THE MACHINE SCHEDULING PROBLEM

by

ARUNKUMAR VASANTLAL PAREKH

B. E. (Mechanical), Sardar Vallabhbhai Vidyapeeth, 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

1968

Approved by:

Said Ashour

Major Professor

ACKNOWLEDGEMENT

The author wishes to express his sincere gratitude to his Major Professor, Dr. Said Ashour, for his kind guidance and assistance in carrying out the report in the most systematic manner.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In an industrial organization the objeetive of the management is to optimize the system. The problem of seheduling is eneountered in many spheres of the organization. Henee this problem forms a signifieant part of the eost eontrolling meehanism.

The seheduling problem may be elassified as: (1) seheduling arrivals or demands, that is determining the timing of the arrivals, (2) seheduling activities of large eomplex projeets, and (3) determining the sequenee in whieh a number of jobs are to be proeessed on various maehines.

The first type of these seheduling problems eonsists of obtaining an optimum balanee between eosts resulting from idle machines, and that of the arrivals waiting for serviee. Queuing theory is mostly utilized to solve this dynamie situation of seheduling problem. This is beeause, queuing theory deals with random demands (arrivals).

The seeond type of these problems deals with analyzing, planning and seheduling eomplex projeets. The problems are represented by means of networks, where eaeh network represents the possible sequenees to complete the projeet. The most eommon teehniques available for solving these networks are Projeet Evaluation and Review Technique (PERT) and Critieal

Path Method (CPM). The latter technique determines the expected completion times of the subprojects where as the former goes a step further and estimates the variances associated with these completion times.

Finally, the third type of these problems is referred to as the sequencing problem. In this problem, determining the optimal sequence of a number of tasks is difficult to resolve. Analytical solutions obtained for this type of problems are restricted to very simple cases. The assembly line balancing, the travelling-salesman, and the machine scheduling problem are special cases of this general category.

The machine scheduling problem arises whenever a number of jobs has to be processed on various machines. The problem in essence consists of determining a sequence in which the jobs are to be processed on the machines to achieve an objective. Because of the diversity and complexity of the problem, it is almost impractical to account for every factor in any single analysis. The problem becomes complicated when more than one machine of a given type exists, machine times and/or costs are of probabilistic nature, machines are subjected to breakdown and operators get injured. So far, several simplifications have been restricted on to the problem.

In the machine scheduling problem, the optimal sequence of operations for a job may be a function of the sequence of operations for other jobs. Hence in such case, it is usually neces-

sary to determine the optimal sequence for all jobs simultaneously. As a result, this problem can become one of considerable size and complexity. For example, consider a problem of six jobs to be processed on each of the 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 even on a high speed computer. Many of these sequences are technologically non-feasible. An exhaustive enumeration must consider all feasible sequences and eliminate the non-feasible. The next step is to select the optimal sequence. This report will present different approaches to the machine scheduling problem. The efficiency of the various techniques for solving this problem will also be reported.

1.1 Measure of Effectiveness

Once the scheduling problem is solved, the schedule is evaluated with respect to one of the various measures of effectiveness. The optimal solution is a function of the objective function. For an industry, the ultimate goal, in general, is to optimize profits. Beenhakker (11) has listed 27 system goals but many of them are redundancies. Researchers have agreed to the difficulties of measuring the effectiveness of a sequence when there is no common measure of value for the various desirable properties. Hence, for research work the most common objective has been minimization of the schedule

time. This is a simple measure and is also related to other criteria such as minimizing idle time of the machines and minimizing cost. In dynamic situations the effectiveness is also measured in terms of work-in-process inventory costs, meeting due dates, over-all flow time of jobs, and minimizing waiting time. The schedule time, T, which is to be minimized, can be expressed mathematically as

$$T = \sum_{m=1}^{M} \sum_{j=1}^{J} t_{jm} + \sum_{m=1}^{M} \sum_{j=1}^{J} I_{jm}$$

where,

t_{jm} = processing time of job j on machine m, and

I_{jm} = idle time immediately prior to processing job j on machine m.

The over all flow time of job j, f_j, may be expressed as

$$f_j = \sum_{m=1}^{M} t_{jm} + \sum_{m=1}^{M} I_{jm}.$$

Lateness, tardiness and earliness are three different means of comparing the actual completion time with the desired completion time. Lateness considers the algebraic difference for each job regardless of the sign of difference. Tardiness considers only positive differences — jobs which are completed after their due dates, and carliness considers only negative

differences — jobs completed ahead of their due dates.

The lateness for job j, L_j, may he expressed as

$$L_j = c_j - d_j$$

where,

c_j is the seheduled completion time of job j, and

d_j is the due date.

Therefore the tardiness, T_j, is defined as

$$T_j = \max[0, L_j]$$

and accordingly, the earliness, E_j, is defined as

$$E_j = \max[0, -L_j].$$

The interdependence of measure of effectiveness is important in comparison of scheduling procedures. It may be observed that for any given situation, mean flow time is directly proportional to the mean work-in-process inventory (as measured by the number of jobs); a scheduling method which minimizes mean flow-time also minimizes mean lateness, mean waiting-time and the mean number of jobs in the system.

1.2 Formulation of the Machine Scheduling Problem*

The machine scheduling problem consists of a number of jobs to be performed by a number of machines. Each job has to be

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

proeessed on all the machines in a specified machine ordering. The objective is to schedule the jobs on the machines so that the sequencc is optimal with respect to a eertain eriterion. In this paper the criterion selected is minimization of the schedule time. By schedule time is meant, the total time of processing all the jobs on the various machines.

The machine schcduling problem may be classified as:

1. Flow Shop: where all the jobs have the same machine ordering.
2. Job Shop: where each job has a different machine ordering.

Most of the researchers have used the terms sequeneing and schcduling interchangcably. However, in this report, a sequence will specify the order of processing a number of jobs on each machine, and a schedule will specify the starting and the completion times of various operations.

A summary of the necessary notation is next presented to formulate the job sequencing, machine ordering and processing time matrices of the machine scheduling problem. The notation is as follows:

J total number of jobs

M total number of machines

j job designation, j = 1, 2, ..., J

m machinc 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

The numbering of jobs and machines is arbitrary and it does not necessarily correspond to the sequence in which the jobs are processed on each machine or the order in which the machines process each job. Therefore, the sequence of jobs on each machine will be designated as $j_1, j_2, \ldots, j_x, \ldots, j_J$ with respect to a preconceived sequence while the machines will be designated as $m_1, m_2, \ldots, m_y, \ldots, m_M$ when considering a permutation of machines with respect to the preconceived order.

The various sets of job sequencings on each machine for a given sequence are designated as:

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

The above sets, one for each machine, may be combined in a (MxJ) matrix called the job sequencing matrix and denoted by S*. For the problem of J jobs and M machines, one of the possible sequencing matrices can be shown as

$$S^* = \begin{bmatrix} S_1^* \\ S_2^* \\ \cdot \\ \cdot \\ \cdot \\ S_N^* \end{bmatrix} = \begin{bmatrix} j_1{}^1 & j_2{}^1 & \cdots & j_x{}^1 & \cdots & j_J{}^1 \\ j_1{}^2 & j_2{}^2 & \cdots & j_x{}^2 & \cdots & j_J{}^2 \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ j_1{}^M & j_2{}^M & \cdots & j_x{}^M & \cdots & j_J{}^M \end{bmatrix}$$

The above job sequencing matrix shows that all the J jobs are processed on all the M machines in the same order.

Next, the various sets of machine orderings for each job are designated as:

$$M_j^* = \{jm_1\ jm_2 \cdots jm_y \cdots jm_M\}\ ,\ j = 1, 2, \ldots, J.$$

The above sets one for each job, may be combined in a (JxM) matrix called the machine ordering matrix and denoted by M*. This matrix for the above problem will become:

$$M^* = \begin{bmatrix} M_1^* \\ M_2^* \\ \cdot \\ \cdot \\ \cdot \\ M_J^* \end{bmatrix} = \begin{bmatrix} 1m_1 & 1m_2 & \cdots & 1m_y & \cdots & 1m_M \\ 2m_1 & 2m_2 & \cdots & 2m_y & \cdots & 2m_M \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ Jm_1 & Jm_2 & \cdots & Jm_y & \cdots & Jm_M \end{bmatrix}$$

For example, consider a problem of three jobs and two machines. The job sequencing matrix can be shown as:

$$S^* = \begin{bmatrix} S_1^* \\ S_2^* \end{bmatrix} = \begin{bmatrix} j_1{}^1 & j_2{}^1 & j_3{}^1 \\ j_1{}^2 & j_2{}^2 & j_3{}^2 \end{bmatrix} = \begin{bmatrix} 11 & 31 & 21 \\ 22 & 32 & 12 \end{bmatrix}$$

This indicates that machine 1 processes jobs in the sequence {1 3 2} and machine 2 processes jobs in the sequence {2 3 1}. It should be noted that $j_1{}^1$ means the job j_1 on machine 1, which may or may not be the same job as $j_1{}^2$.

The machine ordering matrix becomes:

$$M^* = \begin{bmatrix} M_1^* \\ M_2^* \\ M_3^* \end{bmatrix} = \begin{bmatrix} 1m_1 & 1m_2 \\ 2m_1 & 2m_2 \\ 3m_1 & 3m_2 \end{bmatrix} = \begin{bmatrix} 11 & 12 \\ 22 & 21 \\ 31 & 32 \end{bmatrix}$$

This matrix indicates that jobs 1 and 3 are processed on machine 1 first and on machine 2 last. Job 2 is processed on machine 2 first and on machine 1 last. Again $1m_1$ means job 1 on machine m_1 which is not necessarily the same as $2m_1$ or $3m_1$.

Since the machine ordering is specified, the processing time for each job on each machine must be given. These processing times are placed in a matrix form which is called the processing matrix and is denoted by

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

The machine scheduling problem may now be stated as: given τ^* and M^*, find the optimal sequence S^* which gives the minimum schedule time T.

Research workers usually specify their own models. The principal assumptions made on the models may be stated as follows:

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.
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 at each station.

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 is integer.

3.3 Set up and transportation times between machines are included in the processing time.

It is not that all available techniques need all of the above assumptions. In fact some of the assumptions are not required in certain techniques.

1.3 Literature Review:

Research in the machine scheduling problem has increased recently. This indicates the importance this problem has gained. The objective of the research has been to develop computationally efficient algorithms for arriving at optimal solutions. Except for very small size problems a practical procedure has not yet been developed.

The approaches available to solve the machine scheduling problem are: 1) Combinatorial Analysis, 2) Integer-Linear Programming, 3) Graphical, 4) Graphical-Dynamic Programming, 5) Schedule Algebras, 6) Heuristics, and 7) Simulation.

The most simple scheduling problem consists of J jobs and only one machine. For example, an integrated production line forms one machine problem. The trend towards automation will integrate all processes and hence may be considered as one machine problem. Smith (107), Jackson (57) and (58) have worked independently on this case. The criteria considered are minimizing maximum tardiness by obtaining a sequence based on due-dates; and minimizing sum of all waiting lines. Jackson has considered the cases where the machine may be left idle and also where a job, if any, must be placed on the machine. The computation procedure, however, is not efficient. Held and Karp (47) have developed dynamic programing approach for this case.

For flow shop problem having J jobs and two machines, Johnson (63) has developed a simple algorithm with the criterion of minimizing the schedule time. He has also extended the above algorithm to cover a special case of three machines. Jackson (59), Mitten (80) and Johnson (64) have generalized the results to some extent. Jackson (59) has considered the case when jobs have different machine orderings. Mitten (80) has dealt with some arbitrary time lags between the operations. Johnson (64) has considered the case with lags, where different job sequences are allowed. He has derived some rules which reduces the problem size from $(J!)^2$ to $(J!)$ sequences.

Dudek and Teuton (30) have extended Johnson's algorithm to solve the flow shop problem of J jobs and M machines. The procedure involves minimization of the cumulative idle time

on the last machine. Karush (65) has formulated a counterexample of three jobs and three machines in which the above algorithm failed to give an optimal sequence. As a result, Smith and Dudek (105) have revised the above algorithm but the amount of computations required cannot be appreciated.

For problems of two jobs and M machines, Akers and Friedman (5) have developed a non-numeric approach. It consists of applying logical means to cut down the number of sequences that must be searched for finding an optimal solution. As some of the sequences are dropped from further consideration without resort to specific processing times, the approach is called non-numeric. This approach is also applicable to job shop problems.

Hardgrave and Numhauser (46) have developed a graphical approach for problems having two jobs and M machines. This involves generation of a schedule network within a rectangle that represents the data of the problem. The shortest path in the network gives the optimal solution. Akers (4) also gives solution to this problem.

The combination of dynamic programing and graphical approaches which has been developed by Szwarc (109), makes the method of solution specially effective. It consists of finding the best path between two nodes. Optimal sequence is obtained when all nodes are considered. The case of J jobs and M machines has also been discussed.

An approach that has proved computationally more efficient is that of generating a small subset of complete set of feasible sequences. The Branch-and-Bound technique of Little et. al. (73) developed to solve the travelling-salesman problem has been used for solving the machine scheduling problem. Ignall and Schrage (56) have applied this technique to the two-and three-machine flow shop problem.

Lomnicki (74) has also applied this technique to three-machine flow shop problem. Brown and Lomnicki (20) have extended this to an arbitrary number of machines. McMahon and Burton (76) have also worked with this technique. Their computational experience involves up to 45 jobs and three machines.

Giffler and Thompson (40) have developed an algorithm for generating the subset of feasible sequences, which contain the optimal sequence. In practice, however due to computations, only a sample of this subset is generated.

Brooks and White (19) have modified Giffler and Thompson's algorithm by using lower bound as a decision rule. Computational experience shows that computer time increases rapidly with the problem size. This technique is also applicable to job shop problem.

Palmer (87) has presented a slope order method to obtain an approximate solution for the scheduling flow shop problem. His approach is based on heuristic arguments.

An approach which seems to be promising for the machine scheduling problem of J jobs and M machines is that of Integer

programing approach. This is made possible by Gomary's (43) integer programing algorithm. Presently, there are three published formulations to this problem. Bowman (18) estimates that formulating a simple problem involving three jobs and four machines, in his terms, would require an integer programing problem containing 300 to 600 variables, and many more constraints depending on the schedule time T. Wagner's formulation (111) is also of the same magnitude. The most compact one, that of Manne (75) requires 31 variables and 94 constraints. This appears to be a reasonable formulation. None of these authors claim practicality of their formulations.

Giglio and Wagner (42) have reported on computational experience with several methods. They have concluded that integer-linear programing approach does not converge fast enough to make it practical. This difficulty is likely to increase as problem size increases.

Dantzig (27) and (28) has proposed rounding linear programming solutions and also developed a shortest route subroutine for reducing the number of variables to use the simplex method. But it has been reported by Giglio and Wagner (42) that the solution is far from the optimal.

Heller (52) has done some work on developing a graph-theoretic approach to the scheduling problem. The problem is expressed in graph-theory term. Heller and Logemann (53) have developed an algorithm which is based on linear graph properties. This algorithm evaluates the feasible sequences.

An operation of processing job j on machine m for the return i is referred to as a node (mji). The nodes are linked as per the machine ordering. One of the J nodes is picked by the algorithm and is scheduled. The process is repeated until all operations are scheduled.

Heller (49) has done some experimentation on flow shop problem.. He has shown that the limit distribution of the schedule times is asymptotically normal as the number of jobs increases.

Ashour (6) and (7) has developed a decomposition approach for the machine scheduling problem of J jobs and M machines. The approach consists of decomposing the original problem into a number of smaller, more manageable subproblems, which minimizes the computational effort. He has found that the mean of the schedule times obtained by complete or partial enumeration is greater than that obtained by decomposition. This mean increases as the number of jobs in each subgroup decreases. His computational experiments consist of six to 40 jobs and three to ten machines.

Another approach is that of Heuristic rules, priority rules and combinations of these. Many of these rules have been compared by simulating their performances on computers. For the "due-date criterion", Gere (34) has experimented a heuristic approach. Conway (24), Dudek and Ghare (29), and Burstall (21) have also studied some heuristic rules.

A Monte Carlo version of Giffler and Thompson's (40) algorithm has also been studied. Although it does not guarantee an optimal schedule, but it does permit rapid computations for fairly large problems encountered in industry. It selects fairly large number of feasible sequences at random, and the shortest one can then be used. Giffler, et. al. (41) have reported on numerical experience with linear and Monte Carlo algorithm. They have found that a Monte Carlo process, that uses rules as guides in its random choices are considerable superior to a purely random choice device. Fisher and Thompson (33) have reported on a study in which they have devised some learning strategies to guide the program in its use of rules. Simulation makes it possible to find a sequencing procedure which is better than the rule of thumb techniques now used in practice. It also provides a useful laboratory for the further investigation of scheduling.

