COMPARISON OF FLOWSHOP SCHEDULING ALGORITHMS

by 45

LAXMAN SINGH BENIWAL

B.E.(Mech.) Birla Institute of Technology & Science
Pilani, Rajasthan, India, 1965

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1969

Approved by:

TABLE OF CONTENTS

ACKNOWLEDGMENT

I wish to extend my most profound gratitude to Dr. Said Ashour for his acting as my supervisor and giving his valuable guidance and inspiration throughout the period of this report. For this privilege and benevolence, not only, I remain indebted, but, I sincerely and fervently wish to place on record that but for his bestowing these favours, this little presentation could not have been possible.

My thanks are also due to Dr. Frank A. Tillman, Head Department of Industrial Engineering and Professor J. J. Smaltz for their kind patronage.

I also wish to extend my thanks, to the Director, Computing Center for providing me all the facilities in completing my work.

I greatly acknowledge and thank Mrs. Sharon Morrisey for her valuable assistance in typing.

My thanks are also due to Mr. Prem Chand and Mr. Promod Mehrotra for their assistance in the proof reading.

I shall be failing in my duty if I don't thank Dr. L. E. Grosh, who very kindly agreed to be a member of my committee at very short notice.

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Scheduling is one of the critical problems in management. The scheduling problem may be classified into three categories: Production scheduling, Project scheduling and Job scheduling. This report is concerned with the job scheduling problem.

The job scheduling problem consists of processing J jobs on M machines such that a certain criterion is optimized. Some of the criteria are: (1) minimization of the total time required to process all jobs; (2) minimization of the total idle time on all the machines; (3) minimization of the time required for each machine, starting from the first job to the last job; and (4) maximization of the profit by meeting of due dates. However, the criterion considered in this report is that of minimizing the schedule time.

The job scheduling problem is of interest because of its diversity, complexity and magnitude. For example, consider a problem of six jobs to be processed on each of three different machines. The possible number of sequences is $(J!)^M$ or $(6!)^3 = 373,248,000$. A complete enumeration of these sequences would require years on a high speed computer. Many of these sequences are technologically non-feasible. An exhaustive enumeration must consider all sequences to eliminate the non-feasible and then select the optimal sequence.

1.1. Problem Formulation:*

Consider a job scheduling problem which consists of a certain number of jobs to be processed on various machines in a specified machine ordering. The optimal sequence of the jobs is to be determined so as to minimize the schedule time - the total time of processing all jobs.

The job scheduling problems are generally classified into: (1) Flowshop - where all jobs have the same machine ordering, and (2) jobshop - where each job has a different machine ordering.

The notations which have been used to formulate the problem mathematically are given below:

J	total number of jobs
M	total number of machines
j	job designation, j = 1, 2, . . ., J
m	machine designation, m = 1, 2, . . ., M
jm	operation designation
$j_x m$	sequence of jobs through machine m, x = 1, 2, . . ., J
jm_y	order of machines for job j. y - 1, 2, . . ., M
$j_x m_y$	a specific operation
t_{jm}	processing time of job j on machine m
T*	processing time matrix of the original problem.
M_j*	machine ordering vector for job j
M*	machine ordering matrix of the original problem
S_m*	job sequencing vector through machine m
S*	job sequencing matrix of the original problem
T	schedule time

---

*Adapted from Ashour, S., "A Decomposition Approach for the Machine Scheduling Problem," The International Journal of Production Research, Vol. 6, No. 2, 1967.

The numbering of jobs and machines is preconcieved and not necessarily correspond to the sequence in which jobs are processed on each machine or to the order in which the machines process each job. Thus, the sequence of the jobs will be designated as $j_1, j_2, \ldots, j_x, \ldots, j_J$ and the order of the machines as $m_1, m_2, \ldots, m_y, \ldots, M_M$. The term j_x indicates that job j in position x; and the term m_y means that machine m in position y.

