G. CIOBANU
Department of Economic Cybernetics
Academy of Economic Studies

A General Algorithm for Solving Flow Shop Scheduling

The flow shop scheduling problem, as presented in this paper, consists in establishing the order of processing of n jobs on m machines, by taking into account the following hypotheses on the jobs (noted with A), on the machines (noted with B) and on the process time (noted with C).

A1. Each job covers the machines in the same technologically preestablished order,
A2. A job cannot be processed, at the same time, on several machines,
A3. Each job can be processed only when it is available,
A4. All jobs are considered to have the same importance,
B1. Each machine processes the jobs in the same order,
B2. A machine does not process, at the same time, more than one job,
B3. Each machine starts processing a job as soon as it is free,
B4. An item started on a machine cannot be interrupted,
C1. The process time for each job on each machine is deterministic.
C2. The process time does not depend on the sequence of processing of jobs.
C3. Set-up times and transfer times from one machine to the next are included in the process time.

In these conditions, let us consider n jobs $J_1, J_2 \ldots, J_n$ which are processed on m machines $M_1, M_2, \ldots, M_n$ and d_{ij} the time for processing of the jobs J_i on the machine M_j. Each job has a single copy. Consider $M_1, M_2, \ldots, M_m$ the technologically preestablished order of covering by the machines of the jobs.

If the order of processing of the jobs on the machines is, for example, J_{i_1}, J_{i_2}, ..., J_{i_n} then $(J_{i1}, J_{i2}, \ldots, J_{in})$ will be called a processing sequence of jobs and $(i_1, i_2, \ldots, i_n)$ the corresponding permutation. Any sequence is an feasible solution of the flow shop scheduling problem.

For a given sequence $S_i = (J_{i_1}, J_{i_2}, \ldots, J_{i_n})$ let us establish the starting and ending date of the items of processing the jobs $J_{i_1}, J_{i_2}, \ldots, J_{i_n}$ on

the machines M_1, M_2, ..., M_m. We note with $t_{ik,j}$ the starting date of the processing of job J_{ik} on the machine M_j.

Two situations occur:

a) The processing of job J_{ik} cannot start on the machine M_j because its processing on the machine M_{j-1} has not been finished. This corresponds to conditions A3 and A2 (see fig. 1a) and is represented through the relation

$$t_{ik,j} = t_{ik,j-1} + d_{ik,j-1} \tag{1}$$

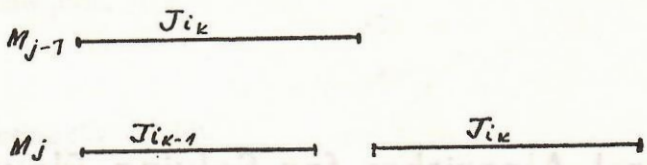

fig. 1a

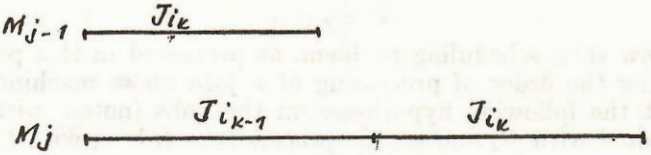

fig 1b

b) The processing of job J_{ik} cannot start on machine M_j because the processing of job J_{ik-1} has not been finished. This corresponds to conditions B3 and B2 (fig. 1.b) and correspondingly we have the relation:

$$t_{ik,j} = t_{ik-1,j} + d_{ik-1,j}, \tag{2}$$

From (1) and (2) the starting date $t_{ik,j}$ is given by the relation

$$t_{ik,j} = max\,(t_{ik,j-1} + d_{ik,j-1},\ t_{ik-1,j} + d_{ik-1,j}). \tag{3}$$

It results from (3) that *the total elapsed time for processing of jobs corresponding to sequence S_i is*:

$$T_{S_i} = t_{in,m} + d_{in,m}. \tag{4}$$

The flow shop scheduling problem consists in establishing sequences S_0 which minimize the total elapsed time for the completion of all jobs:

$$T_{S_0} = \min_{S_i \in \mathcal{S}} T_{S_i}$$

where:

$$\mathcal{S} = \{S_i = (J_{i_1}, J_{i_2}, ..., J_{i_n})\}$$

is the set of the $n!$ *possible sequences.*

The difference:

$$\tau_{i_k, j} = t_{i_k, j} - (t_{i_{k-1}, j} + d_{i_{k-1}, j}) \tag{5}$$

where: $t_{i_0, j} = 0\ j = 1, 2, ..., m$ is called the idle time for *machine* M_j before the processing of job j_{i_k}. The total duration T_{S_i} can be written:

$$T_{S_i} = \sum_{k=1}^{n} d_{i_k, m} + \sum_{k=1}^{n} \tau_{i_k, m} \tag{6}$$

In this relation (6), the total elapsed time T_{S_i} represents the sum of of the processing time of jobs with the idle time on the last machine M_m. This results from the conditions of the problem, the jobs covering the machines in the same order.