It is intended to illustrate some of the different techniques available for solving the machine scheduling problem, by sample problems.

CHAPTER II

COMBINATORIAL APPROACH

The combinatorial analysis approach for solving the machine scheduling problem represents quasi-enumeration techniques, and their efficiency depends on how effectively enumeration is curtailed. Several techniques within the concept of the combinatorial approach have been developed for solving the problem. Each technique has its own advantages and limitations. The various techniques considered in this paper are, Direct, Extended Direct, Branch-and-Bound, Lower Bound, and Boolean Algebra Techniques.

2.1 Direct Technique

The Direct technique has been developed by Johnson (63) and is applicable for the flow shop problem. In flow shop problem, the sequence that minimizes the cumulative idle time on the last machine, becomes the optimal sequence. Johnson's approach proceeds so that the cumulative idle time on the last machine is minimized. Hence, it is referred to as Direct Technique. However, this technique is feasible for flow shop problem of J jobs and two machines. A special case of three machines problem may be solved by this technique.

For solving the problem, the algorithm is based on two lemmas and one theorem. These are mentioned below without proof. For their proofs, see Appendix A.

Lemma 1: The sequence on either machine can be made the same as that of the other machine without loss of time.

Theorem 1: An optimal sequence is given by the following rule:

The job j_x precedes the job j_{x+1} if

$$\min[t_{j_x 1}, t_{j_{x+1} 2}] < \min[t_{j_{x+1} 1}, t_{j_x 2}] \qquad (2.1)$$

Lemma 2: Inequality (2.1) is transitive.

The algorithm may be summarized in the following steps.

Step 1: Arrange the processing times of the jobs on machines as follows:

Job Designation	Machine 1	Machine 2
1	t_{11}	t_{12}
2	t_{21}	t_{22}
.	.	.
.	.	.
.	.	.
j	t_{j1}	t_{j2}
.	.	.
.	.	.
.	.	.
J	t_{J1}	t_{J2}

Step 2: Examine all processing times, t_{jm}, for the minimum value.

2.1 If the minimum processing time is t_{j1}, schedule the corresponding job first on machine 1.

2.2 If the minimum processing time is t_{j2}, schedule the corresponding job last on machine 1.

Step 3: Cross off the job just assigned and repeat step 2 on the reduced set of processing times.

Step 4: Check the ties.

4.1 If the tie is among the processing times on one of the machines, schedule the job with the smallest designation first.

4.2 If the tie is for the same job on both machines, consider it as in step 2.1.

A flow shop problem of six jobs and two machines is presented to illustrate the algorithm. The processing time and machine ordering matrices are given below:

$$T^* = \begin{bmatrix} 6 & 7 \\ 12 & 2 \\ 4 & 6 \\ 3 & 11 \\ 6 & 8 \\ 2 & 14 \end{bmatrix}, \qquad M^* = \begin{bmatrix} 11 & 12 \\ 21 & 22 \\ 31 & 32 \\ 41 & 42 \\ 51 & 52 \\ 61 & 62 \end{bmatrix}$$

Applying the above algorithm step by step, computations are carried out.

Step 1: Arrange the above processing time matrix as follows:

j	t_{j1}	t_{j2}
1	6	7
2	12	2
3	4	6
4	3	11
5	6	8
6	2	14

Step 2: The minimum processing time is 2 units for job 6 on machine 1 and job 2 on machine 2. Therefore, job 6 is scheduled first on machine 1, and job 2 is scheduled last on machine 2.

Step 3: Jobs 6 and 2 are crossed off.

Repeating step 2 on the reduced set of processing times, the optimal sequence {6 4 3 1 5 2} is obtained. The schedule time is obtained as 50. This may be shown in the Gantt Chart, Figure 2.1.

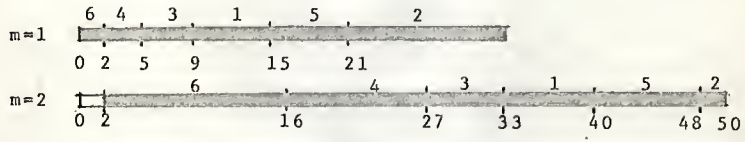

Figure 2.1 Gantt Chart for a Flow Shop Problem of Size (6x2).

The above algorithm has been extended for a special case of three machines problem when the inequality

$$\min[t_{j_x 1}] \geq \max[t_{j_x 2}]$$

or

$$\min[t_{j_x 3}] \geq \max[t_{j_x 2}],$$

holds, a solution similar to that of the two machines problem can be found. Again, two lemmas and one theorem supporting the algorithm for this special case are presented below without proof. For their proofs, see Appendix A.

Lemma 3: An optimal sequence can be reached if the same ordering is assumed for all the jobs.

Theorem 2: An optimal sequence is given by the following rule:

The job j_x precedes the job j_{x+1} if

$$\min[t_{j_x 1}+t_{j_x 2}, t_{j_{x+1} 3}+t_{j_{x+1} 2}] < \min[t_{j_{x+1} 1}+t_{j_{x+1} 2}, t_{j_x 3}+t_{j_x 2}] \quad (2.2)$$

In case of an equality either job is permissible.

Lemma 4: The inequality (2.2) is transitive.

In this special case, the processing times of the three machines are reduced to that of a two machines problem such that

$t_{j_x 1}$ is replaced by $(t_{j_x 1} + t_{j_x 2})$,

and

$t_{j_x 2}$ is replaced by $(t_{j_x 2} + t_{j_x 3})$.

A sample problem of six jobs and three machines is presented below to illustrate the algorithm. The processing time and machine ordering matrices are:

$$\tau^* = \begin{bmatrix} 4 & 5 & 8 \\ 9 & 6 & 10 \\ 8 & 2 & 6 \\ 6 & 3 & 7 \\ 5 & 4 & 11 \\ 7 & 3 & 6 \end{bmatrix} \qquad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix}$$

From the above processing time matrix

$$\min[t_{j_x 1}] = 4,$$

$$\max[t_{j_x 2}] = 6,$$

and

$$\min[t_{j_x 3}] = 6.$$

Since $\max[t_{j_x 2}] \leq \min[t_{j_x 3}]$, the special condition is satisfied. Hence, the three machines problem can be reduced to an equivalent two machines problem. The processing times of this reduced problem are $(t_{j_x 1} + t_{j_x 2})$ and $(t_{j_x 2} + t_{j_x 3})$.

Now, applying the above algorithm, the processing times are arranged as follows:

j	$(t_{j_x 1} + t_{j_x 2})$	$(t_{j_x 2} + t_{j_x 3})$
1	9	13
2	15	16
3	10	8
4	9	10
5	9	15
6	10	9

Proceeding step by step as described in solving the two machines problem the optimal sequence is {1 4 5 2 6 3} with schedule time of 57. This may be shown in Gantt Chart, Figure 2.2.

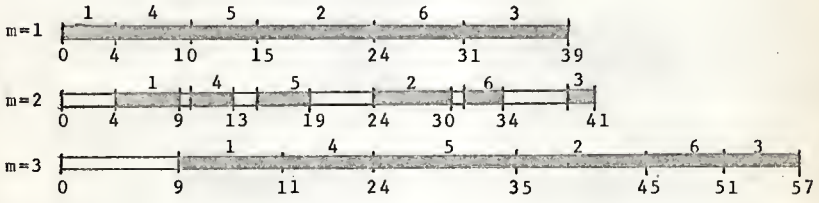

Figure 2.2 Gantt Chart for a Flow Shop Problem of Size 6x3

2.2 Extended Direct Technique

Dudek and Teuton (30) have developed an algorithm for solving the flow shop problem of J jobs and M machines. This algorithm is an extension to that of Johnson (63). Therefore, it is referred to as Extended Direct Technique.

In this technique, the first x-1 sequence-positions are assumed to be filled. The sequence-position x has to be filled. However, in solving a new problem, there are no presequences. To fill the sequence-position x, two partial sequences S_1 and S_2 are compared by M-1 conditions discussed later. These two partial sequences differ in sequence-positions x and x+1 as shown below:

$$S_1 = \{j_1, j_2, \cdots, j_{x-1}, j_x, j_{x+1}\}$$

$$S_2 = \{j_1, j_2, \cdots, j_{x-1}, j_{x+1}, j_x\}$$

The decision rule for scheduling is that the job in the sequence-position x will precede the job in the sequence-position x+1 if M-1 conditions are satisfied.

It has been claimed that this algorithm would generate an optimal solution, but Karush (65) has developed a counterexample. The reason is that in the partial sequence S_2, job j_{x+1} is necessarily followed by job j_x. This does not consider the case where, in an optimal sequence, job j_{x+1} is followed by a job other than j_x. Because of this difficulty, the basic algorithm has been modified by Smith and Dudek (105). It consists of comparing the two partial sequences S_1 and S_3:

$S_1 = \{j_1, j_2, \ldots, j_{x-1}, j_x, j_{x+1}\}$

$S_3 = \{j_1, j_2, \ldots, j_{x-1}, j_{x+1}\}$

As in the basic algorithm, x-1 sequence-positions are assumed to be filled. However, S_3 has one less job in the sequence. Hence, job j_{x+1} is not necessarily followed by job j_x but, that position is open for any of the unscheduled jobs $j_x, j_{x+2}, \ldots, j_J$. This modified approach guarantees an optimal solution. The following notation is considered:

s — a presequence consisting of x-1 scheduled jobs

a_1, a_2 — jobs competing for the sequence-position x and are taken from the unscheduled subset.

I(m,s) — idle time on machine m through all jobs included in the presequence s.

In order to facilitate the description of the algorithm, some definitions are considered:

1. Candidate sequences are those partial sequences generated through any sequence-position except the last.
2. Dominated jobs are those jobs eliminated from consideration as a possible candidate for a sequence-position.
3. Dominated sequences are the partial sequences eliminated from consideration.
4. A candidate set of jobs are those jobs which
   4.1 are not in the presequence
   4.2 have not been dominated

4.3 have not been used for dominance check.

The algorithm is based on two dominance checks: one is for a job and the other is for a sequence. Dominance of a job or a sequence is checked by M-1 conditions such that

$$H(m,sa_2) \geq \max[H(m,sa_1), H(m,sa_1a_2)], \qquad (2.3)$$

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

where

$$H(m,sa_2) = K(m,sa_2) + \sum_{j\epsilon s} t_{jm} - \sum_{j\epsilon s} t_{j1}, \qquad (2.4)$$

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

$$H(m,sa_1) = K(m,sa_1) + \sum_{j\epsilon s} t_{jm} - \sum_{j\epsilon s} t_{j1} \qquad (2.5)$$

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

$$H(m, sa_1a_2) = K(m,sa_1a_2) + \sum_{j\epsilon s} t_{jm} - \sum_{j\epsilon s} t_{j1}, \qquad (2.6)$$

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

and

$$K(m,sa_1) = \sum_{j\epsilon sa_1} t_{j,m-1} - \sum_{j\epsilon s} t_{jm} + \max[I(m-1,s), K(m-1,sa_1)] \qquad (2.7)$$

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

Note that $K(0,sa_1) = I(0,sa_1) = 0$.

The expressions (2.4), (2.5), and (2.6) may be simplified for computational purposes as follows:

$$H(m,sa_2) = \sum_{j\epsilon s} t_{j,m-1} + t_{a_2m-1} - \sum_{j\epsilon s} t_{jm} + \max[I(m-1,s), K(m,sa_2)]$$

Substitute from 2.8 for $K(m,sa_2)$

$$= \sum_{j\epsilon s} t_{j,m-1} + t_{a_2m-1} - \sum_{j\epsilon s} t_{jm} + \max[R(m-1,s), H(m-1,sa_2)] - \sum_{j\epsilon s} t_{j,m-1} + \sum_{j\epsilon s} t_{j1}$$

$$= t_{a_2 m-1} + \max[R(m-1,s),\ H(m-1,sa_2)] \qquad (2.8)$$

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

Similarly,

$$H(m,sa_1) = t_{a_1 m-1} + \max[R(m-1,s),\ H(m-1,sa_1)] \qquad (2.9)$$

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

and

$$H(m,sa_1a_2) = t_{a_1 m-1} + t_{a_2 m-1} - t_{a_1 m} + \max[R(m-1,s),\ H(m-1,sa_1,\ H(m-1,sa_1a_2)] \qquad (2.10)$$

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

where

$$R(m,sa_1) = I(m,sa_1) + \sum_{j\varepsilon s} t_{jm} - \sum_{j\varepsilon s} t_{j1},\ m = 2,\ 3,\ \ldots,\ M \qquad (2.11)$$

For simplicity, the M-1 conditions appearing in (2.3), may now be expressed in terms of processing times.

Condition 1:

$$H(2,sa_2) \geq \max[H(2,sa_1),\ H(2,sa_1a_2)], \qquad (2.12)$$

or

$$t_{a_2 1} \geq \max[t_{a_1 1},\ (t_{a_1 1} + t_{a_2 1} - t_{a_1 2})] \qquad (2.13)$$

which is obtained by substituting

$$H(2,sa_2) = K(2,sa_2) + \sum_{j\varepsilon s} t_{j2} - \sum_{j\varepsilon s} t_{j1}$$

$$= \sum_{j\varepsilon s} t_{j1} + t_{a_2 1} - \sum_{j\varepsilon s} t_{j2} + \sum_{j\varepsilon s} t_{j2} - \sum_{j\varepsilon s} t_{j1}$$

$$= t_{a_2 1}.$$

Similarly,

$$H(2,sa_1) = t_{a_1 1},$$

and

$$H(2,sa_1a_2) = t_{a_1 1} + t_{a_2 1} - t_{a_1 2} .$$

Each of conditions 2 through M-1 may be expressed in terms of the previous condition as follows:

Condition 2:

$$H(3,sa_2) \geq \max[H(3,sa_1), H(3,sa_1a_2)]$$

or

$$t_{a_2 2} + \max[R(2,s), H(2,sa_2)]$$
$$\geq \max\{t_{a_1 2} + \max[R(2,s), H(2,sa_1)], t_{a_1 2} +$$
$$t_{a_2 2} - t_{a_1 3} + \max[R(2,s), H(2,sa_1),$$
$$H(2,sa_1a_2)]\} \qquad (2.14)$$

Similarly, the other conditions may be obtained. However, the last condition will appear as follows:

Condition M-1:

$$H(M,sa_2) \geq \max[H(M,sa_1), H(M,sa_1a_2)]$$

or

$$t_{a_2 M-1} + \max[R(m-1,s), H(m-1,sa_2)]$$
$$\geq \max\{t_{a_1 M-1} + \max[R(M-1,s), H(M-1,sa_1)], t_{a_1 M-1} +$$
$$t_{a_2 M-1} - t_{a_1 M} + \max[R(M-1,s), H(M-1,sa_1),$$
$$H(M-1,sa_1a_2)]\} \qquad (2.15)$$

The algorithm may now be summarized in the following steps.

Step 1: Calculate R(m,s) for the presequence such that

$$R(m,s) = I(m,sa_1) + \sum_{j\varepsilon s} t_{jm} - \sum_{j\varepsilon s} t_{j1}, \quad m = 2, 3, \ldots M$$

Step 2: Select job a_1 from the unscheduled subset of jobs, such that

$t_{a_1 1}$ is $\min_{j \neq s}[t_{j1}]$.

Step 3: Check $t_{a_1 2}$:

3.1 if $t_{a_1 2} \geq t_{a_1 1}$, condition 1 is satisfied. Go to step 4.

3.2 if $t_{a_1 2} < t_{a_1 2}$, go to step 7.

Step 4: Select other job a_2 from the unscheduled subset and check conditions 2 through M-1 such that

$H(m,sa_2) \geq \max[H(m,sa_1), H(m,sa_1a_2)]$,

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

4.1 if all conditions are satisfied job a_2 is dominated, go to step 5.

4.2 if one condition is not satisfied, stop checking further and retain job a_2 in the candidatc subset.

Step 5: Repeat step 4 on all other jobs as job a_2 in the candidate subset.

Step 6: Repeat step 2 for all jobs in the candidate subset using each other job as a_1.

Step 7: If more than one presequence exists, repeat steps 1 through 7 for each of these presequences.

Step 8: Devclop candidate sequences, those partial sequences generated through any sequence-position except the J-1, by placing each undominated job in the sequence-position

x, and arrange the sequences that are permutations of the same jobs as s_1, s_2, ..., s_k.