The machine ordering for each job is designated as follows:

$$M_j^* = \{ jm_y \mid y \varepsilon M \quad \text{and} \quad (jm_y) < (jm_{y+1}) \} , \; j = 1, 2, \ldots, J$$

where,

$(jm_y) < (jm_{y+1})$ indicates that the operation of job j on machine m_y directly preceeds the operation of the same job on machine m_{y+1}. The above relation may be written as follows:

$$M_j^* = \{ jm_1 \; jm_2 \ldots jm_x \ldots jm_M \} , \quad j = 1, 2, \ldots, J.$$

The machine ordering vector for each job can be combined in a (J x M) matrix called the machine ordering matrix denoted by M*.

$$M^* = \begin{bmatrix} 1m_1 & 1m_2 & \ldots & 1m_y & \ldots & 1m_M \\ 2m_1 & 2m_2 & \ldots & 2m_y & \ldots & 2m_M \\ \vdots & \vdots & & \vdots & & \vdots \\ Jm_1 & Jm_2 & \ldots & Jm_y & \ldots & Jm_M \end{bmatrix}$$

The processing time of each job on each machine is designated by T*.

$$T^* = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1m} & \cdots & t_{1M} \\ t_{21} & 2_{22} & \cdots & t_{2m} & \cdots & t_{2M} \\ \vdots & \vdots & & \vdots & & \vdots \\ t_{j1} & t_{j2} & \cdots & t_{jm} & \cdots & t_{jM} \\ \vdots & \vdots & & \vdots & & \vdots \\ t_{J1} & t_{J2} & \cdots & t_{Jm} & \cdots & t_{JM} \end{bmatrix}$$

It should be pointed out that the processing time matrix, T*, does not imply any ordering of the operations. If a job is not to be processed on a particular machine, a zero processing time is placed in the processing time matrix.

The sequencing of jobs on each machine are designated by S_m^*.

$$S_m^* = \{ j_x m \mid x \varepsilon J \quad \text{and} \quad (j_x m) < (j_{x+1} m) \} , \quad m = 1, 2, \ldots, M$$

where

$(j_x m) < (j_{x+1})m$ indicates that job j_x directly preceeds job j_{x+1} on machine m.

The above relation may be rewritten as follows:

$$S_m^* = \{ j_1^m \; j_2^m \; . \; . \; . \; j_x^m \; . \; . \; . \; j_J^m \} , \quad m = 1, 2, \; . \; . \; ., M.$$

The above job sequencing vectors can be combined in a (M x J) matrix, known as the job sequencing matrix and denoted by S*.

$$S^* = \begin{bmatrix} j_1^1 & j_2^1 & . \; . \; . & j_x^1 & . \; . \; . & j_J^1 \\ j_1^2 & j_2^2 & . \; . \; . & j_x^2 & . \; . \; . & j_J^2 \\ \vdots & \vdots & & \vdots & & \vdots \\ j_1^m & j_2^m & . \; . \; . & j_x^m & . \; . \; . & j_J^m \\ \vdots & \vdots & & \vdots & & \vdots \\ j_1^M & j_2^M & . \; . \; . & j_x^M & . \; . \; . & j_J^M \end{bmatrix}$$

In summary, the job scheduling problem may be stated such that: given the machine ordering matrix M* and the processing time matrix T*, find the optimal job sequencing matrix S* which minimizes the schedule time T.

In the machine ordering matrix M*, the element jm_1, for example, indicates that job j must be processed on machine m first. The element jm_2 means that job j must be performed on machine m second, which is not the same as that in element jm_1. Also in the job sequencing matrix S, the element j_1m, for example, indicates that machine m processes job j first. The element j_2m means that machine m performs job j second, which is not the same job as that in the element j_1m. As mentioned earlier, the subscript here indicates the position of job j in the job sequencing

or position of machine m in the machine ordering.

An example may clarify the above formulation. Consider a jobshop problem of four jobs and two machines. The machine ordering matrix is given below:

$$M^* = \begin{bmatrix} 1m_1 & 1m_2 \\ 2m_1 & 2m_2 \\ 3m_1 & 3m_2 \\ 4m_1 & 4m_2 \end{bmatrix} = \begin{bmatrix} 11 & 12 \\ 22 & 21 \\ 31 & 32 \\ 42 & 41 \end{bmatrix}$$

This indicates that jobs 1 and 3 are to be processed on machine 1 first and machine 2 last; however, jobs 2 and 4 are to be performed on machine 2 first and machine 1 last.

One of the feasible sequences which must be consistent with the above machine ordering matrix is shown below

$$S_o = \begin{bmatrix} j_1{}^1 & j_2{}^1 & j_3{}^1 & j_4{}^1 \\ j_1{}^2 & j_2{}^2 & j_3{}^2 & j_4{}^2 \end{bmatrix} = \begin{bmatrix} 11 & 31 & 21 & 41 \\ 12 & 22 & 32 & 42 \end{bmatrix}$$

This particular job sequencing matrix indicates that machine 1 processes the jobs in the sequence { 1 3 2 4 } and machine 2 performs the jobs in the sequence { 1 2 3 4 } .