Theorem 1. By the minimization of the total elapsed time for completion of all jobs or by the minimization of the idele time on the last machine M_m, on the set of sequences $\mathcal{S}$ the problem has the same optimal slution.

This result derives from (6) because $\sum_{k=1}^{n} d_{i_k, m}$ is a constant value. The algorithms for minimizing the total elapsed time of processing of jobs, were given by Gupta [1], Brown and Lomnicki [3] and for minimizing the idele time on machines building lower bounds for each machine by Muller Merbach [2].

This paper will present an algorithm which minimizes the total elapsed time of processing of jobs, based on the principle of incomplete enumeration, in which the lower bounds on each machine are calculated.

Consider $(j_1, j_2, ..., j_i)$ a partial sequence of jobs and, $\sigma = (1, 2, ..., i)$ the corresponding permutation. In this case

$$T_{ij} = \sum_{k=1}^{i} d_{kj} + \sum_{k=1}^{i} \tau_{kj} \quad i = 1, 2, ..., n, \quad j = 1, 2, ..., m \tag{7}$$

is the finish date of job J_i on machine M_j.

This term T_{ij} is easily calculated with the recurrence relation (3), namely

$$T_{ij} = t_{ij} + d_{ij} \tag{8}$$

Corresponding to the partial permutation σ we define for machine M_j the lower bound:

$$g_j(\sigma) = T_{ij} + \sum_{k=i+1}^{n} d_{kj}, \qquad j = 1, 2, ..., m \tag{9}$$

where: $\sum_{k=i+1}^{n} d_{kj}$ is the processing time on machine M_j of the unscheduled jobs, this being independent of the order in which these jobs will be

scheduled. On machine M_j they work, as can be noticed, with at least a time equal to $g_j(\sigma)$.

Thus, $g_j(\sigma)$ is the estimate of the total elapsed time T_S of processing of all jobs on machine M_j.

Taking into account conditions A1 and B1 the last job in a certain sequence S can be processed on machines M_{j+1}, ..., M_m for at least a time equal to:

$$D_j = \min_{k=\overline{1,n}} \sum_{l=j+1}^{m} d_{kl} \tag{10}$$

Thus, the jobs in a sequence S can be processed on machine M_j for at most:

$$TS_j = T_S - D_j \tag{11}$$

where D_j is calculated with (10).

Theorem 2. If for a certain machine M_j we have $g_j(\sigma) > TS_j$ the sequences obtained by completing the partial permutation σ are eliminated.

If another index of an unscheduled job is added to permutation σ, the value $g_j(\sigma)$ increases with the idle time on machine M_j introduced by this added job. In this way, the sequences obtained are not achieved in a time shorter than sequence S, considered as a comparison sequence.

Thus, it is noticed that $g_j(\sigma)$ is an lower bound of the length of processing of jobs on machine M_j.

If P is a complete sequence and $T_P < T_S$, the Sequence P is preferred to Sequence S and in this case P replaces S.

The sizes TS_1, TS_2, ..., TS_m are considered upper bounds of the processing time on machine M_1, M_2 ..., M_m.

THE SOLVING ALGORITHM

The optimum sequences S_0 can be determined by applying the principle of incomplete enumeration. This consists in establishing a lexicographical list of the partial permutations.

Permutation $(j_1, j_2, ..., j_v)$ lexicographically precedes permutation $(i_1, i_2, ..., i_v)$ if $j_1 < i_1$ or $j_1 = i_1$ and $j_2 < i_2$ or $j_1 = i_1, j_2 = i_2, j_3 < i_3$ and soon.

An initial sequence S is built for which the total elapsed time of processing of jobs is calculated. Generally, this sequence is found out by using a heuristic algorithm which is leading to an ever better solution. For example:

- The sum of the processing time is calculated for each job J_i.

$$H_i = \sum_{k=1}^{m} d_{ik}, \qquad i = 1, 2, ..., n.$$

- The sizes H_i are sorted in an increasing order

$$H_{i_1} \leqslant H_{i_2} \leqslant ... \leqslant H_{i_n}$$

- The initial sequence will be:

$$S = (J_{i_1}, J_{i_2}, ..., J_{i_n})$$

The upper bounds $TS_1, TS_2, ..., TS_m$ are calculated for this sequence, with relation (11).

In continuation, we shall work with the permutations corresponding to the sequences.

In the course of the enumeration, certain partial permutations are analysed and the seeking of the best permutations is facilitated by the use of the lower bound $g_j(\sigma)$ which sometimes reduces the effort of calculation.