Step 9: Calculate $I(m,s_i)$, $m = 2, 3, \ldots, M$; $i = 1, 2, \ldots, k$.

Step 10: Check the following inequality:

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

10.1 If it is satisfied for all m, this means that sequence s_2 is dominated. Go to step 11.

10.2 If it is not satisfied for all m, this indicates that the sequence s_1 is dominated. Go to step 12.

10.3 If it is satisfied for only some m, neither sequence s_1 or s_2 is dominated. Go to step 11.

Step 11: Repeat step 10 using sequences s_3 through s_k in place of s_2.

Step 12: Repeat step 10 using the next sequence that has not been dominated in place of s_1.

Step 13: Repeat steps 1 through 12 until the first J-2 sequence positions are filled.

Step 14: For each candidate sequence, compare $I(M,sa_1a_2)$ and $I(M,sa_2a_1)$.

14.1 If $I(M,sa_1a_2 < I(M,sa_2a_1)$ then, the sequence sa_1a_2 is feasible.

14.2 If $I(M,sa_1a_2 > I(M,sa_2a_1)$ then, the sequence sa_2a_1 is feasible.

Step 15: Select the sequence(s) from the set of feasible sequences that has the minimum schedule time T*. This is one of the optimal sequences.

A sample flow shop problem of six jobs and three machines is solved to illustrate the technique. The processing time and machine ordering matrices are:

$$\tau^* = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \\ 4 & 6 & 8 \\ 3 & 11 & 7 \\ 6 & 8 & 10 \\ 2 & 14 & 12 \end{bmatrix}, \qquad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix}$$

The number of conditions for job and sequence dominance checks is M-1 or 2.

Condition 1:

$$t_{a_2 1} \geq \max[t_{a_1 1}, (t_{a_1 1} + t_{a_2 1} - t_{a_1 2})]$$

Condition 2:

$$t_{a_2 2} + \max[R(2,s), t_{a_2 1}] \geq \max\{t_{a_1 2} + \max[R(2,s), t_{a_1 1}],$$

$$t_{a_1 2} + t_{a_2 2} - t_{a_1 3} + \max[R(2,s), t_{a_1 1}, t_{a_1 1} + t_{a_2 1} - t_{a_1 2}]\}$$

The presequence is empty. Therefore $R(2,s) = 0$

min. t_{j1} is = 2 for job 6. Hence $a_1 = 6$. Now, as $t_{62} > t_{61}$, condition 1 is satisfied. According to step 3.1, condition 2 remains to be checked for job 6 versus each of the remaining jobs.

Considering $a_2 = 1$ according to step 4 and checking job 6 versus job 1.

Condition 2:

$$7 + \max[0\ 6] \geq \max\{14 + \max[0\ 2];\ 14 + 7 - 12 + \max[0\ 2\ 2 + 6 - 14]\}$$

or

$$13 \geq \max[16\ 11].$$

This condition is not satisfied for job 6 versus job 1. Therefore, check job 6 versus job 2.

Condition 2:

$$2 + \max[0\ 12] \geq \max\{14 + \max[0\ 2];\ 14 + 2 - 12 + \max[0\ 2\ 2 + 12 - 14]\}$$

or

$$14 \geq \max[16\ 8].$$

This condition is also not satisfied for job 6 versus job 2. Therefore, check job 6 versus job 3.

Condition 2:

$$6 + \max[0\ 4] \geq \max\{14 + \max[0\ 2];\ 14 + 6 - 12 + \max[0\ 2\ 2 + 4 - 14]\}$$

or

$$10 \geq \max[16\ 10]$$

This also is not satisfied. Thus, check job 6 versus job 4:

Condition 2:

$$11 + \max[0\ 3] \geq \max\{14 + \max[0\ 2];\ 14 + 11 - 12 + \max[0\ 2\ 2 + 3 - 14]\}$$

or

$$14 \geq \max[16\ 15].$$

This condition is not satisfied for job 6 versus job 4. Therefore, check job 6 versus job 5.

Condition 2:

$$8 + \max[0\ 6] \geq \max\{14 + \max[0\ 2];\ 14 + 8 - 12 + \max[0\ 2\ 2 + 6 - 14]\}$$

or

$$14 \geq \max[16\ 12].$$

It is also not satisfied.

At this point, as none of the conditions are satisfied, all jobs are still candidates for sequence-position 1. Hence, we repeat step 2 such that for job 4

$$\min[t_{j1}] = 3.$$

Hence $a_1 = 4$. Again, as $t_{42} > t_{41}$, condition 1 is satisfied; and condition 2 remains to be checked for job 4 versus each of the remaining jobs for sequence-position 1. Repeating the above procedure, the results are summarized below.

Job 4 versus job j	Condition 2	Result
4 vs. 1	$13 \geq \max[14\ 14]$	not satisfied.
4 vs. 2	$14 \geq \max[14\ 10]$	not satisfied.
4 vs. 3	$10 \geq \max[14\ 13]$	not satisfied.
4 vs. 5	$14 \geq \max[14\ 15]$	not satisfied.
4 vs. 6	$16 \geq \max[14\ 21]$	not satisfied.

Again, none of the conditions are satisfied. This means that job 4 does not dominate any job for first sequence-position.

Returning to step 2, next job to be checked for first sequence-position is job 3. As $t_{32} > t_{31}$, condition 1 is satisfied. Checking condition 2 for job 3 versus all jobs in succession, the results are summarized below.

Job 3 versus job j	Condition 2	Result
3 vs. 1	$13 \geq \max[10\ 9]$	Satisfied.
3 vs. 2	$14 \geq \max[10\ 10]$	Satisfied.
3 vs. 4	$14 \geq \max[10\ 13]$	Satisfied.
3 vs. 5	$14 \geq \max[10\ 10]$	Satisfied.
3 vs. 6	$16 \geq \max[10\ 16]$	Satisfied.

Job 3 dominates all jobs for first sequence position. At this point, sequence dominance check cannot be made as the presequence is empty. To fill the second sequence-position, steps 1 and 2 are repeated.

Now, the presequence, s = {3}. From equation (2.11)

$$R(2,\underline{3}) = I(2,3) + t_{32} - t_{31}$$

$$= t_{31} + t_{32} - t_{31}$$

or

$$R(2,\underline{3}) = 4 + 6 - 4$$

$$= 6 .$$

Excluding job 3 from consideration,

$$\min_{j}[t_{j1}] = 2, \qquad j \neq 3$$

for job 6. As $t_{62} > t_{61}$, condition 1 is satisfied. Job 6 is to be checked against jobs 1, 2, 4, and 5 for dominance. Check first job 6 versus job 1.

Condition 2:

$$7 + \max[6\ 6] \geq \max\{14 + \max[6\ 2];\ 14 + 7 - 12] + \max[6\ 2\ 2 + 6 - 14]\},$$

or

$$13 \geq \max[20\ 15]$$

This is not satisfied. Check job 6 versus job 2.

Condition 2:

$$2 + \max[6\ 12] \geq \max\{14 + \max[6\ 2];\ 4 + 2 - 12 + \max[6\ 2\ 2 + 12 - 14]\}$$

or

$$14 \geq \max[20\ 10]$$

This is not satisfied. Check job 6 versus job 4.

Condition 2:

$$11 + \max[6\ 3] \geq \max\{14 + \max[6\ 2];\ 14 + 11 - 12 + \max[6\ \ 2\ \ 2 + 3 - 14]$$

or

$$17 \geq \max[20\ 19]$$

This is not satisfied. Check job 6 versus job 5.

Condition 2:

$$8 + \max[6\ 6] \geq \max\{14 + \max[6\ 2];\ 14 + 8 - 12 + \max[6\ \ 2\ \ 2 + 6 - 4]\}$$

or

$$14 \geq \max[20\ 16]$$

This is also not satisfied. Job 6 does not dominate. Hence, as before, job 4 is selected. As $t_{41} > t_{42}$ condition 1 is satisfied. For condition 2, the results are summarized as below.

Job 4 versus job j	Condition 2	Result
4 vs. 1	$13 \geq \max[17\ 17]$	Not satisfied.
4 vs. 2	$14 \geq \max[17\ 12]$	Not satisfied.
4 vs. 5	$14 \geq \max[17\ 18]$	Not satisfied.
4 vs. 6	$20 \geq \max[17\ 24]$	Not satisfied.

As none are satisfied, following step 6, the next jobs are 1 and 5, each with processing time of 6. Both the jobs compete for sequence-position 2. Dominance checks will be made for both jobs. Selecting first job 1, condition 1 is satisfied as

$t_{12} > t_{11}$. Summarizing the results for job 1 against jobs 2, 4, 5, and 6 for condition 2, we obtain,

Job 1 versus job j	Condition 2	Result
1 vs. 2	$14 \geq \max[13\ 17]$	Not satisfied.
1 vs. 4	$17 \geq \max[13\ 21]$	Not satisfied.
1 vs. 5	$14 \geq \max[13\ 18]$	Not satisfied.
1 vs. 6	$20 \geq \max[13\ 24]$	Not satisfied.

Job 1 does not dominate over any of the jobs 2, 4, 5, and 6. Selecting job 5, condition 1 is satisfied as $t_{52} > t_{51}$. Summarizing the results for job 5 against jobs 1, 2, 4, and 6, for condition 2, we get

Job 5 versus job j	Condition 2	Result
5 vs. 1	$13 \geq \max[13\ 11]$	Satisfied.
5 vs. 2	$14 \geq \max[14\ 10]$	Satisfied.
5 vs. 4	$17 \geq \max[14\ 15]$	Satisfied.
5 vs. 6	$20 \geq \max[14\ 18]$	Satisfied.

As condition 2 is satisfied for all jobs, job 5 dominates the second sequence-position.

Following step 8, the candidate sequences are

$s_1 = 35$ and $s_2 = 53$.

$I(m,s_i)$, $i = 1, 2$, are now calculated for machines 2 and 3 such that

$$I(2, \underline{35}) = t_{31}$$

$$= 4,$$

$$I(3, \underline{35}) = t_{31} + t_{32}$$

$$= 4 + 6$$

$$= 10,$$

$$I(2, \underline{53}) = t_{51}$$

$$= 6,$$

and

$$I(3, \underline{53}) = t_{51} + t_{52}$$

$$= 6 + 8$$

$$= 14 .$$

It is observed that

$$I(m,s_1) < I(m,s_2) , \qquad m = 2, 3.$$

Hence, according to step 10.1, the partial sequence s_1 dominates the partial sequence s_2.

At this point, the partial sequence is $s = \{3\ 5\}$. Following the step 13, the steps 1 through 12 are repeated till the first four sequence-positions are filled. The candidate sequences obtained are

$\{3\ 5\ 6\ 4\}$, $\{3\ 5\ 4\ 1\}$, $\{3\ 5\ 1\ 2\}$,

$\{3\ 5\ 6\ 1\}$, $\{3\ 5\ 6\ 2\}$, $\{3\ 5\ 4\ 2\}$.

Following the step 14, the idle times on machine M, $I(M,sa_1a_2)$, may be obtained by drawing Gantt Charts.

The Gantt Chart for the sequence $\{3\ 5\ 6\ 4\ 2\ 1\}$ is given in Figure 2.3 and $I(3, \underline{356421})$ is 14. Repeating this step, the following undominated sequences may be optimal.

{3 5 6 4 1 2}, {3 5 6 4 2 1},

{3 5 1 2 6 4}, {3 5 6 1 2 4},

{3 5 6 2 4 1}, {3 5 4 2 6 1},

{3 5 4 1 2 6}.

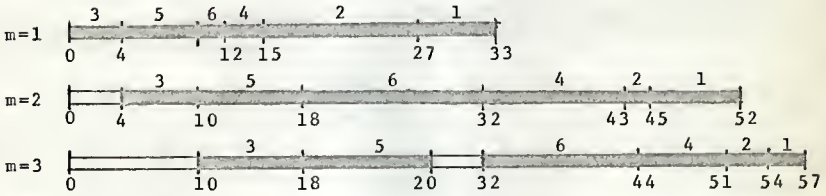

Figure 2.3 Gantt Chart for a Flow Shop Problem of Size (6x3)

The processing times of jobs 6 and 4 on machine 1 are less than that of job 3. Hence, jobs 6 and 4 are candidates for the sequence-position 1. Repeating steps 1 through 15 by placing jobs 6 and 4 in the first sequence-position, some more undominated sequences are obtained.

According to step 15, the sequence(s) with minimum schedule time are the optimal sequence(s). Referring to Table 2.1, the sequence numbers 1, 2, and 5 are optimal with schedule time of 57.

Table 2.1

Table of Feasible Sequences

Sequence Number	Sequence	Schedule Time
1	3 5 6 4 1 2	57
2	3 5 6 4 2 1	57
3	3 5 1 2 6 4	63
4	3 5 6 1 2 4	59
5	3 5 6 2 4 1	57
6	3 5 4 2 6 1	60
7	3 5 4 1 2 6	64
8	4 3 1 2 5 6	65
9	4 3 5 1 2 6	63
10	4 3 5 2 6 1	59
11	6 4 3 1 2 5	60
12	6 3 1 5 2 4	59
13	6 3 5 2 1 4	59
14	6 4 3 5 1 2	59
15	6 4 3 2 5 1	59
16	6 3 1 2 5 4	59

The number of sequences generated by the algorithm is 16. The algorithm guarantees an optimal solution. However, it should be noted that the procedure involves an excessive amount of computation.

2.3 Branch-and-Bound Technique

The Branch-and-Bound technique which is used to solve the flow shop scheduling problem of J jobs and M machines has been originally developed by Little, et. al. (73) for solving the travelling salesman problem. Ignall and Schrage (56) have

applied the above technique to flow shop problem consisting of two and three machines and up to nine jobs. Lomnicki (74) has applied this technique to a three machine problem and later, Brown and Lomnicki (20) have extended it to flow shop problems having arbitrary number of machines.

The concept of this technique is that the solution may be found by taking the set of feasible sequences, splitting it up into disjoint subsets, finding the lower bound on the objective function for each subset with the smallest lower bound. This technique may best be described by a tree which is referred to as a scheduling tree. It consists of nodes, each of which represents partial or complete sequence of jobs. The tree starts with the node "ALL" which will be referred to as level 0 in the scheduling tree, Figure 2.4. Level 1 consists of J nodes, each of which has one job. Further, the node(s) which have the least lower bound(s) are branched into J-1 nodes of level 2. Each of these J-1 nodes consists of a sequence of two jobs. As one moves down the scheduling tree, the number of nodes decreases by one at each level, and the number of jobs in each node increases by one. The labelling of lower bounds at each of the nodes helps identify the node(s) with least lower bound(s) and to discard those with higher lower bounds. The lower bound cannot be decreased as one moves down the tree. The maximum number of nodes to be explored is $J + J(J-1) + J(J-1)(J-2) + \ldots + J!$. However the algorithm requires the exploration of at least $J + (J-1) + (J-2) + \ldots + 2$ nodes and may be more in most of the cases.

Lomnicki (74) has defined lower bound for the node n on machine m as,

$$g_n^m = c_n^m + \sum_{\substack{j'=1,\\ j'\neq n}}^{J} t_{j'm} + \min_{j'} \sum_{m'=m+1}^{M} t_{jm'} \quad , \quad \begin{matrix} j = 1, 2, \ldots, J, \\ m = 1, 2, \ldots, M, \end{matrix} \qquad (2.16)$$

and

$$g_n^M = c_n^M + \sum_{\substack{j'=1\\ j'\neq n}}^{J} t_{j'm} \quad , \qquad (2.17)$$

where the first term on right hand side of equation (2.16) is the completion time of the sequence of a number of jobs represented by node n on machine m; the second term is the processing time for the remaining jobs on the machine m; and the third term is the minimum processing time to process the last job on the remaining machines.

The algorithm may now be stated as follows:

Step 1: Let L = 1.

Step 2: Calculate the lower bounds at level L, for all nodes on all machines such that

$$g_n^m = c_n^m + \sum_{\substack{j'=1,\\ j'\neq n}}^{J} t_{j'm} + \min_{j'} \sum_{m'=m+1}^{M} t_{jm'} \quad , \quad \begin{matrix} j = 1, 2, \ldots, J, \\ m = 1, 2, \ldots, M, \end{matrix}$$

and

$$g_n^M = c_n^M + \sum_{\substack{j'=1\\ j'\neq n}}^{J} t_{j'm} \quad ,$$

where c_n^m is the completion time of the partial sequence, n on machine m.