A simple formulation is given below to the flowshop problem where all jobs have the same machine ordering.

Given the machine ordering matrix

$$M^* = [jm_1\ jm_2\ \ldots\ jm_x\ \ldots\ jm_M], \quad j = 1, 2, \ldots, J,$$

and the processing time matrix

$$T^* = [t_{j1}\ t_{j2}\ \ldots\ t_{jm}\ \ldots\ t_{jM}], \quad j = 1, 2, \ldots, J,$$

find the optimal sequence, represented by the job sequencing matrix

$$S_m^* = [j_1m\ j_2m\ \ldots\ j_xm\ \ldots\ j_Jm], \quad m = 1, 2, \ldots, M,$$

which gives the minimum schedule time T.

In the machine ordering matrix, M^*, the element jm_1, for example, indicates that job j must be processed on machine m first. The element jm_2 means that job j must be performed on machine m second, which is not the same machine as that in the element jm_1. Also in the job sequencing matrix, S_m^* the element j_1m, for example, indicates that machine m processes job j first. The element j_2m means that machine m performs job j second, which is not the same job as that in the element j_1m.

The job scheduling problem as formulated above is restricted to the following assumptions:

1. Assumptions regarding jobs:
    - 1.1 A job may not be processed by more than one machine at a time.
    - 1.2 Each job must follow a specified machine ordering.
    - 1.3 A job is processed as soon as possible, subject to the machine ordering.
    - 1.4 All jobs are equally important; i.e., no priorities, due dates, or rush orders.

2. Assumptions regarding machines:

   2.1 No machine may process more than one job at a time.

   2.2 Once started, each operation must be completed.

   2.3 There is only one machine from each type.

   2.4 No job is processed more than once by any machine.

3. Assumptions regarding processing times:

   3.1 The processing time of each job on each machine does not depend on the sequence in which the jobs are processed.

   3.2 The processing time of each job on each machine is determinate and integer.

   3.3 Transportation times between machines and set-up times, if any, are included in the processing times.

1.2. Literature Review:

Considerable research has been done in the field of job scheduling problem. However, an optimal solution for practical problems have not yet been obtained except for small size problems. The approaches available to solve the job scheduling problem are: (1) Combinatorial Analysis, (2) Integar-Linear Programming, (3) Graphical, (4) Graphical-Dynamic Programming, (5) Schedule Algebras, (6) Simulation.

Some of the researchers have concentrated on generating the feasible sequences only, for the job scheduling problem. A Boolean Algebra approach has been developed by Akers and Friedman (4) which eliminate initially a great number of sequences. Certain decision rules have been developed for minimizing the schedule time. An illustration has been made

by solving two-job and M-machine problem.

Giffler and Thompson (55) have developed an algorithm in which they suggested to search for optimal sequences over only feasible schedules referred to as active feasible schedules. They have defined an active feasible schedule as a feasible schedule having the property that no operation can be made to start sooner by permissible left shifting. In their paper, it has been proved that the subset of active feasible schedule contains a subset of optimal schedules; and every optimal schedule is equivalent to an active optimal solution. Equivalence means that one schedule can be obtained from the other by a sequence of permissible shifts. The main reasons for concentrating on the active schedules are: (1) they contain a subset of optimal sequences; (2) they are superior to the inactive schedules; and (3) their number is usually very small compared to the number of all schedules. For smaller problems, it is possible to generate the complete set of active schedules and then pick-up the optimal. However, in large problems, it is impractical to generate all active schedules. Therefore, a random sample is generated from the set of all active schedules and the optimal schedule(s) may be obtained with some probability close to one as desired.

Using the linear graphs, Heller and Logemann (68) have developed an algorithm to generate feasible schedules and compute the corresponding schedule time. An operation of processing job j on machine m for the return i is referred to as a node (mji). These appropriate nodes are linked as per the machine ordering. One of the available nodes is chosen and scheduled. This process is continued until all operations are

scheduled. The result is one feasible schedule and the corresponding schedule time.

Furthermore, there exists several scheduling algorithms for obtaining a solution to the scheduling problem.

For J-jobs and two-machine flowshop problem, Johnson (80) has developed a simple algorithm to determine an optimal schedule for minimizing the schedule time. He also extended his algorithm to cover a special case of the three-machine problems. For jobshop problems, however, Jackson (71) extended Johnson's results.

