) < c(S)$, then no specific conclusions can be drawn and job a should be augmented to σ (see step 3 below).

The iterative procedure requires a complete schedule S and its associated penalty cost $c(S)$. For the general case, it is not possible to give any method to get a feasible solution. For some special cases, a good trial solution can be obtained. Thus, in the general case, the trial value δ may be set to infinity and the first complete schedule is considered as the trail solution. However, wherever a trial solution is available (as in the case of linear cost functions) it should be used. The following step by step procedure results in generating on optimal schedule:

Step 1: Let $\sigma = \Phi$, $a=1$, $c(\sigma)=0$, $L=0$, and $\delta = \infty$ Enter step 2.

Step 2: Compute $c(\sigma a)$. Is $c(\sigma a) < \delta$?

a. Yes. Go to step 3.

b. No. Go to step 4.

Step 3: Lengthen the partial schedule by augmenting a to σ. Set $L = L+1$, $\sigma = \sigma a$. Is $L = n$?

a. Yes. Go to step 5.

b. No. Let a be the next available job. Return to step 2.

Step 4: Is $L=0$?

a. No. Go to step 6.

b. Yes. Go to step 7.

Step 5: Set $\delta = c(\sigma)$, $S = \sigma$. Is $\delta = 0$?

a. Yes. Go to step 7.

b. No. Set $L = L - 1$. Go to step 6.

Step 6: Let the last job of σ be b and $\sigma = \sigma' b$. Set $L = L - 1$ and $\sigma = \sigma'$. Is $b = n$?

a. Yes. Return to step 4.

b. No. Set a as the first available job greater than b. Return to step 2.

Step 7: Stop the search. The last complete schedule S is optimal with Penalty cost δ.

5. Numerical Illustration

The working of the above algorithm is explained by solving the following 4-job, 5-machine problem of Table 1 where $g_a(s_a) = p_a s_a$.

TABLE 1

Problem Data

a \ m	1	2	3	4	5	d_a	p_a
1	4	3	7	2	8	28	2
2	3	7	2	8	5	33	3
3	1	2	4	3	7	17	4
4	3	4	3	7	2	20	5

The trial solution, obtained by arranging the jobs in the descending order of p_a is 4321 with Penalty cost $\delta = 60$.

The working of the algorithm may be set up in a tabular form as shown in Table 2. The calculations of Table 2 are self explanatory and result in an optimal schedule 3412 with zero Penalty cost.

6. Computational Experience

Apart from empirical investigation and numerical comparision, there seems to be no other method to test the efficiency of any combinatorial

TABLE 2
The Search Table

σ	L	a	$T(\sigma a, m)$					$c(\sigma a)$	$\gtreqless \delta$	Comments
			1	2	3	4	5			
Φ	0	1	4	7	14	16	24	0	$<$	$\sigma=1,\ L=1,\ a=2$
1	1	2	7	14	16	24	29	0	$<$	$\sigma=1,\ L=2,\ a=3$
12	2	3	8	16	20	27	36	76	$>$	$\sigma=1,\ L=1,\ a=3$
1	1	3	5	9	18	21	31	56	$<$	$\sigma=13,\ L=2,\ a=2$
13	2	2	8	16	20	29	36	65	$>$	$\sigma=1,\ L=1,\ a=4$
1	1	4	7	11	17	24	26	30	$<$	$\sigma=14,\ L=2,\ a=2$
14	2	2	10	18	20	32	37	42	$<$	$\sigma=142,\ L=3,\ a=3$
142	3	3	11	20	24	35	44	150	$>$	$\sigma=14,\ L=2,\ a=3$
14	2	3	8	13	21	27	32	90	$>$	$\sigma=\Phi,\ L=0,\ a=2$
Φ	0	2	3	10	12	20	25	0	$<$	$\sigma=2,\ L=1,\ a=1$
2	1	1	7	13	20	22	30	4	$<$	$\sigma=21,\ L=2,\ a=3$
21	2	3	8	15	24	27	33	68	$>$	$\sigma=2,\ L=1,\ a=3$
2	1	3	4	12	16	23	32	60	$=$	$\sigma=\Phi,\ L=0,\ a=3$
Φ	0	3	1	3	7	10	17	0	$<$	$\sigma=3,\ L=1,\ a=1$
3	1	1	5	8	15	17	24	0	$<$	$\sigma=31,\ L=2,\ a=2$
31	2	2	8	15	18	24	30	0	$<$	$\sigma=312,\ L=3,\ a=4$
312	3	4	11	19	22	29	31	55	$<$	$S=3124,\ L=2,$ $\sigma=31,\ a=4,\ c(S)=55$
31	2	4	8	12	18	25	27	35	$<$	$\sigma=314,\ L=3,\ a=2$
314	3	2	11	19	27	35	40	56	$>$	$\sigma=3,\ L=1,\ a=2$
3	1	2	4	11	13	21	26	0	$<$	$\sigma=32,\ L=2, a=1$
32	2	1	8	14	21	23	31	6	$<$	$\sigma=321,\ L=3,\ a=4$
321	2	4	11	18	24	31	33	71	$>$	$\sigma=32,\ L=2,\ a=4$
32	2	4	7	15	18	30	32	60	$>$	$\sigma=3,\ L=1,\ a=4$
3	1	4	4	8	11	18	20	0	$<$	$\sigma=34,\ L=2,\ a=1$
34	2	1	8	11	18	20	28	0	$<$	$\sigma=341,\ L=3,\ a=2$
341	3	2	11	18	20	28	33	0	$<$	$S=3412,\ c(S)=0$
Optimal Solution is 3412 with Zero Penalty Cost.										Since $c(S)=0$. Stop