Step 3: Find g_n for each node such that

$$g_n = \max[g_n^1, g_n^2, \ldots, g_n^M] \quad .$$

Step 4: Find the nodes which have g* such that

$$g^* = \min_n[g_n] \quad .$$

Step 5: If L = 1 go to step 6. Otherwise, check the lower bounds.

5.1 If g*'s are greater than that of the branched node, terminate search in this direction. Branch at the node which has the second lowest lower bound. Go to step 2.

5.2 If g*'s are not greater than that of the branched node, go to step 6.

Step 6: Branch off at the nodes which have g*.

Step 7: Check L.

7.1 If $L \leq J-1$, let L = L+1 and go to step 2.

7.2 If $L > J-1$, go to step 8.

Step 8: Select the node which represents the minimum g*. This is the schedule time of the optimal sequence shown in the node.

To illustrate the above algorithm, the same problem presented in subsection 2.2 is solved. For convenience, the processing time and machine ordering matrices of this problem are reproduced below:

$$\tau^* = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \\ 4 & 6 & 8 \\ 3 & 11 & 7 \\ 6 & 8 & 10 \\ 2 & 14 & 12 \end{bmatrix} \qquad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix}$$

In solving the problem, the steps of the above algorithm are followed.

Following step 2, the lower bounds are found by computing the completion times such that, for node 1,

$$c_1^1 = t_{11},$$

$$c_1^2 = c_1^1 + t_{12},$$

and

$$c_1^3 = c_1^2 + t_{13}.$$

or

$$c_1^1 = 6 \qquad = 6,$$

$$c_1^2 = 6 + 7 = 13,$$

and

$$c_1^3 = 13 + 3 = 16.$$

Hence, the lower bounds are computed as follows.

$$g_1^1 = 6 + (12 + 4 + 3 + 6 + 2) + \min[5\ 14\ 18\ 18\ 26] = 38,$$

$$g_1^2 = 13 + (2 + 6 + 11 + 8 + 14) + \min[3\ 8\ 7\ 10\ 12] = 57,$$

and

$$g_1^3 = 16 + (3 + 8 + 7 + 10 + 12) \qquad = 56.$$

According to step 3, find g_n for each node such that

$$g_n = \max[g_1^1, g_1^2, g_1^3]$$

or

$$g_n = \max[38\ 57\ 56]$$
$$= 57.$$

Proceeding as above for nodes 2, 3, 4, 5 and 6 at level 1, the results are summarized in Table 2.2.

Following step 4, the least lower bound is for node 3. As L = 1, step 5 has to be skipped and branching is done at node 3. As in step 7, L is increased by 1 and returning to step 1, completion times of nodes on all three machines are determined at level 2 as follows:

For node 31,

$$c_{31}^1 = c_3^1 + t_{11},$$

$$c_{31}^2 = \max[c_{31}^1, c_3^2] + t_{12},$$

and

$$c_{31}^3 = \max[c_{31}^2, c_3^3] + t_{13},$$

or

$$c_{31}^1 = 4 + 6 = 10,$$

$$c_{31}^2 = \max[10\ 10] + 7 = 17,$$

and

$$c_{31}^3 = \max[17\ 18] + 3 = 21.$$

The lower bounds are:

$$g_{31}^1 = 10 + (12 + 3 + 6 + 2) + \min[5\ 18\ 18\ 26] = 38,$$

Table 2.2

Table of Lower Bounds

	Node	Completion Times			Lower Bounds			
	n	c_n^1	c_n^2	c_n^3	g_n^1	g_n^2	g_n^3	g_n
Level 1								
	1	6	13	16	38	57	56	57
	2	12	14	17	43	63	58	63
	3	4	10	18	38	55	53	55*
	4	3	14	21	38	54	57	57
	5	6	14	24	38	57	57	57
	6	2	16	28	38	53	59	54
Level 2								
	31	10	17	21	38	55	53	55
	32	16	18	21	43	61	53	61
	34	7	18	25	38	52	56	56
	35	10	18	28	38	55	53	55*
	36	6	20	32	38	51	59	59
Level 3								
	351	16	25	31	38	55	53	55
	352	22	24	31	43	59	53	59
	354	13	29	36	38	55	54	55
	356	12	32	44	38	55	57	57*
Level 4								
	3561	18	39	47	38	55	57	57
	3562	24	34	47	43	55	57	57
	3564	15	43	51	38	57	54	57*
Level 5								
	35641	21	50	54	38	55	57	57
	35642	27	45	74	43	55	57	57

* indicates the node at which branching is done to obtain the optimal sequence. It should be noted that in following the algorithm's steps the resulting sequence is not optimal. Therefore, one concludes that most of the nodes must be branched regardless of the value of the lower bound. Figure 2.4 shows most of the branches in the scheduling tree.

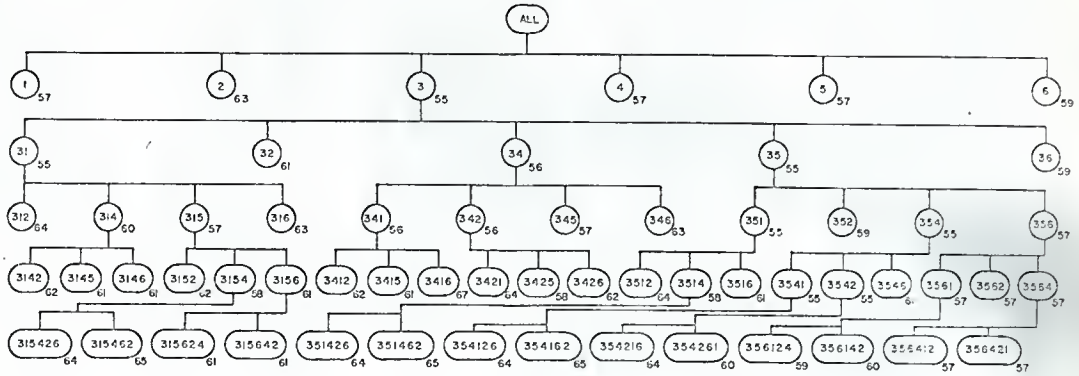

FIGURE 2.4 THE SCHEDULING TREE

$$g_{31}^{2} = 17 + (2 + 11 + 8 + 14) + \min[3\ 7\ 10\ 12] = 55,$$

and

$$g_{31}^{3} = 21 + (3 + 7 + 10 + 12) = 53.$$

For node 32 at level 2, the completion times are:

$$c_{32}^{1} = c_{3}^{1} + t_{21},$$

$$c_{32}^{2} = \max[c_{32}^{1}, c_{3}^{2}] + t_{22},$$

and

$$c_{32}^{3} = \max[c_{32}^{2}, c_{3}^{3}] + t_{23},$$

or

$$c_{32}^{1} = 4 + 12 = 16,$$

$$c_{32}^{2} = \max[16\ 10] + 2 = 18,$$

and

$$c_{32}^{3} = \max[18\ 18] + 3 = 21.$$

The lower bounds are:

$$g_{32}^{1} = 16 + (6 + 3 + 6 + 2) + \min[10\ 18\ 18\ 26] = 43,$$

$$g_{32}^{2} = 18 + (7 + 11 + 8 + 14) + \min[3\ 7\ 10\ 12] = 61,$$

and

$$g_{32}^{3} = 21 + (3 + 7 + 10 + 12) = 53.$$

Proceeding as above for remaining nodes under level 2, the results are summarized in Table 2.2. Here, nodes 31 and 35 both have the same lower bounds of 55. Referring to the scheduling tree, it is observed that branching at the node 31 produced minimum lower

bound of 57 at level 3. This is greater than that of node 31. Hence, the search in this direction is terminated. The remaining results are summarized in Table 2.2.

In solving the above problem, some points have been noted. From the scheduling tree, it is observed that if the algorithm is strictly followed and one concentrates only on minimum lower bounds, optimal solution may not be obtained. For instance, at level 3, node 354 leads to final node 354261. This has schedule time of 60. There is no way to check whether the final solution is optimal or not. Sub-section 2.2 has generated an optimal sequence {3 5 6 4 2 1}. Based on this information, the node 356 was explored. However, this did produce optimal sequences {3 5 6 4 1 2} and {3 5 6 4 2 1}. Hence, one is led to conclude that the algorithm does not guarantee optimality without branching the whole scheduling tree.

This technique is computationally efficient and is competitive with the Extended Direct Technique.

Brown and Lomnicki (20) have conducted computational experiments on ICT 1301 computer. The problems solved, have up to ten jobs and seven machines. They have found that, as the number of machines increases (number of jobs remaining constant), the number of nodes to be explored also increases; the computational effort had to be doubled to obtain all the solutions instead of one; and for the same J with the increasing number of machines, the additional effort in obtaining all the solutions instead of one, should decrease. They have found that the appli-

cation of the algorithm to the reversed order of machines in some cases reduces about 33 percent of computational effort. In such cases the order of jobs in the solution is reversed. This is made possible due to the fact that the scheduling problems are symmetrical with respect to time reversed.

Ignall and Schrage (56) have also developed Branch-and-Bound algorithm similar to that of Lomnicki (74). Their computation experience on CDC 1604 was up to ten jobs and three machines. These authors have introduced concept of "dominated nodes". This concept has reduced the number of nodes to be explored by about 13 percent as no more branching is done from the dominated nodes.

McMahon and Burton (76) have developed reversed approach to some of the flow shop problems by applying Branch and Bound technique. The decision rule is that, reverse the machine order if the total processing time for the first machine is larger than the last machine. The approach of these authors consists of dividing the set of all sequences of jobs into smaller and smaller subsets, and to calculate for each of them a lower bound on the lowest schedule time of all permutations in the set. Their computational experience on CDC 3600 computer involves up to 45 jobs and three machines. They have found that the use of composite bound (machine based bound and job based bound) decision rule is more efficient.

2.4 Lower Bound Technique

Giffler and Thompson (40) have developed an algorithm for generating the subset of feasible sequences. They have defined these sequences as active feasible schedules. These schedules have the following properties: (1) no machine is idle for a length of time sufficient to process an idle job completely; and (2) each operation starts as soon as both, job and the machine are available. In obtaining these schedules they have suggested to resolve randomly, the conflicts among jobs overlapping on a machine.

Brooks and White (19) have modified the above algorithm by introducing the concept of lower bound. Lower bound is defined as the time required to process all jobs on the last machine without conflict. In the modified algorithm the conflict is resolved in favor of the job which produces the least lower bound. White (113) has found that the lower bound concept produces better results than either Monte Carlo, developed by Giffler and Thompson (40), or Shortest Imminent Time (SIT) and Longest Remaining Time (LRT) criteria devised by Fisher and Thompson (33).

The modified algorithm is feasible for job shop problems of J jobs and M machines. The solution is developed in a Table called Work Array. It consists of M blocks. Each block has J columns. A variable X is considered to indicate the time prior to which the schedule is complete and fixed without conflicts; and the time at and after which conflicts may exist

at the corresponding level, L. The algorithm may be summarized as follows.

Step 1: Set L = 1.

Step 2: Enter the completion times of first operation of all jobs in the work array and set X equal to the smallest of these times.

Step 3: Check for conflicts among jobs ending at and after time X within each machine block.

Step 4: Calculate the lower bound for each job in conflict.

Step 5: Find the job(s) which have the least lower bounds.

Step 6: If L = 1, go to step 7, otherwise, check the lower bounds:

6.1 If the least lower bound(s) are higher than that of the previous level in the same machine block, terminate search in this direction. Select another job, from level L-1, which has the second lowest lower bound. Go to step 3.

6.2 If the lower bound(s) are equal to that of the previous level, go to step 7.

Step 7: Resolve the conflict in favor of the job which has the least lower bound. If a tie exists, select a job randomly.

Step 8: For each X, in the array find the next machine, if any, to process the job and enter X plus the processing time for this next operation in the corresponding block.

Step 9: Check X, in the array.

9.1 If X is the largest entry, the corresponding level gives the optimal sequence on each machine.

9.2 If X is not the largest entry, set it equal to the next higher value X' in the array. Go to step 3.

It should be noted that step 6 is modified in this report because lower bounds at lower levels may be equal to or greater than that of the previous levels. In no case the lower bound decreases in lower levels. White (113) has suggested to check lower bounds after conflicts on all machines are cleared. However, this involves unnecessary computations.

In job shop problems, any of the M machines may perform the last operations on the jobs. Therefore, in order to evaluate the lower bound for job j, it is necessary to calculate the completion times on all M machines. The lower bound will be the maximum of the above times.

A sample job shop problem of three jobs and three machines is presented to illustrate the above algorithm. The processing time and machine ordering matrices are as follows:

$$\tau^* = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 4 & 2 \\ 6 & 3 & 5 \end{bmatrix} \qquad M^* = \begin{bmatrix} 12 & 13 & 11 \\ 21 & 22 & 23 \\ 33 & 32 & 31 \end{bmatrix}$$

Note that job 1 is processed first on machine 2, second on machine 3, and finally on machine 1. Job 2 is processed first on machine 1, second on machine 2, and finally on machine

3. Job 3 is processed first on machine 3, second on machine 2 and lastly on machine 1.

According to step 2, completion times 5, 3, and 5 of the first operations of all jobs are entered in appropriate blocks at level 1, as shown in Table 2.3.

Table 2.3

Work Array For a Job Shop Problem of Size (3x3)

Level \ Job	Machine 1			Machine 2			Machine 3		
	1	2	3	1	2	3	1	2	3
1		5		3	9^c	8^c	7^c		5^c
2		5		3	9^c	8^c	9		5
3	11^c	5	14^c	3	12	8	9	14	5
4	16	5	14	3	12	8	9	14	5

^{c} denotes the jobs in conflict.

At level 1, the values of X at machines 1, 2, and 3 are 5, 3, and 5, respectively. Therefore, conflicts do not exist, since the value of X corresponds to different jobs in each machine block.

Following step 8, processing times of next operations of the three jobs are added to X. For job 1, the next operation is on machine 3 and takes 4 units of time. Adding this to the value of X in machine block 2, the value 3+4 or 7 is entered in machine block 3. Similarly, the values 9 and 8 are entered in machine block 2 at level 2.

It is observed that X with a value of 5 is still the largest entry in machine block 1. But X with the value of 3 is not largest entry in block 2. Hence, according to step 9, X is increased to X' with a value of 8; and X which has a value of 5 in block 3 remains without change. It is observed that jobs 1 and 3 are in conflict on machine 3 and jobs 2 and 3 are in conflict on machine 2. Job 3 is in conflict for its first operation on machine 3 and for its second operation on machine 2. Therefore, the conflict will be resolved first on machine 3.

Resolving conflict in favor of job 1, the completion time of all operations on machine 1 is

$$t_{12} + t_{13} + t_{11} + t_{31} + t_{32} + t_{33}$$

or

$$3 + 4 + 2 + 6 + 3 + 5 = 23,$$

on machine 2

$$t_{12} + t_{33} + t_{32} + t_{22}$$

or

$$3 + 5 + 3 + 4 = 15,$$

and on machine 3

$$t_{12} + t_{13} + t_{33} + t_{23}$$

or

$$3 + 4 + 5 + 2 = 14.$$

Thus, the lower bound is

$$\max[23\ 15\ 14] = 23.$$

By resolving conflict in favor of job 3, the completion time of all operations on machine 1 is

$$t_{32} + t_{33} + t_{31} + t_{11}$$

or

$$3 + 5 + 6 + 2 = 16,$$

on machine 2

$$t_{33} + t_{32} + t_{22}$$

or

$$5 + 3 + 4 = 12,$$

and on machine 3

$$t_{22} + t_{33} + t_{13} + t_{23}$$

or

$$4 + 5 + 4 + 2 = 15.$$

Thus, the lower bound is

$$\max[16\ 12\ 15] = 16.$$

As the lower bound is less for job 3, conflict is resolved in favor of this job. The completion time of job 1 on machine 3 then becomes 5+4 or 9. This value is entered in level 2.