Dudek and Teuton (38) have extended Johnson's algorithm to solve the flowshop problems of J-job and M-machine. The algorithm involves the minimization of the cumulative idle time on the last machine. Although they have claimed that optimality is guaranteed, Karush (82) has formulated a counter example of three-job and 3-machine. Smith and Dudek (156) have revised the above algorithm to obtain the optimal solution.

Another approach which gives optimal or near optimal solution after the generation of only a small subset of the possible sequences. This is referred to as branch-and-bound technique which has been developed by Little et. al. (92) to solve the travelling salesman problem. Ignall and Scharge (70) have applied the above technique to the two-and three-machine flowshop problem. Their computational experience involves up to nine jobs. Brown and Lomnicki (27) have generalized the branch-and-bound algorithm developed by Lomnicki (93) for three machines to arbitrary number of machines. McMahon and Burton (99) have applied branch-and-bound technique for the three-machine problem, giving a new method of

obtaining the bound and utilized the fact that scheduling problems are symmetrical and with respect to time-reversal. Their computational experience on CDC 3600 computer involves up to 45 jobs and three-machines. They concluded that the use of the composite bound (machine based bound and job based bound) decision rule is more efficient.

Brooks and White (26) have modified Giffler and Thompson's (54) algorithm using the lower bound as a decision rule for developing an optimal or near optimal solution. Since practical problems are much larger than could be solved economically by this procedure, they used lower bound as a decision rule for developing a near optimal solution.

Integar-Linear programming approach has been utilized for job scheduling problem. Up to this time, three different formulations developed by Bowman (25), Wagner (165) and Manne (95) are available. However, the computations are not feasible. Dantzig (35, 36) formulated the job scheduling problem as an ordinary linear programming problem by neglecting the integar constraints. The drawback of this method is that it may give fractional optimal solution. To overcome this problem, Giglio and Wagner (57) have solved 100 flowshop problems of six-job and three-machine using an ordinary linear programming and rounding off the final solution. However, computations were not encouraging.

In his investigation, Heller (53) has shown that the limit distribution of the schedule times is asymptotically normal as the number of jobs increases.

Ashour (6) has developed a decomposition approach to the job scheduling problem. In this approach, the problem is decomposed to a

smaller, more manageable problems which requires less computational effort. In his investigation, Ashour has found that the schedule time distribution obtained by decomposition shifts toward the minimum value. Furthermore, the shift increases as the number of jobs in each subgroup increases. His computational experiment consists of six to 40 jobs and three to ten machines.

Hardgrave and Nemhauser (62) and Akers (3) have developed a graphical approach for two-job and M-machine problem. Graphical-Dynamic programming approach has been presented by Held and Karp (64) for J-job and one-machine problem. Szwarc (160) has given a solution for a problem of two-job and M-machine by a combination of dynamic programming and graphical methods. Szwarc has also developed a technique for the job-shop problem, but it does not guarantee optimality.

In practice, the job scheduling problem is dynamic in nature. The break-down of machines, efficiency of the labor, quality of the products make the scheduling problems quite complicated. For such complex problems, a computer simulation is used. Eilon and Hodgson (40) have developed a simulation model for jobshop scheduling problems consisting of two identical machines operating in parallel. Conway et. al. (33) have simulated jobshop problem of five-machine and 100 arriving jobs to test the priority rules. Gere (47) has studied the performance of a number of combinations of priority rules and heuristics for several criteria such as due dates, minimization of the sum of latness.

1.3. Proposed Research:

Several techniques have been suggested for the solution of job scheduling problems. In general, these techniques are practical for small size problems. A little progress has been made toward the development of efficient algorithms.

Among the promising techniques, which are used to solve the flowshop problem are: (1) Direct technique reported by Smith and Dudek (156); (2) Branch-and-bound technique developed by Brown and Lomnicki (27); and (3) lower bound modified by Brooks and White (26). In reviewing the literature, it appears that no comparison among these procedures has been made. The purpose of this report is to investigate the solution obtained by different procedures. Thus the techniques are compared considering basically the following: (1) the relative efficiency of the different solutions to the optimal value; (2) the statistical characteristics of the distinct schedule times; and (3) the computational efficiency. To obtain adequate comparison among these techniques and to minimize the variations, considerable experiments were conducted. In this report, the solution by Branch-and-bound and Direct algorithms are compared to the complete enumeration.

In the next chapter, the scheduling techniques are discussed and illustrated by a sample problem. Chapter III is devoted to the experimental investigation designed to compare the performance of the scheduling techniques. The results of these computational experiments and the conclusions are reported in Chapter III and IV.