algorithm. The results of the proposed algorithm were compared to the complete enumeration approach, the only available method to solve the present problem. The proposed and complete enumeration algorithms were programmed on a UNIVAC 1108 computer in FORTRAN language to solve a considerably large number of problems. In order to carry out the experimental investigations, problems with linear penalty cost functions were generated from a uniform distribution. The process times, the due dates and the penalty cost rates of jobs were randomly generated from a rectangular distribution and ranged from 00-99, 000-999, and 00-09 respectively. The number of jobs in the problems varied from 4 to 7 and the number of machines ranged from 3 to 7.

In order to compare the efficiency of the proposed algorithm with that of complete enumeration, two factors—the average computation time per problem and the average number of schedules generated—were noted. Tables 3 and 4 contain the data for the proposed and complete enumeration algorithms. From the results of Tables 3 and 4, it is seen that the proposed search algorithm is comparatively more efficient than the complete enumeration approach.

7. Conclusions

The search algorithm described above provides a method for solving the flowshop scheduling problem when jobs are assigned due dates and penalty costs. The algorithm is flexible enough to accommodate linear as well as non-linear cost functions in as far as they are non-decreasing functions of time of lateness. From a practical view-point, the proposed algorithm gives additional information on partial solutions as the search progresses. The proposed search algorithm recognizes near optimal solutions at an early stage of search and the computation may be curtailed if only near optimal solutions, which are good enough in many practical situations, are desised. In fact, if it is not worthwhile to go in for a solution better than the current one unless the former has a penalty cost less than the latter by, say '*h*' units, the value of the

TABLE 3

Computation Times (in secs.) by Proposed Algorithm and Complete Enumeration*

Problem Size		#Problems	Computation Time Per Problem by Proposed Algorithm		Average Computation Time Per Problem by Complete Enumeration
M	n		Average	Range	
3	4	50	0. 0021	0. 0004—0. 0054	0. 0069
	5	50	0. 0042	0. 0008—0. 0160	0. 0351
	6	50	0. 0259	0. 0008—0. 0968	0. 2189
	7	50	0. 0912	0. 0008—0. 3354	1. 4577
4	4	50	0. 0027	0. 0006—0. 0070	0. 0084
	5	50	0. 0070	0. 0006—0. 0025	0. 0385
	6	50	0. 0340	0. 0008—0. 1522	0. 2323
	7	50	0. 1338	0. 0010—0. 5748	1. 5667
5	4	50	0. 0027	0. 0006—0. 0060	0. 0075
	5	50	0. 0116	0. 0008—0. 0308	0. 0383
	6	50	0. 0428	0. 0008—0. 1148	0. 2370
	7	50	0. 1914	0. 0010—0. 6426	1. 6070
6	4	50	0. 0034	0. 0006—0. 0072	0. 0079
	5	50	0. 0105	0. 0008—0. 0268	0. 0400
	6	50	0. 0292	0. 0008—0. 1654	0. 2558
	7	50	0. 2440	0. 0012—0. 8486	1. 7210
7	4	50	0. 0043	0. 0008—0. 0084	0. 0084
	5	50	0. 0125	0. 0008—0. 0314	0. 0427

* Computational times reported here do not include the time required for input (reading the data) and output (printing the results).

trial solution, δ, in the algorithm can be replaced by δ-h and the search continued. This increases the computional efficiency of search considerably. Any optimal solution, of value less than (δ-h), is necessarily found out. However, if the value of the actual optimal solution is between δ and δ-h, the optimal schedule will not be identified indicating the chances of an error, anywhere between 0 and $100\, h/\delta$ percent.

TABLE 4

Number of Schedules Per Problem by Proposed Algorithm and Complete Enumeration

Problem Size		Number of Schedules Per Problem by		
		Proposed Algorithm		Enumeration
M	n	Average	Range	
3	4	2. 550	01—07	24
	5	3. 125	01—10	120
	6	5. 025	01—16	720
	7	8. 050	01—30	5040
4	4	2. 450	01—07	24
	5	3. 950	01—10	120
	6	7. 525	01—19	720
	7	8. 200	01—26	5040
5	4	2. 600	01—08	24
	5	5. 200	01—18	120
	6	6. 625	01—18	720
	7	10. 900	01—33	5040
6	4	2. 925	01—08	24
	5	4. 650	01—12	120
	6	7. 175	01—17	720
	7	11. 050	01—31	5040
7	4	2. 900	01—07	24
	5	4. 825	01—08	120

References

[1] Gupta, J.N.D., (1969) "A General Algorithm for the $n\times M$ Flowshop Scheduling Problem" *The International Journal of Production Research*, 7, 241-247.

[2] Gupta, J.N.D., (1969) "Economic Aspects of Scheduling Theory" Ph. D. Dissertation, Texas Tech University, Lubbock, Texas.

[3] Johnson, S.M., (1954) "Two and Three Stage Production Schedules with Set-up Times Included" *Naval Research Logistics Quarterly*, 1, 61-68.

[4] Smith, M.L., (1968) "A Critical Analysis of Flowshop Sequencing" Ph. D. Dissertation, Texas Technological College, Lubbock, Texas.