Next, conflicts are resolved for machine 2 at level 2. By resolving the conflict in favor of job 2, completion time of all operations on machine 1 is

$$t_{21} + t_{22} + t_{32} + t_{31}$$

or

$$5 + 4 + 3 + 6 = 18,$$

on machine 2

$$t_{21} + t_{22} + t_{32}$$

or

$$5 + 4 + 3 = 12,$$

and on machine 3

$$t_{33} + t_{13} + t_{23}$$

or

$$5 + 4 + 2 = 11.$$

Thus, the lower bound is

$$\max[18\ 12\ 11] = 18.$$

By resolving conflict in favor of job 3, completion time of all operations on machine 1 is

$$t_{33} + t_{13} + t_{11} + t_{31}$$

or

$$5 + 4 + 2 + 6 = 17,$$

on machine 2

$$t_{33} + t_{32} + t_{22}$$

or

$$5 + 3 + 4 = 12,$$

and on machine 3

$$t_{33} + t_{32} + t_{22} + t_{23}$$

or

$$5 + 3 + 4 + 2 = 14.$$

Thus, the lower bound is

$$\max[17\ 12\ 14] = 17$$

The lower bound is less for job 3. Hence, conflict is resolved in favor of this job. Completion time of job 2 on machine 2 is 8+4 or 12. This value is entered in level 3. The value of X is 12 in machine block 2. As this is the largest entry, all conflicts are cleared for operation of all jobs on machine 2.

Following step 8, the next operation of job 3 is on machine 1. Hence, 8+6 or 14 is entered in machine block 1. Similarly for job 1, the value 9+2 or 11 is entered in machine block 1. These two jobs are in conflict.

Repeating step 4, lower bound for job 1 is 17 and for job 3 is 16. Therefore, conflict should be resolved in favor of job 3. The completion time of job 1 becomes 14+2 or 16. From Table 2.3 the value 16 is the largest entry in the array at level 4. This level then gives the optimal sequence on each machine.

In flow shop problems, all jobs have the same machine ordering. Therefore, a sequence of jobs which is optimal for machine 1, will also be optimal for all other machines. Moreover, the last machine is also the same for all jobs. Hence, it is not necessary to resolve conflicts of jobs on all machines. Second, an expression for lower bound can be derived as the last machine is known. Based on this reasoning, the above algorithm is modified which reduces the computations and may be easily programed on computer.

The following notation is considered.

L level of the scheduling tree.

n node consisting of a partial sequence of jobs $\{j_1, j_2, \ldots, j_L\}$.

C_n column vector where its elements represent completion time of the jobs on machine M-1 for node n.

A column vector where its elements are the same as those in vector C_n arranged in ascending order.

B column vector where its elements represent the corresponding processing times of jobs on machine M to the elements of vector A.

The modified algorithm may be summarized as follows:

Step 1: Let L = 1.

Step 2: For each node n, where n consists of a partial sequence $\{j_1, j_2, \ldots, j_L\}$ compute C_n such that

$$C_n = \sum_{m=1}^{M-1} t_{jm} \qquad \text{for } j = j_1 ,$$

$$= t_{j_1 1} + \sum_{m=1}^{M-1} t_{jm} \qquad \text{for } j = j_2 ,$$

$$\vdots$$

$$= \sum_{x=1}^{L-1} t_{j_x 1} + \sum_{m=1}^{M-1} t_{jm} \qquad \text{for } j = j_L ,$$

$$= \sum_{x=1}^{L} t_{j_x 1} + \sum_{m=1}^{M-1} t_{jm} \qquad \text{for } j \neq j_1, j_2, \ldots, j_L .$$

Step 3: Arrange the elements of C_n in ascending order and call them $A_1, A_2, \ldots, A_J$.

Step 4: Calculate the lower bound $G^{L,n}$ for each node such that

$$G^{L,n} = D_J$$

where

$$D_J = \max[D_{J-1}, A_J] + B_J$$

$$D_{J-1} = \max[D_{J-2}, A_{J-1}] + B_{J-1}$$

$$\vdots$$

$$D_2 = \max[D_1, A_2] + B_2$$

$$D_1 = A_1 + B_1$$

Step 5: Find the job(s) which have the least lower bounds, G^L such that

$$G^L = \min_n [G^{L,n}]$$

Step 6: If L = 1, go to step 7, otherwise, check the lower bounds:

6.1 If the least lower bound(s) are higher than that of the previous level, terminate search in this direction. Select another job which has the second least lower bound. Go to step 2.

6.2 If the lower bound(s) are equal to that of the previous level, go to step 7.

Step 7: Resolve the conflict in favor of the job(s) which have the least lower bound(s).

Step 8: Check L:

8.1 If $L \leq J-1$, set L = L+1 and go to step 2.

8.2 If $L > J-1$ go to step 9.

Step 9: Select the node(s) having the least lower bounds. Determine the schedule time of all nodes and select the one which gives the minimum schedule time.

It should be pointed out that step 5 is not always true, however it is better to branch all nodes regardless of the value of the lower bound, in order to obtain the optimal solution.

The sample flow shop problem of six jobs and three machines presented in subsection 2.3, is solved to illustrate this technique. The processing time and machine ordering matrices are reproduced for convenience.

$$\tau^* = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \\ 4 & 6 & 8 \\ 3 & 11 & 7 \\ 6 & 8 & 10 \\ 2 & 14 & 12 \end{bmatrix} \qquad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix}$$

According to step 2, at level 1, the number of jobs in each node is 1. Then

$$C_1 = \sum_{m=1}^{M-1} t_{jm}, \qquad \text{for } j = 1,$$

and

$$C_1 = t_{11} + \sum_{m=1}^{M-1} t_{jm} \qquad \text{for } j = 2, 3, \ldots, 6,$$

or

$$C_1 = 13 = 13, \qquad j = 1,$$

$$= 6 + 14 = 20, \qquad j = 2,$$

$$= 6 + 10 = 16, \qquad j = 3,$$

$$= 6 + 14 = 20, \qquad j = 4,$$

$= 6 + 14 = 20, \qquad j = 5,$

$= 6 + 16 = 22, \qquad j = 6.$

Arranging the C_1's in ascending order according to step 3,

$$A = \begin{bmatrix} 13 \\ 16 \\ 20 \\ 20 \\ 20 \\ 22 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 3 \\ 8 \\ 3 \\ 7 \\ 10 \\ 12 \end{bmatrix} .$$

To calculate the lower bound,

$$D_1 = A_1 + B_1 ,$$

$$D_2 = \max[D_1, A_2] + B_2 ,$$

$$\vdots \qquad \vdots$$

$$D_6 = \max[D_5, A_6] + B_6,$$

or

$$D_1 = 13 + 3 = 16,$$

$$D_2 = \max[16\ 16] + 8 = 24,$$

$$D_3 = \max[24\ 20] + 3 = 27,$$

$$\vdots \qquad \vdots$$

$$D_6 = \max[44\ 22] + 12 = 56.$$

Thus, the lower bound is

$$G^{1,1} = D_6$$

$$= 56.$$

Next, the computation for node 2 at level 1 is

$$C_2 = \sum_{m=1}^{M-1} t_{jm}, \qquad \text{for } j = 2$$

$$= t_{21} + \sum_{m=1}^{M-1} t_{jm}, \qquad j = 1, 3, \ldots, 6$$

or

$$C_2 = 12 + 13 = 25, \qquad j = 1,$$

$$= 14 \quad = 14, \qquad j = 2,$$

$$= 12 + 10 = 22, \qquad j = 3,$$

$$= 12 + 14 = 26, \qquad j = 4,$$

$$= 12 + 14 = 26, \qquad j = 5,$$

$$= 12 + 16 = 28, \qquad j = 6.$$

Arranging the above C_2's in ascending order

$$A = \begin{bmatrix} 14 \\ 22 \\ 25 \\ 26 \\ 26 \\ 28 \end{bmatrix} \qquad \text{and} \qquad B = \begin{bmatrix} 3 \\ 8 \\ 3 \\ 7 \\ 10 \\ 12 \end{bmatrix}$$

Then,

$$D_1 = 14 + 3 \quad = 17,$$

$$D_2 = \max[17\ 22] + 8 \quad = 30,$$

$$\vdots \qquad \vdots$$

$$D_6 = \max[50\ 28] + 12 = 62.$$

Thus, the lower bound is

$$G^{1,2} = 62.$$

Repeating the above procedure, lower bounds for the nodes 3, 4, 5, and 6 at level 1 are 53, 56, 59, and 55 respectively. The least lower bound is that of node 3. Hence, the conflict is resolved in favor of node 3, i.e., job 3.

For level 2, compute the lower bounds for all nodes encountered such that

$$C_{31} = \sum_{m=1}^{M-1} t_{jm}, \qquad j = 3$$

$$= t_{31} + \sum_{m=1}^{M-1} t_{jm}, \qquad j = 1$$

$$= t_{31} + t_{11} + \sum_{m=1}^{M-1} t_{jm}, \quad j = 2, 4, 5, 6$$

or

$$C_{31} = 4 + 13 = 17, \qquad j = 1,$$

$$= 4 + 6 + 14 = 24, \qquad j = 2,$$

$$= 10 = 10, \qquad j = 3,$$

$$= 4 + 6 + 14 = 24, \qquad j = 4,$$

$$= 4 + 6 + 14 = 24, \qquad j = 5,$$

and

$$= 4 + 6 + 16 = 26, \qquad j = 6.$$

Arranging the C_{31}'s in ascending order,

$$A = \begin{bmatrix} 10 \\ 17 \\ 24 \\ 24 \\ 24 \\ 26 \end{bmatrix} \qquad B = \begin{bmatrix} 8 \\ 3 \\ 3 \\ 7 \\ 10 \\ 12 \end{bmatrix}$$

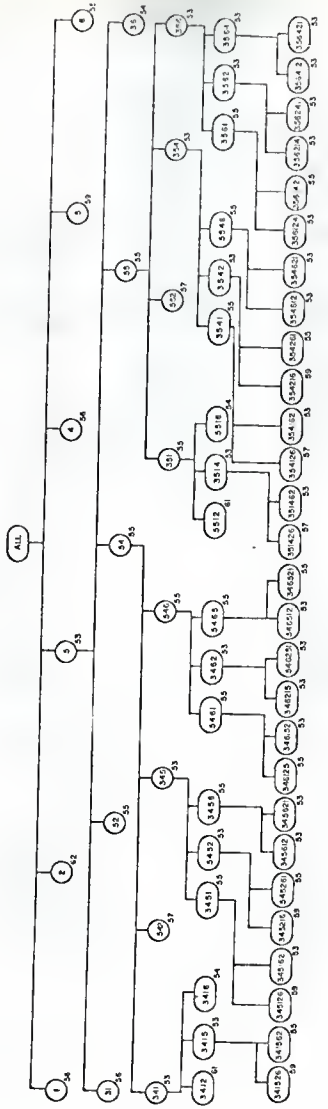

FIGURE 2.5 THE SCHEDULING TREE

Then,

$$D_1 = 10 + 8 = 18,$$

$$D_2 = \max[18\ 17] + 3 = 21,$$

$$\vdots \qquad \vdots$$

$$D_6 = \max[44\ 26] + 12 = 56.$$

Thus, the lower bound is

$$G^{2,31} = 56.$$

Repeating the above procedure, lower bounds for the nodes 32, 34, 35, and 36 are 61, 53, 53, and 54, respectively. According to step 7, branching should be done at two nodes 34 and 35 because each has the least lower bound of 53. Returning to step 2 and repeating the same procedure, it is noted that a set of 20 sequences are generated by the algorithm. These sequences are evaluated to obtain the corresponding schedule times. The sequences with their schedule times are shown in Table 2.4. Note that sequences 11, 12 and 17 yield the minimum schedule time which is 57. Therefore, the sequences are optimal.

Table 2.4

Table of Feasible Sequences

Sequence Number	Sequence	Schedule Time
1	3 4 1 5 6 2	65
2	3 4 6 1 6 2	65
3	3 4 5 6 1 2	61
4	3 4 5 6 2 1	61
5	3 4 6 1 2 5	63
6	3 4 6 2 5 1	63
7	3 4 6 1 5 2	63
8	3 4 6 5 1 2	63
9	3 4 6 5 2 1	63
10	3 4 6 2 1 5	63
11	3 5 6 4 2 1	57
12	3 5 6 4 1 2	57
13	3 5 4 6 2 1	61
14	3 5 6 1 4 2	60
15	3 5 1 4 6 2	65
16	3 5 6 1 2 4	59
17	3 5 6 2 4 1	57
18	3 5 6 2 1 4	59
19	3 5 4 6 1 2	61
20	3 5 4 1 6 2	65

2.5 Boolean Algebra Technique

A non-numerical approach to the job shop scheduling problem has been devised by Akers and Friedman (5) employing methods of Boolean Algebra. The job shop is characterized by a different machine ordering for each job. The sequences which do not follow the specified machine ordering are called the technologically non-feasible sequences. The non-numeric technique does not require to specify the processing times of jobs on the machine, to arrive at the feasible sequences. However, these feasible sequences must be evaluated in order to find the optimal sequence(s). This technique is feasible for the problem of two jobs and M machines.

The following notation is considered to present the technique:

m = the decision that job 1 is processed first on the machine m, $m = 1, 2, \ldots, M$,

$\bar{m}$ = the decision that job 2 is performed first on machine m, $m = 1, 2, \ldots, M$.

The technique consists of eliminating non-feasible and non-optimal sequences. This is done by applying certain decision rules formulated by Akers and Friedman (5). The decision rules are based on two theorems which are stated below without proof. For their proof, reference is made to the paper of Akers and Friedman (5).

Theorem 1: A necessary and sufficient condition that a sequence be feasible is that for the two machines, m and n, where m precedes n for job 1 and n precedes m for job 2, then, the sequence which includes the term $\bar{m}n$ is non-feasible.

Theorem 2: A necessary and sufficient condition that a feasible sequence belong to the set of optimal sequences is that it contains no free machines.

By the term free machine, it is meant that a machine is idle for a length of time, sufficient to process a job completely. The decision rules are given in Table 2.5.

Table 2.5

Table of Decision Rules

Rule No.	Machine Orderings Job 1	Job 2	Delete sequences containing
1	... m	m ...	$\bar{m}$
2	m ...	... m	$\bar{m}$
3	m ... n	n ...	$m\bar{n}$
4	m ... n ...	... m n ...	$\bar{m}n$
5	... m ... n	... m n ...	$\bar{m}n$
6	... m n ...	m ... n ...	$\bar{m}\bar{n}$
7	... m n ... k ...	... m ... n k ...	$\bar{m}\bar{n}k$
8	... m ... n k ...	... m n ... k ...	$m\bar{n}\bar{k}$

The approach is illustrated by solving a sample job shop problem of two jobs, to be processed on three machines. The processing time and machine ordering matrices of the sample problem are presented below:

$$T^* = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \end{bmatrix} \quad , \quad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 23 & 22 & 21 \end{bmatrix}$$

Note that job 1 is to be processed on machine 1 first, machine 2 second, and machine 3 last. Job 2 is to be performed by machine 3 first, machine 2 second, and machine 1 last.

The problem is solved in two steps. The first step is to eliminate non-feasible sequences and the second step is to eliminate the non-optimal sequences. The total number of possible sequences is $(J!)^M$ or $(2!)^3 = 8$. In order to eliminate non-feasible sequences, all the eight sequences are generated and represented as follows.

Sequence Number

1	2	3	4	5	6	7	8
$\overline{1}$	1	$\overline{1}$	1	$\overline{1}$	1	$\overline{1}$	1
$\overline{2}$	$\overline{2}$	2	2	$\overline{2}$	$\overline{2}$	2	2
$\overline{3}$	$\overline{3}$	$\overline{3}$	$\overline{3}$	3	3	3	3

A systematic procedure to fill the above table is to fill the first 2^{k-1} spaces on row k with barred numbers, and completing the row with alternate blocks of 2^{k-1} numbers without and with bars. For example row 2 starts with 2^{2-1} or two barred numbers, $\overline{2}\ \overline{2}$, followed by two numbers without bars, 2 2, and so forth.

In the present example, note that first, machine 1 precedes machine 3 for job 1 and machine 3 precedes machine 1 for job 2. Second, machine 2 precedes machine 3 for job 1 and machine 3 precedes machine 2 for job 2. Finally, machine 1 precedes

machine 2 for job 1 and machine 2 precedes machine 1 for job 2. Hence, applying theorem 1 it is observed that the sequences including the terms $\overline{1}$ 3, $\overline{2}$ 3, and $\overline{1}$ 2 are non-feasible. Thus, the sequences 3, 5, 6, and 7 are eliminated and the remaining four sequences 1, 2, 4, and 8 are feasible.

The second step is to eliminate those feasible sequences that cannot be optimal. These sequences are eliminated by the decision rules obtained from theorem 2 and described in Table 2.5. The remaining set of feasible sequences is such that; for any assignment of processing times the optimal sequence is included in the set and every sequence in the set is optimal for some assignment of processing times.

Referring to Table 2.5 it is observed that rule 3 is applicable in present case. Note that the jobs 1 and 2 are processed on machines 1, 2, 3 and 3, 2, 1 respectively. Therefore, the sequence containing the term 13 cannot be optimal. The sequences 8 and 1 are thus climinated. The remaining two sequences are 2 and 4.