CHAPTER II

COMPUTATIONAL SCHEDULING ALGORITHMS

This chapter includes clear presentation of various algorithms for the solution of the flow shop scheduling problem, such as (1) Linear graph algorithm by Ashour (6) which is used in this report to obtain the complete enumeration; (2) Direct algorithm by Smith and Dudek (156); and (3) Branch-and-bound algorithm by Brown and Lomnicki (27). A sample problem of flow shop having six jobs and three machines is solved by the above various techniques to illustrate these algorithm step-by-step.

2.1. Complete Enumeration Technique *

The linear graph algorithm which has been developed by Ashour (6) to generage feasible sequences and compute the schedule time for flow shop problems is used to obtain the complete enumeration solutions. This algorithm is based on linear graph theory presented by Heller (67). An operation of processing job j on machine m is represented by node (jm). Important features of the algorithm are: (1) it can permute a complete set of sequences or generate, at random, a subset of all sequences. Thus the algorithm is flexible in the sense that any number of sequences can be constructed and evaluated; (2) it constructs and evaluates a sequence of J jobs in exactly J iterations regardless of the number of machines involved. As a result, the algorithm schedules m nodes in one iteration and a sequence is eval-

*Adapted from Ashour, S., "A Decomposition Approach for the Machine Scheduling Problem," Ph.D. Thesis, University of Iowa, 1967.

uated immediately; and (3) the machine ordering can arbitrarily be assigned but the same for all jobs. For example, in the case of three machines, the machine ordering could be that all jobs are processed on machine 3 first, machine 1 second and machine 2 last. These features enables the algorithm to compute quite easily and quickly the schedule time as well as the construction of a sequence.

Consider the processing time and machine ordering matrices appeared on page 16 of this report, the initial schedule table is constructed. This table includes the following columns

Q_1: node designation, (jm),

Q_2: processing time, t_{jm},

Q_3: index of the sequence of job j on machine m

Q_4: starting time of node (jm)

Q_5: finishing time of node (jm)

The following algorithm evaluates the schedule time for the sequence $\{j_1\ j_2 \cdot \cdot \cdot j_J\}$

Step 1: Construct the initial scheduling table, by setting:

1.1. the nodes (jm) in column 1 of the machine ordering matrix under Q_1, first, those in column 2 second, . . , and those in column M last,

1.2. the corresponding processing times t_{jm} under Q_2, and by leaving

1.3. Q_3 (jm), Q_4 (jm) and Q_5 (jm) blank.

Step 2: Let U = 1, and X = 1

Step 3: Set j = jx

Step 4: Index the job sequence in all machines by setting:

$$Q_3 (jm_y) = U, \quad y = 1, 2, \ldots, M$$

Step 5: Check U

5.1. if U = 1, go to step 6.

5.2. if U > 1, let jo = j_{x-1} and replace Q_4 (jm_4)

by

$$\max [Q_4 (jmy), Q_5 (j_o m_y)], \quad = 1, 2, \ldots, M$$

Step 6: Compute the finishing time:

6.1. Compute $Q_5 (jm_1) = Q_4 (jm_1) + Q_2 (jm_1)$

6.2. replace $Q_5 (jm_y)$ by $\max [Q_4 (jm_y), Q_5 (jm_{y-1})]$

and compute

$$Q_5 (jm_y) = Q_r (jm_y) + Q_2 (jm_y)$$

$$y = 2, 3, \ldots, M.$$

Step 7: Increase both x and U by one. Then repeat step 3 through 6 until all jobs are scheduled.

Step 8: Find the schedule time such that $T^* = \max [Q_5 (jm)]$

The following is a sample problem, which has been solved. The processing time and machine ordering matrices are

$$T^* = \begin{bmatrix} 4 & 5 & 5 \\ 2 & 17 & 7 \\ 2 & 10 & 4 \\ 10 & 8 & 2 \\ 7 & 15 & 6 \\ 9 & 4 & 11 \end{bmatrix} \qquad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix}$$

Let the sequence {3 6 2 5 1 4} is there and by applying the above algorithm, the schedule time can be found.

According to step 1.1. at the algorithm, initial scheduling table is constructed by writing nodes (jm) in the first column as shown in the table I and according to step 1.2. corresponding processing times are filled in the second column. It is a point to be noted in this table that nodes under Q_1 are arranged in three groups such that those in column 1 of the machine ordering matrix are set first, those in column 2 second and those in column 3 last. In table I, nodes (j1) are set in the first group, nodes (j2) in the second, and nodes (j3) in the third, where j = 1, 2, . . ., 6.