If in the process of enumeration, during the building of permutations the upper bound is surpassed according to theorem 2, this branch is abandoned and the construction of another permutation starts.

Though we shall work with m upper bounds $TS_1, TS_2, ..., TS_m$ the enumeration process is achieved in the same way as in the Gupta algorithm [1]. Each time when a new job J_l is scheduled, thus a new element l is added to the partial permutation σ, the finish date T_{lj} on machine M_j is calculated, to which is added the sum of the processing times on machine M_j of the other unscheduled jobs, according to relation (9), the lower bound $g_j(\sigma l)$ being obtained. If $g_j(\sigma l) < TS_j$, the same calculations are made for job j_l on machine M_{j+1}. These calculations are made starting with machine M_1, then with machine M_2, and so on, up to machine M_m. If there is an index j for which $g_j(\sigma l) > TS_j$, no calculations are made on the next machines $M_{j+1}, M_{j+2}, ...$ $..., M_m$ and at the same time the construction of partial permutations which have (σl) as a leader is abandoned. Further on, the lexicographical construction of a new permutation is started having as a leader the initial permutation σ' obtained from σ by eliminating the last element.

If, in the enumeration process there is a complete permutation for which g_j are less to the upper bounds TS_j for all $j = 1, 2, ..., m$. this permutation is chosen as an initial solution and the upper bounds $TS_1, TS_2, ..., TS_m$ are recalculated.

NUMERICAL EXAMPLE

We shall illustrate the algorithm by solving the problem given by Gupta [1]. in which we have five machines and four jobs; the processing time d_{ij} is given in the following table:

i \ j	M_1	M_2	M_3	M_4	M_5
J_1	4	3	7	2	8
J_2	3	7	2	8	5
J_3	1	2	4	3	7
J_4	3	4	3	7	2

In the following table the permutations corresponding to the sequences are generated. The optimum sequence is (J_3, J_4, J_1, J_2) and will be processed in 33 time units.

The initial sequence was not determined with the presented heuristic algorithm but with the sequence (J_1, J_2, J_3, J_4) which is disadvantageous and which is achieved in 38 time units. The upper bouns are $TS_1 = 22$, $TS2 = 26$, $TS3 = 29$, $TS4 = 36$, $TS5 = 38$,

No calculations are made in the places with a dash according to the theorem we pass on to building a new permutation with another leader.

Permutations (σ)				Finish date					Lower bounds $g_j(\sigma)$					Upper bounds TS_j				
I	*II*	*III*	*IV*	*M1*	*M2*	*M3*	*M4*	*M5*	*M1*	*M2*	*M3*	*M4*	*M5*	*M1*	*M2*	*M3*	*M4*	*M5*
1	2	3	4	11	20	23	34	38										
1	2	4		10	18	21	31	33	11	20	25	34	40	22	26	29	36	38
1	3			5	9	18	21	—	11	20	24	36						
1	4			7	11	17	23	26	11	20	23	34	38					
2				3	10	12	20	25	11	19	26	32	42					
3				1	3	7	10	17	11	17	19	27	32					
3	1			5	8	15	17	25	11	19	20	32	32					
3	1	2		8	15	17	25	30	11	19	20	32	32					
3	1	2	4	11	19	22	32	34	—	—	—	—	—	18	22	25	32	34
3	1	4		8	16	—	—	—	11	23	—	—	—					
3	2			4	11	13	21	16	11	18	23	30	36					
3	4			4	8	11	18	20	11	18	20	28	33					
3	4	1		8	11	18	20	28	11	18	20	28	33					
3	4	1	2	11	18	20	28	33	—	—	—	—	—	17	21	24	31	33
3	4	2		7	15	19	—	—	11	18	26	—	—					
4				3	7	10	17	19	11	19	23	30	39					

This algorithm does not generate more partial permutations than Gupta's algorithm, sometimes fewer.

REFERENCES

1. GUPTA, J. N. D. *A General Algorithm for the* N×M *Flowshop Scheduling Problem,* International J. Production Res. 3, 241—247. (1969)
2. MULLER-MERBACH, *Optimale Reihenfolgen,* Springer Verlag Berlin, Heidelberg, New York, 225 pages. (1970)
3. BROWN, A.P.G. LOMNICKI, Z. A.: *Some Applications of the Branch-and-Bound Algorithm to the Machine Scheduling Problem.* Opnal. Res. Quart 17, 2, 173—186, 1966.
4. BAKSHI, M. S., ARORA, S. R.: *The Sequencing Problem.* Management Sci, 16,4, B 247 B263, 1969.
5. ASHOUR, S.: *An Experimental Investigation and Comparative Evolution of Flow-Shop Scheduling Technique,* Operations Res. 18, 3, 541—548, 1970.