These two sequences are feasible as well as each can be optimal for some processing times. For the sample problem, Gantt charts are drawn to determine the optimal sequence. The schedule time for the sequence 2 from the Figure 2.6 is 18 and that of sequence 4 from the Figure 2.7 is 27. Hence, the sequence 2, {1 $\overline{2}$ $\overline{3}$} is optimal. It indicates that job 2 has to be processed first on machines 2 and 3; and job 1 has to be processed first on machine 1.

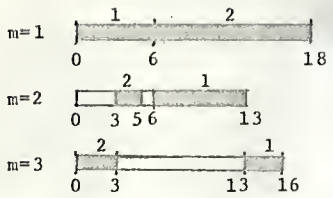

Figure 2.6 Gantt Chart for a Job Shop Problem of Size (2x3)

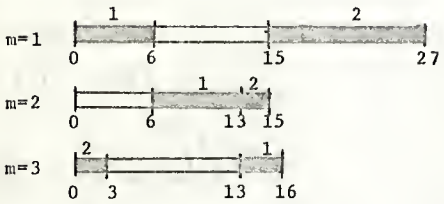

Figure 2.7 Gantt Chart for a Job Shop Problem of Size (2x3)

It is observed that this is a very efficient method for solving job shop problems consisting of two jobs and M machines.

CHAPTER III

INTEGER-LINEAR PROGRAMING APPROACH

The machine scheduling problem has been formulated as integer-linear programing problem. Some computational experience has been gained by using the integer-linear programing algorithm of Gomory (43). Presently, there are three published formulations to the problem, those of Bowman (18), Wagner (111), and Manne (75). Manne's formulation is the most compact; however, none of the authors claim practicality of their formulations.

Neglecting the integer constraints, Dantzig (27) and (28) has formulated this problem as an ordinary linear programing problem. The weakness of this model is that it may lead to a fractional optimal solution. The solution may call for non-integer number of jobs to be processed on the machines. One way to overcome this, would be to round off the fractional solution. Following this procedure, Giglio and Wagner (42) have solved 100 flow shop problems of six jobs and three machines. They have found that the results are not encouraging.

It is evident that any job cannot be scheduled to start on any machine until its predecessors have finished on that machine. Thus, a variable which is set equal to 0 or 1, depending on whether the job is scheduled or not, is the basic idea in the linear programing formulations. The three formulations are presented below:

3.1 Bowman's Formulation

This formulation is feasible for the job shop problem of J jobs and M machines. It requires an estimate of the schedule time, T, of the problem. Thus, the number of constraints and variables is a function of T.

In constructing the constraints, the basic variables in the formulation are of the form $[j_x m_y]_t$ which indicates that the operation $j_x m_y$ is taking place during the unit time period t ($t = 1, 2, \ldots, T$). All variables will have a value either 0 or 1 in the solution, i.e., the operation has or has not occured during the unit time period t. These constraints are such that

$$0 \leq [j_x m_y]_t \leq 1, \qquad \begin{matrix} x = 1, 2, \ldots, J, \\ y = 1, 2, \ldots, M, \\ t = 1, 2, \ldots, T. \end{matrix} \qquad (3.1)$$

Since all individual operations must be performed, the constraints

$$\sum_{t=1}^{T} [j_x m_y]_t = t_{jm}, \qquad \begin{matrix} j,x = 1, 2, \ldots, J, \\ m,y = 1, 2, \ldots, M \end{matrix} \qquad (3.2)$$

are constructed. To ensure that two or more jobs are not processed by the same machine at the same time, i.e., to avoid overlapping, the constraints

$$\sum_{x=1}^{J} [j_x m_y]_t \leq 1, \qquad \begin{matrix} y = 1, 2, \ldots, M, \\ t = 1, 2, \ldots, T \end{matrix} \qquad (3.3)$$

hold.

Some other constraints are constructed to ensure proper sequencing of the jobs. No operation may not take place until the previous operation on the same job in the specified sequence has been completed in a previous unit time period. The constraints are:

$$t_{jm}[j_x m_y]_t \leq \sum_{i=1}^{t-1} [j_x m_y]_i, \qquad \begin{matrix} j,x = 1, 2, \ldots, J, \\ m,y = 1, 2, \ldots, M, \\ t = 1, 2, \ldots, T. \end{matrix} \tag{3.4}$$

According to the above constraints, any operation may be interrupted. Therefore, the following set of constraints are required to eliminate such possibility:

$$t_{jm}[j_x m_y]_t - t_{jm}[j_x m_y]_{t+1} + \sum_{i=t+2}^{T} [j_x m_y]_i \leq t_{jm}, \tag{3.5}$$

$$\begin{matrix} j,x = 1, 2, \ldots, J, \\ m,y = 1, 2, \ldots, M, \\ t = 1, 2, \ldots, T. \end{matrix}$$

This does not allow a variable with value 1, to be followed by a variable with a value 0, and yet be followed by variables, each of which has value of 1.

The objective function is constructed such that a sequence for which the schedule time is minimum, is obtained. In other words, the objective is to have final operations on all jobs performed as early as possible. Thus, the objective function is expressed as

Minimize:

$$Z = \sum_{i=1}^{T-F} (M)^{i-1} \{ [j_1 m_M]_{F+i} + [j_2 m_{M-1}]_{F+i} + \ldots + [j_x m_y]_{F+i} + \ldots + [j_J m_1]_{F+i} \} \tag{3.6}$$

where

F = maximum of, the total processing times of the J jobs on all machines,

$$= \max\left[\sum_{m=1}^{M} t_{1m}, \sum_{m=1}^{M} t_{2m}, \ldots, \sum_{m=1}^{M} t_{jm}, \ldots, \sum_{m=1}^{M} t_{Jm} \right].$$

The rationale of the objective function is that it makes operations (the last ones on each job) towards the end of the time periods, costly. The number of unit time periods estimated in advance of solution, may certainly be equal to or less than the total processing times of all jobs on all machines, i.e., $\sum_{j=1}^{J} \sum_{m=1}^{M} t_{jm}$, and cannot be less than F. Bowman, however does not recommend how to estimate T, whether it should be near to F or near to $\sum_{j=1}^{J} \sum_{m=1}^{M} t_{jm}$. Selection of T makes a significant difference in the number of constraints and variables involved in solution. Hence, for estimation of T, the formula developed by Heller (49) for flow shop problems is presented.

$$T \geq \max_{m} \sum_{j'=1}^{J} t_{j'm} + \min_{j} \sum_{m'=1}^{m-1} t_{jm'} + \min_{\hat{j} \neq j} \sum_{m'=m+1}^{M} t_{\hat{j}m'} \qquad (3.7)$$

where m in the second and third sum is that m giving the maximum in the first sum, and j in the third term is that j giving the minimum in the second sum. The above formula states that the optimal schedule time cannot be shorter than the total processing time for all jobs on machine m plus the shortest time of processing, say, job j, on machines 1, 2, ..., m-1 plus the shortest time of processing, say, job $\hat{j}$ on machines m+1, m+2, ..., M, where clearly the job j must be different from the job $\hat{j}$.

The cost associated with any operation in a time period is a synthetic one equal to the sum of all prior costs plus one. This exploding cost function thus forces operations toward the beginning for economic reasons. No later time period will be ultimately used than the minimum (optimal), as this one cost is

larger than the sum of all prior costs. That is, given some feasible solution, the latest operation would be moved earlier by one time period, and all other operations could be moved later by any number of time periods and the exchange would be favorable.

The formulation involves a large number of variables, depending on the estimated schedule time, T, [the number equals (jobs) x (machines) x (time periods)]. The number of constraints is substantially larger than this.

The sample problem of six jobs and three machines, solved in Chapter II, will be formulated. The processing time and machine ordering matrices are:

$$T^* = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \\ 4 & 6 & 3 \\ 3 & 11 & 7 \\ 6 & 8 & 10 \\ 2 & 14 & 12 \end{bmatrix} \qquad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix}$$

The estimate of T is evaluated according to Heller (49) as this is a flow shop problem.

$$T \geq \max_m \sum_{j'=1}^{J} t_{j'm} + \min_j \sum_{m'=1}^{m-1} t_{jm'} + \min_{\hat{j}\neq j} \sum_{m'=m+1}^{M} t_{\hat{j}m'}$$

$$\geq 48 + 2 + 3$$

$$\geq 53 .$$

Note that F is the maximum of the total processing times of the J jobs on all machines. Thus,

$$F = \max[16\ 17\ 18\ 21\ 24\ 28] ,$$

$$= 28.$$

The problem is formulated as follows:

Minimize

$$Z = 1\{[j_1m_3]_{29}+[j_2m_3]_{29}+[j_3m_2]_{29}+[j_4m_2]_{29}+[j_5m_1]_{29}+[j_6m_1]_{29}\}+$$

$$3\{[j_1m_3]_{30}+[j_2m_3]_{30}+[j_3m_2]_{30}+[j_4m_2]_{30}+[j_5m_1]_{30}+[j_6m_1]_{30}\}+$$

$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots +$$

$$3^{24}\{[j_1m_3]_{53}+[j_2m_3]_{53}+[j_3m_2]_{53}+[j_4m_2]_{53}+[j_5m_1]_{53}+[j_6m_1]_{53}\}$$

(3.8)

Subject to

$$0 \leq [j_xm_y]_t \leq 1 \quad , \qquad \begin{matrix} x = 1, 2, \ldots, 6, \\ y = 1, 2, 3, \\ t = 1, 2, \ldots, 53. \end{matrix} \tag{3.9}$$

$$\sum_{t=1}^{53} [j_xm_y]_t = t_{jm} \quad , \qquad \begin{matrix} j,x = 1, 2, \ldots, 6, \\ m,y = 1, 2, 3. \end{matrix} \tag{3.10}$$

$$\sum_{x=1}^{6} [j_xm_y]_t \leq 1 \quad , \qquad \begin{matrix} y = 1, 2, 3, \\ t = 1, 2, \ldots, 53. \end{matrix} \tag{3.11}$$

$$t_{jm}[j_xm_{y+1}]_t \leq \sum_{i=1}^{t-1} [j_xm_y]_t \quad , \qquad \begin{matrix} j,x = 1, 2, \ldots, 6, \\ m,y = 1, 2, 3, \\ t = 1, 2, \ldots, 53. \end{matrix} \tag{3.12}$$

$$t_{jm}[j_xm_y]_t - t_{jm}[j_xm_y]_{t+1} + \sum_{i=t+2}^{53} [j_xm_y]_i \leq t_{jm} \quad , \qquad \begin{matrix} j,x = 1, 2, \ldots, 6 \\ m,y = 1, 2, 3 \\ t = 1, 2, \ldots, 53. \end{matrix} \tag{3.13}$$

The number of constraints are shown below:

Equation Number	Number of Constraints
(3.9)	6x3x53 = 954
(3.10)	6x3 = 18
(3.11)	3x53 = 159
(3.12)	6x3x53 = 954
(3.13)	6x3x53 = 954

As a result, the total number of constraints is (954 + 18 + 159 + 954 + 954) or 3039. The total number of variables is 6x3x53 or 954. It is evident that the formulation involving such a large number of constraints and variables is not practical. Further, it will be observed that expressing the processing times in hours, instead of in minutes (if possible) will make a significant difference in the total number of constraints and variables encountered.

3.2 Wagner's Formulation

The formulation is based on defining a variable which takes value one or zero depending on whether a job is or is not scheduled in a specific sequence-position on a machine. In order to present Wagner's formulation (111) of the machine scheduling problem, the following notation is considered:

$$X_{j_x m_y} = \begin{cases} 1 \text{ if job } j \text{ is scheduled in sequence-position } x \text{ on machine } m_y \\ 0 \text{ otherwise} \end{cases}$$

$S_{j_x m_y}$ = starting time of job j in sequence-position x on machine m_y.

$I_{j_x m_y}$ idle time after finishing job j_x on machine m_y. This is the difference between the finishing time of job j_x and the starting time of job j_{x+1} on the machine m_y.

$D_{j_x m_y}$ idle time after finishing job j_x on machine m_y and starting of job j_x on machine m_{y+1}. This is the difference between the finishing time of job j_x on machine m_y and the starting time of job j_x on the machine m_{y+1}.

$t_{j_x m_y}$ processing time of job j_x on machine m_y.

The formulation of the job shop problem is presented by developing the following constraints:

To ensure that all jobs are processed on all machines, the constraints

$$\sum_{x=1}^{J} X_{j_x m_y} = 1 \quad , \qquad \begin{matrix} j = 1, 2, \ldots, J, \\ m_y = 1, 2, \ldots, M \end{matrix} \tag{3.14}$$

hold.

A second set of constraints is required to ensure that not more than one job is assigned to the sequenee-position x on the machine m_y. Thus, the set of equations

$$\sum_{j=1}^{J} X_{j_x m_y} = 1 \quad , \qquad \begin{matrix} x = 1, 2, \ldots, J, \\ m_y = 1, 2, \ldots, M \end{matrix} \tag{3.15}$$

holds.

The operation preeedence constraints are developed so that all jobs are not started on next machines until the previous

operations are finished. The constraints are

$$S_{j_x m_y} + t_{j_x m_y} X_{j_x m_y} \leq S_{j_x m_{y+1}} + C(1-X_{j_x m_y}) + C(1-X_{j_x m_{y+1}}), \quad (3.16)$$

$$j_x = 1, 2, \ldots, J,$$
$$m_y = 1, 2, \ldots, M$$

where C is a large positive integer.

To ensure that two or more jobs are not processed by the same machine at the same time, i.e., to avoid overlapping, the constraints

$$S_{j_x m_y} + t_{j_x m_y} X_{j_x m_y} \leq S_{j_{x+1} m_y} + C(1-X_{j_x m_y}) + C(1-X_{j_{x+1} m_y}) \quad (3.17)$$

$$j_x = 1, 2, \ldots, J$$
$$m_y = 1, 2, \ldots, M$$

hold.

In job shop problem, it is not possible to predict which of the machines will perform the last operation. As in the previous formulation, an estimate of the schedule time is required. Let this estimate be T. Then, the following constraints allow for the possibility of any machine being the last in operation.

$$S_{Jm} + t_{Jm} \leq T \quad , \qquad m = 1, 2, \ldots, M \qquad (3.18)$$

The objective is to minimize T. However, Wagner (111) does not present an explicit expression for the objective function. The total number of constraints and variables required for the job shop problem of J jobs and M machines may be summarized as follows:

Equation Number	Number of Constraints	Number of Variables
3.14	JM	$(JM)^2 + 4JM + M$
3.15	JM	
3.16	JM	
3.17	JM	
3.18	M	

The formulation of the flow shop problem involves less number of constraints and variables. This is because all jobs have the same machine ordering. Hence, the set of constraints of the form (3.16) through (3.18) are not required. In order to ensure the conditions: (1) a job may not be processed on more than one machine at a time; and (2) a machine may not perform more than one job at a time, a set of constraints may be developed. Condition (1) may be expressed as

$$S_{j_x m_y} = S_{j_{x-1} m_y} + t_{j_{x-1} m_y} + I_{j_{x-1} m_y} \tag{3.19}$$

and, the condition (2) may be expressed as

$$S_{j_x m_y} = S_{j_x m_{y-1}} + t_{j_x m_{y-1}} + D_{j_x m_{y-1}}. \tag{3.20}$$

Subtracting (3.20) from (3.19), the following equation is obtained.

$$0 = S_{j_{x-1} m_y} + t_{j_{x-1} m_y} + I_{j_{x-1} m_y} - S_{j_x m_{y-1}} - t_{j_x m_{y-1}} - D_{j_x m_{y-1}} \tag{3.21}$$

Equation (3.21) expresses a specific operation. Hence, the operation which follows on the same machine may be expressed as

$$0 = S_{j_x m_y} + t_{j_x m_y} + I_{j_x m_y} - S_{j_{x+1} m_{y-1}} - t_{j_{x+1} m_{y-1}} - D_{j_{x+1} m_{y-1}} \tag{3.22}$$

Subtraeting (3.21) from (3.22) and utilizing (3.19), the following equation is obtained.

$$0 = t_{j_x m_y} - t_{j_{x+1} m_{y-1}} + I_{j_x m_y} - I_{j_x m_{y-1}} + D_{j_x m_{y-1}} - D_{j_{x+1} m_{y-1}}, \tag{3.23}$$

$$j_{x+1} = 2, 3, \ldots, J,$$
$$m_y = 2, 3, \ldots, M.$$

Note that (3.23) forms a single set of constraints to meet both eonditions (1) and (2). The starting times in (3.21) and (3.22) are eonverted into proeessing times in (3.23).