In order to compute the schedule time of the sequence {3 6 2 5 1 4} one first schedules job 3 on all machines by setting Q_3 (31), Q_3 (32) and Q_3 (33) equal to one in the schedule table. Since Q_4 (31) = 0, the completion time of the node 31 in the unit of time is

$$Q_5 (31) = Q_4 (31) + Q_2 (31) = 0 + 2 = 2$$

Job 3 cannot be processed on machine 2 unless it is has been processed on machine 1. Therefore, Q_4 (32) is replaced by

$$\max [Q_4 (32), Q_5 (31)] = \max [0, 2] = 2.$$

The finishing time of this node is

$$Q_5 (32) = Q_4 (32) + Q_2 (32) = 2 + 10 = 12.$$

In order to process job 3 on machine 3, replace Q_4 (33) by

$$\max [Q_4 (33), Q_5 (32)] = \max [0, 12] = 12,$$

and compute the completion time as

$$Q_5 (33) = Q_4 (33) + Q_2 (33) = 12 + 4 = 16$$

Now job 3, has been scheduled on machines 1, 2 and 3. The updated values are in Table 2.1.

Table 2.1

Initial Scheduling Table

Q_1	Q_2	Q_3	Q_4	Q_5
11	4			
21	2			
31	2			
41	10			
51	7			
61	9			
12	5			
22	17			
32	10			
42	8			
52	15			
62	4			
13	5			
23	7			
33	4			
43	2			
53	6			
63	11			

Table 2.2

Scheduling Table

Q_1	Q_2	Q_3	Q_4	Q_5
11	4			
21	2			
31	2	1	0	2
41	10			
51	7			
61	9			
12	5			
22	17			
32	10	1	2	12
42	8			
52	15			
62	4			
13	5			
23	7			
33	4	1	12	16
43	2			
53	6			
63	11			

Next, job 6 is scheduled on all machines by setting Q_3 (61), Q_3 (62) and Q_3 (63) equal to two. This indicates that job 6 is the second job to be scheduled on all machines. Obviously, job 6 cannot be processed on machine 1 before job 3 is completed on that machine. Therefore, to avoid conflict or overlapping, Q_4(61) is replaced by

$$\max [Q_4 (61), Q_5 (31)] = \max [0, 2] = 2$$

For the same reason, also replace Q_4 (62) with

$$\max [Q_4 (62), Q_5 (32)] = \max [0, 12] = 12,$$

and Q_4 (63) with

$$\max [Q_4 (63), Q_4 (33)] = \max [0, 16] = 16.$$

The completion time of node 61 is

$$Q_5 (61) = Q_4 (61) + Q_2 (61) = 2 + 9 = 11.$$

Again, job 6 cannot be processed on machine 2 before job 3 is completed on that machine. Therefore, $Q_4(62)$ is replaced by

$$\max [Q_4 (62), Q_5 (61)] = \max [12, 11] = 11$$

Table 2.3

Scheduling Table

Q_1	Q_2	Q_3	Q_4	Q_5
11	4			
21	2			
31	2	1	0	2
41	10			
51	7			
61	9	2	2	11
12	5			
22	17			
32	10	1	2	12
42	8			
52	15			
62	4	2	12	16
13	5			
23	7			
33	4	1	12	16
43	2			
53	6			
63	11	2	16	27

Table 2.4

Scheduling Table

Q_1	Q_2	Q_3	Q_4	Q_5
11	4			
21	2	3	11	13
31	2	1	0	2
41	10			
51	7			
61	9	2	2	11
12	5			
22	17	3	16	33
32	10	1	2	12
42	8			
52	15			
62	4	2	12	16
13	5			
23	7	3	33	40
33	4	1	12	16
43	2			
53	6			
63	11	2	16	27

Table 2.5

Scheduling Table

Q_1	Q_2	Q_3	Q_4	Q_5
11	4	5	20	24
21	2	3	11	13
31	2	1	0	2
41	10	6	24	34
51	7	4	13	20
61	9	2	2	11
12	5	5	48	53
22	17	3	16	33
32	10	1	2	12
42	8	6	53	61
52	15	4	33	48
62	4	2	12	16
13	5	5	54	59
23	7	3	33	40
33	4	1	12	16
43	2	6	61	63
53	6	4	48	54
63	11	2	16	27

Then the finishing time of job 5 on machine 3 is

$$Q_5\ (62) = Q_4\ (62) + Q_2\ (62) = 12 + 4 = 16$$

Further, replace $Q_4\ (63)$ by

$$\max\ [Q_4\ (63),\ Q_5\ (62)] = \max\ [16,\ 16] = 16$$

and the completion time of node {63} is

$$Q_5\ (63) = Q_4\ (63) + Q_2\ (63) = 16 + 11 = 27$$

The updated schedule table shows that jobs 3 and 6 are scheduled on all machines and appear as Table 2.3.

Now job 2 is the third job to be scheduled according to the sequence of {3 6 2 5 1 4}. Table 2.4 shows the updated scheduling table by proceeding as before. Finally, when jobs 5, 1 and 4 are scheduled in this sequence, the final scheduling table of the six jobs is obtained as shown in Table 2.5.

For convenience, Table 2.6 shows the arranged scheduling table. This table is the final scheduling table after its row in each group have been arranged in ascending order according to Q_3 (jm). The schedule time for the sequence {3 6 2 5 1 4} is the maximum entry under Q_5, which is 63.

Table 2.6

Scheduling Table

Q_1	Q_2	Q_3	Q_4	Q_5
31	2	1	0	2
61	9	2	2	11
21	2	3	11	13
51	7	4	13	20
11	4	5	20	24
41	10	6	24	34
32	10	1	2	12
62	4	2	12	16
22	17	3	16	33
52	15	4	33	48
12	5	5	48	53
42	8	6	53	61
33	4	1	12	16
63	11	2	16	27
23	7	3	33	40
53	6	4	48	54
13	5	5	54	59
43	2	6	61	63

2.2. Direct Technique:

The basic concept of the direct technique for the flowshop scheduling problem involves decision rules to help fill each sequence - position by one of the candidate jobs. This procedure is directed to find the complete sequence which minimize the idle time. Johnson (80) has developed an algorithm for two-machine flow shop problem based on the above concept. The algorithm minimizes the accumulated idle time on the last machine, in processing each job which is similar to minimize the schedule time. Dudek and Teuton (38) have devised an algorithm based on Johnson's approach for the M-machine problem; however, counter-example has been given by Karush (82). The drawback is that the possible partial sequences have not been checked for dominance, while constructing the feasible sequences. Smith and Dudek (156) have modified the above algorithm, which overcomes the drawbacks to guarantee an optimal for the flow shop problem with arbitrary number of jobs and machines.

In order to discuss the basic idea of this algorithm, the following definitions and notations are used.

1. Candidate sequences are those partial sequences, which are generated through filling sequence position x except the last.
2. Dominated jobs are those jobs which are eliminated from further consideration as a possible candidate for a sequence-position x.
3. Dominated sequences are those partial sequences eliminated from further consideration.

4. A candidate set of jobs are those jobs which: (1) are not in the pre-sequence being considered; (2) have not been dominated; and (3) have not been used yet for dominance check.
5. Two or more partial sequences are considered equivalent when they have the same combinations of jobs but with different permutations.

The notations considered are

S_i Complete sequence, i, consisting of J jobs, $j_1\ j_2 \cdot \cdot \cdot\ j_J$

s pre-sequence consisting of x-1 scheduled jobs, $\{j_1\ j_2 \cdot \cdot \cdot\ j_{x-1}\}$

s_i partial sequence, i = 1, 2, . . ., n

$\bar{s}$ set of unscheduled jobs, $j_x, j_{x+1}, \cdot \cdot \cdot, j_J$

a_1, a_2 jobs included in $\bar{s}$ and competing for the sequence position x

$\bar{s}_1, \bar{s}_2$ represent all possible exclusive subsets of $\bar{s}$ excluding jobs a_1 and a_2

I(m, s) total idle time resulting from the processing of the pre-sequence s on machine m

k(m, sa) total processing time resulting from the processing of the pre-sequence s on machine m.

To clarify some of the above notations consider an example of nine jobs 1, 2, . . ., 9. Let s = {2, 5} , then $\bar{s}$ consists of the jobs 1, 3, 4, 6, 7, 8 and 9. Let jobs a_1 and a_2 be 1 and 9 respectively. The remaining jobs in the set $\bar{s}$ are then 3, 4, 6, 7 and 8. Consequently, one may consider two subsets, $\bar{s}_1$ and $\bar{s}_2$ such that

$\bar{s}_1$ consists of jobs 3 and 6,

$\bar{s}_2$ consists of jobs 4 and 8.