The objective is to minimize the sehedule time. This means that the objeetive funetion may be expressed as minimizing the idle time on the last maehine. For this, all jobs should be started as early as possible on the last maehine. Thus, the objeetive funetion is,

Minimize

$$Z = \sum_{j=1}^{J} [t_{jM} \sum_{x=1}^{J} X_{j_x M}] + \sum_{x=0}^{J-1} I_{j_x M} \tag{3.24}$$

Wagner (111) ineludes the sum of the idle times in the objeetive funetion; however, it is not required beeause the idle time variables are slaek ones.

The number of constraints and variables required for flow shop problem of J jobs and M machines are shown below:

Equation Number	Number of Constraints	Number of Variables
3.14	JM	$(JM)^2 + 2(M-1)(J-1)$
3.15	JM	
3.16	$(M-1)(J-1)$	

An interesting case of the above formulation is the three-machine problem. It has been shown by Johnson (63) that for $M \leq 3$, optimality of the schedule is not lost if all jobs are assumed to have the same machine ordering. Hence, in formulation of this problem, the number of variables and constraints, are reduced considerably.

The following notation is considered to present the formulation.

$$X_{j_x} = \begin{cases} 1 \text{ if job } j \text{ is scheduled in sequence-position } x, \\ 0 \text{ otherwise.} \end{cases}$$

$\overline{X}_{j_x}$ $\quad [X_{1_x}, X_{2_x}, \ldots, X_{j_x}, \ldots, X_{J_x}]$

Q_1 row vector of processing times for jobs 1 through J on machine 1,

Q_2 row vector of processing times for jobs 1 through J on machine 2,

Q_3 row vector of processing times for jobs 1 through J on machine 3.

The constraints, to ensure that all jobs are processed on all machines, and not more than one job is assigned the sequence position x, may be expressed as the standard assignment problem. Thus,

$$\sum_{x=1}^{J} X_{j_x} = 1 \qquad j = 1, 2, \ldots, J, \qquad (3.25)$$

and

$$\sum_{j=1}^{J} X_{j_x} = 1 \qquad x = 1, 2, \ldots, J. \qquad (3.26)$$

hold.

The timing restrictions among machines 1 and 2 may be expressed as

$$[Q_2][\bar{X}_{j_x}] - [Q_1][\bar{X}_{j_{x+1}}] \leq 0, \qquad x = 1, 2, \ldots, J-1 \qquad (3.27)$$

$$[Q_3][\bar{X}_{j_x}] - [Q_2][\bar{X}_{j_{x+1}}] \leq 0, \qquad x = 1, 2, \ldots, J-1 \qquad (3.28)$$

The objective is to start job in sequence-position 1, on machines 1 and 2, as early as possible. Remaining jobs follow this job. Therefore, the objective function is

Minimize

$$Z = [Q_1 + Q_2][\bar{X}_{j_1}] . \qquad (3.29)$$

The number of constraints and variables required for flow shop problem of J jobs and three machines may be summarized as below.

Equation Number	Number of Constraints	Number of Variables
3.25	J	$J^2 + 4(J-1)$
3.26	J	
3.27	J-1	
3.28	J-1	

The same sample problem of six jobs and three machines is formulated according to Wagner. The objective function and constraints are:

Minimize

$$Z = 13X_{1_1} + 14X_{2_1} + 10X_{3_1} + 14X_{4_1} + 14X_{5_1} + 16X_{6_1} \tag{3.30}$$

subject to

$$\sum_{j=1}^{6} X_{j_x} = 1, \qquad x = 1, 2, \ldots, 6 \tag{3.31}$$

$$\sum_{x=1}^{6} X_{j_x} = 1, \qquad j = 1, 2, \ldots, 6 \tag{3.32}$$

$$\begin{aligned}
&3X_{1_1} + 3X_{2_1} + 8X_{3_1} + 7X_{4_1} + 10X_{5_1} + 12X_{6_1} - 7X_{1_2} - \\
&\qquad 2X_{2_2} - 6X_{3_2} - 11X_{4_2} - 8X_{5_2} - 14X_{6_2} \leq 0 \\
&\qquad\qquad \vdots \\
&3X_{1_5} + 3X_{2_5} + 8X_{3_5} + 7X_{4_5} + 10X_{5_5} + 12X_{6_5} - 7X_{1_6} - \\
&\qquad 2X_{2_6} - 6X_{3_6} - 11X_{4_6} - 8X_{5_6} - 14X_{6_6} \leq 0
\end{aligned} \tag{3.33}$$

$$\begin{aligned}
&7X_{1_1} + 2X_{2_1} + 6X_{3_1} + 11X_{4_1} + 8X_{5_1} + 14X_{6_1} - 6X_{1_2} - \\
&\qquad 12X_{2_2} - 4X_{3_2} - 3X_{4_2} - 6X_{5_2} - 2X_{6_2} \leq 0 \\
&\qquad\qquad \vdots \\
&7X_{1_5} + 2X_{2_5} + 6X_{3_5} + 11X_{4_5} + 8X_{5_5} + 14X_{6_5} - 6X_{1_6} - \\
&\qquad 12X_{2_6} - 4X_{3_6} - 3X_{4_6} - 6X_{5_6} - 2X_{6_6} \leq 0
\end{aligned} \tag{3.34}$$

Story and Wagner (108) have solved various problems of nine jobs and three machines using the above model by Gomory's

integer-linear programing algorithm. This was done on IBM 7090 integer programing package IP03. They have found that in many cases the number of iterations exceeded 1000. Thus, it has been concluded that practical method by integer programing has not yet been developed.

3.3 Manne's Formulation

The formulation of the machine scheduling problem of J jobs and M machines as an integer-linear programing problem developed by Manne (75), is presented. The following notation is required for developing the different constraints.

$X_{jmr} = \begin{cases} 1 \text{ if job j is scheduled on machine m for operation r} \\ 0 \text{ otherwise} \end{cases}$

S_{jm} starting time of job j on machine m,

$Y_{ijm} = \begin{cases} 1 \text{ if job i precedes job j (not necessarily directly) on machine m,} \\ 0 \text{ otherwise.} \end{cases}$

C a constant.

To construct the required constraints, consider two arbitrary jobs i and j. To avoid the overlapping of these two jobs on the same machine at the same time it is required that job i or job j must precede the other by sufficient time so that the first job be completed before the second starts. Therefore, either

$$S_{im} - S_{jm} \geq t_{jm}$$

or

$$S_{jm} - S_{im} \geq t_{im}$$

must hold. Note that, the above two inequalities indicate that job j precedes job i, and job i precedes job j respectively. Such inequalities cannot be handled by ordinary linear programming. Hence, the above condition is converted into two independent linear inequalities in integer variables such that

$$(C + t_{jm})Y_{ijm} + (S_{im} - S_{jm}) \geq t_{jm} \tag{3.35}$$

and

$$(C + t_{im})(1 - Y_{ijm}) + (S_{jm} - S_{im}) \geq t_{im} \tag{3.36}$$

It is evident that if the variable Y_{ijm} equals to zero then the first term in left hand side of inequality (3.35) vanishes. However, if Y_{ijm} equals to one, the first term in inequality (3.36) vanishes. Note that C is sufficiently large constant and it may be set such that

$$C = \sum_{j=1}^{J} \sum_{m=1}^{M} t_{jm}$$

This set of non-overlapping constraints leads directly to a non convex set of constraints upon the variables.

Operation precedence constraints are developed so that all jobs are not started on next machines until the previous operations are finished. For all, but the last operation of a job, the constraints are

$$\sum_{m=1}^{M} X_{jmr}(S_{jm} + t_{jm}) \leq \sum_{m=1}^{M} X_{jm,r+1} S_{jm} \tag{3.37}$$

$$r = 1, 2, \ldots, M-1,$$
$$j = 1, 2, \ldots, J$$

Now, the objective function may be expressed as,

Minimize

$$Z = \sum_{j=1}^{J} \sum_{m=1}^{M} X_{jmM} S_{jm} .$$

This will minimize the sum of the starting times of the last operations on all jobs, which means minimizing the schedule time. It should be pointed out that a variety of objective functions, other than the above, may be considered. Some of these are the minimization of the maximum flow time and the minimization of the mean tardiness.

Again, the same problem presented in the subsections 3.1 and 3.2 is formulated as follows:

Minimize

$$Z = \sum_{j=1}^{6} \sum_{m=1}^{3} X_{jm3} S_{jm} \tag{3.38}$$

subject to

$$(C + t_{jm})Y_{ijm} + (S_{im} - S_{jm}) \geq t_{jm}, \quad \begin{matrix} i = 1, 2, \ldots, 6 \\ j = 1, 2, \ldots, 6, \\ m = 1, 2, 3 \end{matrix} \tag{3.39}$$

$$(C + t_{im})(1 - Y_{ijm}) + (S_{jm} - S_{im}) \geq t_{im}, \tag{3.40}$$

$$\begin{matrix} i = 1, 2, \ldots, 6, \\ j = 1, 2, \ldots, 6, \\ m = 1, 2, 3 \end{matrix}$$

and

$$\sum_{m=1}^{3} X_{jmr}(S_{jm} + t_{jm}) \leq \sum_{m=1}^{3} X_{jm,r+1} S_{jm}, \tag{3.41}$$

$$\begin{matrix} r = 1, 2 \\ j = 1, 2, \ldots, 6 \end{matrix}$$

where

$$C = \sum_{j=1}^{6} \sum_{m=1}^{3} t_{jm}$$

$$= 124.$$

Manne has suggested that if the starting times S_{jm} are permitted to take continuous values rather than integer, the result might be more efficient and realistic. However, the computational time involved in Gomory's mixed integer programing algorithm (45) has also to be taken into account.

In summary, a comparison among the three formulations regarding the number of constraints and variables required, is made in Table 3.1.

Table 3.1

Comparison Among Different Formulations

Description	Bowman		Wagner		Manne	
	General	Example	General	Example	General	Example
Number of jobs	J	6	J	6	J	J
Number of machines	M	3	M	3	M	3
Number of possible sequences	$(J!)^M$	720	$(J!)^M$	720	$(J!)^M$	720
Number of variables	MJT	954	$J^2+4(J-1)$*	56	$JM+\frac{JM(J-1)}{2}$	63
Number of constraints	3MJT+JM+MT	3039	(4J-3)*	21	(M-1)J + JM(J-1)	102

*The expressions refer to the three-machine problem.

CHAPTER IV

FURTHER APPROACHES

In this chapter, some other techniques are presented and illustrated by suitable sample problems aceording to their limitations. These are Graphical, Graphieal-Dynamic Programing, and Heuristics techniques.

4.1 Graphical Approaeh

Sasieni et. al. (99) have presented the graphical approach to the job shop problem of two jobs and M machines. This technique is approximate but simple and easy to apply. Hardgrave and Nemhauser (46) have developed a geometric model and the corresponding computational algorithm. This is an extension to the approach presented in (99). The theoretical analysis has been extended to the case of J jobs and M machines. The concept of the technique is to interpret the sequencing problem geometrieally. The feasible sequences are represented by paths in J-dimensional rectangle. However, it is difficult to visualize higher than two dimensional geometric presentation. Hence, this teehnique is most suitable for the job shop problem of two jobs and M machines. The technique is also feasible for situations where a machine may proeess more than one job at a time; more than one machine of the same type exists; or a machine has to discontinue processing of a job.

A two dimensional coordinate system represents a problem of two jobs and M machines. The abscissa and thc ordinate repre-

sent processing times of job 1 and job 2, respectively. The processing times for each operation on the jobs, in the prescribed machine ordering are determined on the axes. If the total processing times for jobs 1 and 2 are given by N_1 and N_2, then the closed rectangle of width N_1 and length N_2 determine a region in which any point represents a degree of completion for each job. The two points (0,0) and (N_1,N_2) are called the origin and destination nodes, respectively. It is possible to construct a finite network, with these two nodes as extremes and the shortest path becomes the optimal. Some points in this rectangle are non-feasible. This means that the path should not pass through these points. Points which represent a degree of completion such that both jobs are being processed simultaneously on the same machine are non-feasible. Cartesian product of processing times of jobs on the two axes is taken and represented as rectangles. There will be as many rectangles as the number of machines. All the rectangles represent non-feasible regions and hence diagonal movement through them is forbidden. Hence, all the paths between the origin and destination nodes that do not pass through non-feasible regions form feasible sequences. A feasible path is represented by a continuous line consisting of vertical, horizontal and diagonal segments. The horizontal segment corresponds to the processing of job 1, vertical segment to the processing of job 2, and the diagonal segment to the processing of both jobs simultaneously.

It is evident that the feasible path which has the minimum of horizontal and vertical movements is the optimal. It should be noted that a diagonal movement over a unit of time is equivalent, in the sense of completing job, to a vertical movement of one time unit plus a horizontal movement of one time unit.

Sasieni et. al. (99) have suggested to pick up the optimal path by eye, but Hardgrave and Nemhauser (46) have formulated rules to determine rigorously and efficiently the shortest path. Their algorithm may be presented as follows:

Step 1: Set up the axes, one for each job.

Step 2: Enter the cumulative processing times for each operation on the jobs in their prescribed machine ordering.

Step 3: Represent the non-feasible regions by rectangles, sides of which show the processing times of jobs on those machines.

Step 4: Starting at the node (0,0) move diagonally towards the destination node until a region of non-feasibility is hit.

Step 5: Check where the path hits non-feasible region.

5.1 If it hits at left side of the rectangle as in Figure 4.1, return to the point Q and branch off horizontally, PQX, and vertically, PQRS. Go to step 4 with the new points S and X.

5.2 If it hits at the bottom of the rectangle as in Figure 4.2, return to the point Q and branch off

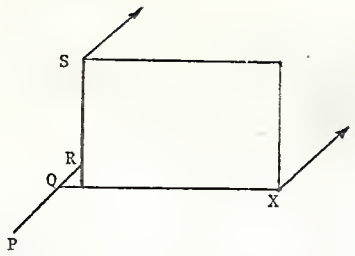

Figure 4.1 Path Hitting Infeasible Region at Left

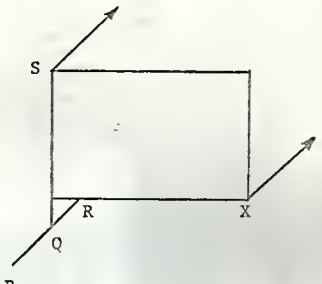

Figure 4.2 Path Hitting Infeasible Region at Bottom

horizontally, PQRX, and vertically, PWS. Go to step 4 with the new points S and X.

5.3 If the top or right edge of the rectangle is hit, move along that edge to the destination node (N_1, N_2).

Step 6: The problem is solved when all paths reach the destination nodes (N_1, N_2). The path(s) with minimum length, i.e., summation of horizontal, vertical, and diagonal segments, corresponds to the best sequence(s).

The above algorithm is illustrated by a sample job shop problem of two jobs and three machines. The same problem which was handled by Boolean algebra in subsection 2.5 is solved. For convenience, the processing time and machine ordering matrices are reproduced.

$$\tau^* = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \end{bmatrix} \quad , \quad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 23 & 22 & 21 \end{bmatrix}$$

A reference is made to the Figure 4.3. The cumulative processing times of jobs 1 and 2 are entered on abscissa and ordinate, respectively. The times during which the same machine is required by both jobs are noted and rectangular non-feasible regions are drawn. For example, machine 1 is required by job 1 from time zero to six, and by job 2 from time five to 17. Hence the rectangle formed by the points (0,5), (6,5), (6,17), and (0,17) represent non-feasible region for machine 1. Similar non-feasible regions are drawn for machines 2 and 3 according to step 3.