According to the above notation, s_1 consisting of the same jobs in s may have the sequence {5, 2}, thus s_1 and s_2 are equivalent.

This algorithm is based on two dominance checks:

1. Job dominance: consider the two sequences

$$S_1 = sa_1a_2\bar{s}_1\bar{s}_2 \qquad \text{and} \qquad S_2 = sa_2\bar{s}_1a_1\bar{s}_2$$

Since the presequence s consists of x - 1 jobs, jobs a_1 and a_2 are competing for the sequence position x. Job a_2 is eliminated from further consideration for this position in favor of job a_1 if the M-1 conditions

$$k\ (m, sa_2) \geq \max\ [k(m, sa_1\ a_2),\ k(m, sa_1)], \quad \ldots\ldots\ldots\ldots \quad (1)$$

$$m = 2, 3, \ldots, M.$$

are satisfied. If any of the above conditions is not satisfied, jobs a_1 and a_2 are retained for further consideration. The job dominance will ensure that no set of sequences $\{sa_2\}$ is discarded unless one equally good or better $\{sa_1a_2\}$ is retained.

2. Sequence dominance: Consider the two sequences

$$S_1 = s_1\bar{s} \qquad \text{and} \qquad S_2 = s_2\bar{s},$$

where the partial sequences s_1 and s_2 are equivalent. The partial sequence s_2 is eliminated from further consideration in favor of the partial sequence s_1, if the M - 1 conditions

$$I(m, s_1) \leq I(m, s_2), \quad m = 2, 3, \ldots, M \qquad (2)$$

are satisfied. If any of the above conditions is not satisfied, both partial sequences are retained for further consideration. The sequence dominance will ensure that no partial sequence $\{s_2\}$ will be discarded unless an equally good or better sequence $\{s_1\}$ is retained. It should be pointed out that this sequence dominance check has been also used by Ignall and Schrage (68).

For the dominance checks, the terms, I (m, s), K (m, sa), and R(m, sa) are presented below in mathematical forms. The idle time I (m, s), m = 1, 2, . . ., M is

$$I(1, s) = 0,$$

$$I(2, s) = \max_{1 \leq u \leq x-1} \left[\sum_{k=1}^{u} t_{j_k 1} - \sum_{k=1}^{u-1} t_{j_k 2} \right],$$

$$I(3, s) = \max_{1 \leq u \leq x-1} \left\{ \sum_{k=1}^{u} t_{j_k 2} - \sum_{k=1}^{u-1} t_{j_k 3} + \max_{1 \leq v \leq v-1} \left[\sum_{k=1}^{v} t_{j_k 1} - \sum_{k=1}^{v-1} t_{j_k 2} \right] \right\},$$

$$I(4, s) = \max_{1 \leq u \leq x-1} \left\{ \sum_{k=1}^{u} t_{j_k 3} - \sum_{k=1}^{u-1} t_{j_k 4} + \right.$$

$$\max_{1\le v\le x-1}\left[\sum_{k=1}^{v} t_{j_k 2} - \sum_{k=1}^{v-1} t_{j_k 3} + \max_{1\le w\le x-1}\left[\sum_{k=1}^{w} t_{j_k 1} - \sum_{k=1}^{w-1} t_{j_k 2}\right]\right]\}$$

$$\vdots$$

$$I(M,s) = \max_{1\le u\le x-1}\left\{\sum_{k=1}^{u} t_{j_k M-1} - \sum_{k=1}^{u-1} t_{j_k M} + \right.$$

$$\max_{1\le v\le u-1}\left[\sum_{k=1}^{u} t_{j_k,M-2} - \sum_{k=1}^{v-1} t_{j_k,M-3} + \cdots\right.$$

$$\left.\left. \cdots + \max_{1\le w\le t}\left[\sum_{k=1}^{w} t_{j_k 1} - \sum_{k=1}^{w-1} t_{j_k 2}\right] \cdots\right]\right\} \quad \ldots\ldots (3)$$

Consequently, the terms k(m, sa) and R(m, sa) which will be used for job dominance checks are

$$K(m, sa) = \sum_{j\in sa} t_{j,m-1} - \sum_{j\in s} t_{j_m} +$$

$$\max\,[I\,(m-1, s)\,,\ K(m-1, sa)], \quad \ldots\ldots\ldots\ (4)$$

$$m = 2, 3, \ldots, M.$$

where

$$K\,(0, sa) = 0,$$

$$I\,(0, s) = 0,$$