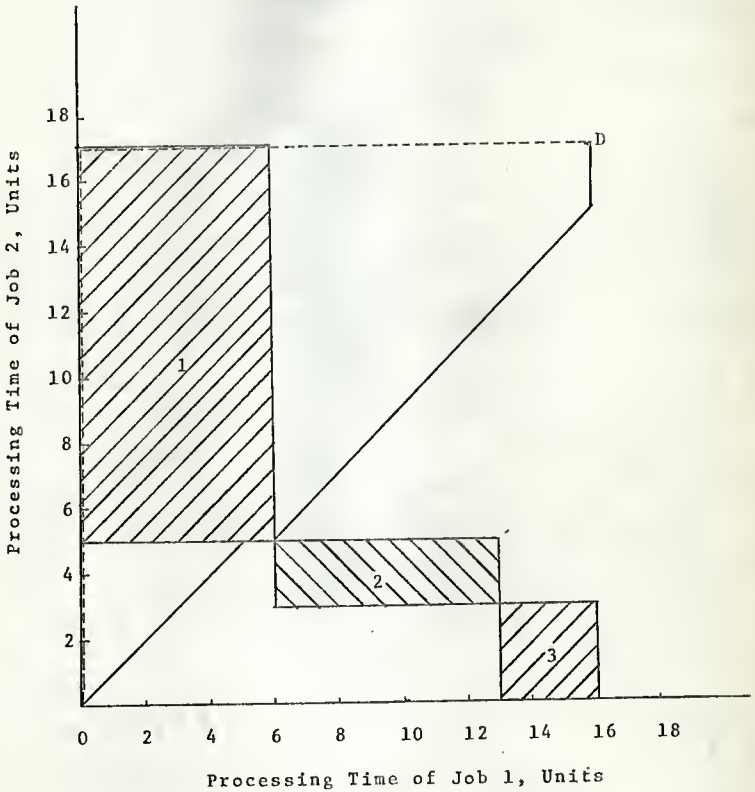

Figure 4.3 Graphical Solution for a Job Shop Problem of Size (2x3)

Following step 4 and moving diagonally, it is observed that the path hits non-feasibility region of machine 1. According to step 5.2, returning to node (0,0) and branching horizontally and vertically, two paths are obtained. Returning to step 4 at the two new points (6,5) and (0,17), the destination node is reached.

The two paths are shown as broken and continuous paths in Figure 4.3. The length of the broken path is

$$17 + 16 = 33,$$

and that of the continuous path is

$$5 + 1 + 10 + 2 = 18.$$

According to step 6, the best path is that represented by the continuous line. This path indicates that the job 2 is processed first on machines 2 and 3, and job 1 is processed first on machine 1. The schedule time is 18.

It is seen that the technique is quite simple for problems having not more than two jobs, otherwise, it is difficult to visualize the geometrical structure. It should be pointed out that this technique does not always produce optimal solution. As the number of machines increases, the accuracy diminishes.

If a machine processes more than one job at a time or if there is more than one machine of a given type, the non-feasible region corresponding to that machine vanishes. Also, if a job is permitted to be removed from a machine before completion allows vertical or horizontal movement through the corresponding non-feasible region.

Hardgrave and Nemhauser (46) have considered their approach as a geometric interpretation of Griffler and Thompson's approach (40) in the sense that the idea of "active schedule" is closely related to "paths in the network"; "conflict" is related to "hitting a non-feasible region"; and, "resolving a conflict in all possible ways" is related to a "branching around a non-feasible region."

4.2 Graphical-Dynamic Programming Approach

Held and Karp (47) have presented the dynamic programming formulations for J jobs and one machine problem. They have discussed the inclusion of precedence constraints. However, the technique is computationally effective for one machine problem.

Szwarc (109) has presented a solution to the problem of two jobs and M machines by combination of dynamic programing and graphical methods. The dynamic programming technique is based on the principle of optimality. This principle states that "an optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." Szwarc (109) has extended his technique to the case of J jobs and M machines. However, it is not guaranteed that the sequence obtained is feasible and the feasible sequence is optimal.

This graph-dynamic programming technique is based on certain properties of a feasible path. The process of setting the set of two axes; entering cumulative processing times for each operation

on the jobs in the prescribed technological ordering; and setting up of the rectangles comprising non-feasible regions, is the same as in the graphical technique.

Szwarc has defined a feasible path as one that satisfies the following conditions:

1. It is a continuous and consists of straight line segments, starting at the origin node (0,0) and terminating at the destination node (N_1, N_2).
2. All the segments of the line must be either horizontal, vertical or diagonal.
3. It does not intersect any non-feasible region.

The paths that satisfy the above conditions constitute feasible paths and the shortest one is the optimal sequence. All northwest and southeast corners of the non-feasible rectangles, along with the origin and the destination points are called nodes. The feasible path passes through these nodes. A node W_i with ordinates (X_i, Y_i) is said to be adjacent to the node W_j with ordinates (X_j, Y_j) if:

1. $X_i \leq X_j$ and $Y_i \leq Y_j$,
2. there exists at least one continuous path consisting of one or more straight lines with the following properties:
   - 2.1 the line links the nodes W_i and W_j,
   - 2.2 the line satisfies the conditions 2 and 3 of feasible path,
   - 2.3 no other node lies on the path.

The distance between two adjacent nodes W_i and W_j is given by

$$d(W_j, W_i) = \max[(Y_i - Y_j) - (X_i - X_j), 0].$$

The technique consists of drawing a set of axes. As in the graphical technique the cumulative processing times of both jobs are entered on the axes. Regions of non-feasibility are also drawn. Two boundary paths, upper and lower are drawn. The boundary paths are such that all feasible paths will lie on or within them. The shortest distance, of a path through the origin and destination is calculated by applying dynamic programing. This shortest distance plus $\max_j[\sum_{m=1}^{M} t_{jm}]$ will give the optimal schedule time and the corresponding path gives optimal schedule.

The algorithm may now be stated as follows:

Step 1: Set up the axes, one for each job.

Step 2: Enter the cumulative processing times for each operation on the jobs in their prescribed machine ordering.

Step 3: Represent the non-feasible regions by rectangles, sides of which show the processing times of jobs on those machines.

Step 4: Label all the nodes, i.e., all northwest and southeast corners, origin and destination points as, $W_1, W_2, \ldots, W_k$, in decreasing order of coordinates with priority of abscissa.

Step 5: Determine the set of nodes $n(W_j)$ in which W_j is adjacent to every node in the set.

Step 6: Determine the distance of each node of the set $n(W_j)$ from the node W_j such that

$$d(W_j, W_i) = \max[(Y_i - Y_j) - (X_i - X_j), 0]$$

Step 7: Determine the minimum distance between the destination node and all other nodes, $f(W_j)$, such that

$$f(W_j) = [d(W_j, W_i) + f(W_i)], \quad \begin{matrix} j = 2, 3, \ldots, k \\ i \neq j, \end{matrix}$$

where

$f(W_i)$ = Distance of node W_i from the destination node.

Step 8: Select the path(s) through the nodes having minimum distance between the origin and the destination nodes. This is the optimal path and the optimal schedule time T* is given by:

$$T^* = \max_j \left[\sum_{m=1}^{M} t_{jm} \right] + \text{optimal distance between the origin and destination nodes.}$$

Step 9: Draw the upper and lower boundary paths.

Step 10: Starting at the origin node, move diagonally till the boundary path is hit.

Step 11: Move along the boundary path upwards or to the right until a node is reached.

Step 12: Starting again at this node move diagonally until the boundary path is hit.

Step 13: Repeat steps 11 and 12 till the destination node is reached. The path so obtained gives the optimal schedule.

To illustrate the technique, the sample problem solved by the graphical technique in subsection 4.1, is solved. For

convenience, the processing time and machine ordering matrices are reproduced as follows:

$$T^* = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \end{bmatrix} \quad , \quad M^* = \begin{bmatrix} 11 & 12 & 13 \\ 23 & 22 & 21 \end{bmatrix} .$$

Referring to Figure 4.4, the cumulative processing times of jobs 1 and 2 are entered on ordinate and abscissa respectively. Regions of non-feasibility for all three machines are also drawn as in graphical technique.

Following step 4, the nodes are labelled as $W_1, W_2, \ldots, W_6$; in decreasing order of coordinates with priority of abscissa. Therefore, the origin node is W_6 and the destination node is W_1.

Next, following step 5 the set of nodes $n(W_j)$ is determined, in which W_j is adjacent to all nodes of the set. For example, the node W_1 is adjacent to each of the nodes W_2, W_3, W_4, and W_5. Hence $n(W_1)$ will contain the set of nodes W_2, W_3, W_4, and W_5. Table 4.1 contains this set for all nodes.

The distances of each node of the set $n(W_j)$ from the node W_j is calculated and entered in Table 4.1.

Table 4.1
Table of Nodes

W_j	$n(W_j)$	$d(W_j, W_i)$	$f(W_j)$
W_1	W_2, W_3, W_4, W_5	16, 0, 0, 0	0
W_2	W_6	0	16
W_3	W_6	1	0
W_4	W_6	10	0
W_5	W_6	16	0
W_6			1

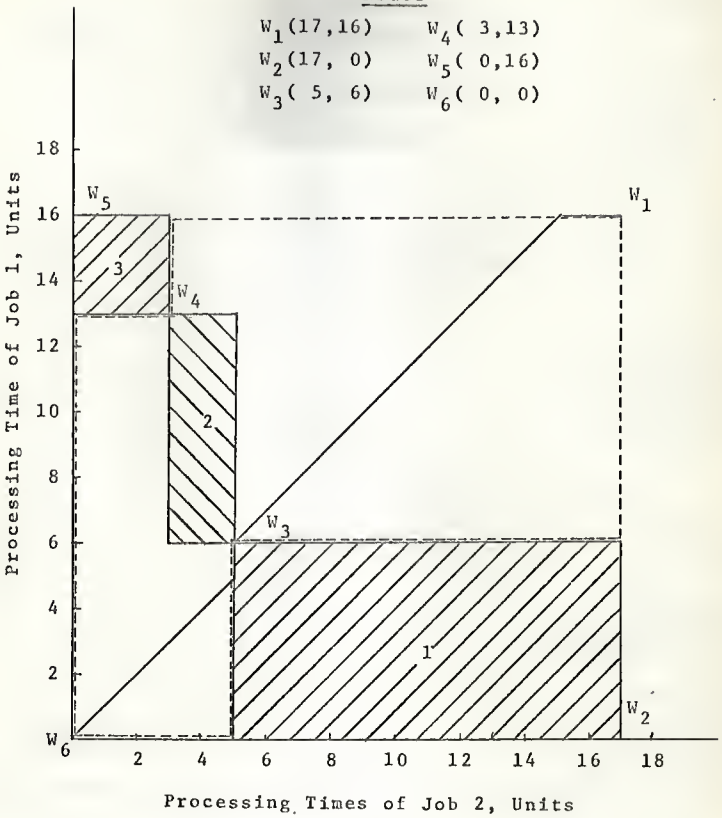

Figure 4.4 Graphical-Dynamic Programing Solution for a Three Machines Job Shop Problem of Size (2x3)

The distances between adjacent nodes are calculated as follows:

$$d(W_2, W_1) = \max[(Y_1 - Y_2) - (X_1 - X_2), 0]$$
$$= \max[(16 - 0) - (17 - 17)\ 0]$$
$$= \max[16\ \ 0]$$
$$= 16,$$

$$d(W_3, W_1) = \max[(Y_1 - Y_3) - (X_1 - X_3), 0]$$
$$= \max[(16 - 6) - (17 - 5)\ 0]$$
$$= \max[-2\ 0]$$
$$= 0,$$

$$\vdots \qquad\qquad \vdots$$

$$d(W_6, W_5) = \max[(Y_5 - Y_6) - (X_5 - X_6), 0]$$
$$= \max[(16 - 0) - (0 - 0)\ 0]$$
$$= \max[16\ \ 0]$$
$$= 16.$$

All the distances are entered in Table 4.1.

According to step 7, shortest path between any node and the destination node is determined. Let $f(W_j)$ denote the cumulative distance of the node W_j from the node W_1. Then, it follows that

$$f(W_1) = 0\ ,$$

$$f(W_2) = [d(W_2, W_1) + f(W_1)]\ ,$$

$$f(W_3) = [d(W_3, W_1) + f(W_1)]\ ,$$

$$f(W_4) = [d(W_4, W_1) + f(W_1)]\ ,$$

$$f(W_5) = [d(W_5, W_1) + f(W_1)] ,$$

and

$$f(W_6) = \min[d(W_6, W_2) + f(W_2), d(W_6, W_3) + f(W_3) ,$$
$$d(W_6, W_4) + f(W_4), d(W_6, W_5) + f(W_5)].$$

Note that the path W_6 to W_1 may be through W_2, W_3, W_4, and W_5 and the shortest of the four paths is desired. Hence on substitution of numerical values.

$$f(W_2) = 16 + 0$$
$$= 16,$$
$$f(W_3) = 0 + 0$$
$$= 0,$$
$$f(W_4) = 0 + 0$$
$$= 0,$$
$$f(W_5) = 0 + 0$$
$$= 0,$$

and

$$f(W_6) = \min[0 + 16 \quad 1 + 0 \quad 10 + 0 \quad 16 + 0],$$
$$= \min[16 \quad 1 \quad 10 \quad 16],$$
$$= 1.$$

This indicates that the shortest path lies along the nodes W_1, W_3, and W_6 and the distance between the nodes W_1 and W_6 is 1. The optimal schedule time, T*, is

$$T^* = \max[\sum_{m=1}^{3} t_{jm}] + 1$$
$$= \max[16 \ 17] + 1$$
$$= 17 + 1$$
$$= 18.$$

To obtain the optimal schedule, a path is drawn through the nodes W_6, W_3, and W_1. To do so, the upper and lower boundary paths are drawn according to step 9. Starting from the origin and moving diagonally, the non-feasibility region is hit. Hence, moving vertically till the node W_3 is reached and again moving diagonally, upper boundary path is hit. On moving horizontally to the right, the destination node is reached according to step 13. This path gives the optimal schedule which indicates that process job 2 on machines 2 and 3 first, and job 1 on machine 1 first. The optimal schedule time T* is given by length of the path. Thus,

$$T^* = 5 + 1 + 10 + 2$$
$$= 18 .$$

It should be noted that the above solution is the same as that obtained by graphical and boolean algebra techniques. Note also that the optimal schedule time can be checked by the graphical-dynamic programing technique.

It has been found that this technique is quite simple and the solution is obtained very quickly. The problem involves at most 2M + 2 nodes.

4.3 Heuristic Techniques

The machine scheduling problem appearing most commonly in industry is dynamic in nature. It becomes complicated because of many variables such as variability in factory performance due to fluctuating demand, unpredictable labor performance, unexpected

machine delays, variable product quality, and queuing effects. One of the most practical techniques to solve such problems is simulation and then imposing heuristics, i.e., rules of thumb and priority rules.

Simulation has been defined as the process of simulating the behavior of a system to study its effects to specific changes. The results help gain insights, test hypotheses, demonstrate or verify new ideas, establish feasibility, and compare alternatives. Computer simulation however, is seldom an exact analogue of the operation of an actual system. To simulate factory operations, it is necessary to have a model which provides a formal statement of the system behavior. Simulation study can be exploratory in that, new design will result from information obtained by simulation. The second type is a form of statistical sampling. This is referred to as Monte Carlo sampling in which a given design is subject to many conditions in order to determine its suitability.

Eilon and Hodgson (31) have developed a simulation model for job shop consisting of two identical machines operating in parallel. It is assumed that the jobs arrive according to Poisson distribution; form a single queue and may be processed on either machine. Experimenting with 2000 arriving jobs for various loading rules, it has been found that loading the job in the queue with the shortest processing time first, yields best results in terms of minimizing job waiting times, flow time, machine idle times, delay factors and queue length.

Conway et. al. (25) have conducted an experiment on priority rules using a five machines job shop problem with 100 arriving jobs. The investigation led them to conclude that a general statement regarding effectiveness of a particular priority rule is not obvious. Priority rules can differ with respect to the mean lateness of all jobs. It can accelerate one or more jobs at the expense of others. However, simple priority rules can do an effective job of reducing weighed average completion time, as compared to the selection of jobs at random.

Gere (34) has developed computer simulation program to study the effects of various priority rules and heuristics individually, as well as combinations of them. The objective function considered has been that of meeting due-dates, or failing this, minimizing the sum of lateness. The experiments include static and dynamic situations. In the dynamic situation, arriving jobs were assumed to be governed by Poission distribution. One experiment involves 25 static problems of six to 20 jobs, one to 16 operations per job, and four to 16 machines. Other experiment includes 16 dynamic problems involving 20 to 60 jobs, one to 16 operations per job and four to 16 machines. It has been concluded that:

1. The selection of priority rule for discriminating between jobs for competing on the machines is not as important as the selection of a set of heuristics.
2. There is little difference in effectiveness of the priority rules after they are combined with two or more heuristics.

3. On comparing the sequences obtained by heuristics, with the 3000 sequences generated by Monte Carlo technique for flow shop problems of up to 100 jobs and 10 machines reported by Heller (51), Gere (34) has found that heuristics are more effective in handling the problem of finding the minimum schedule time.

Regarding the priority rules, Gere has concluded that:

1. Non-random rules are significantly more effective than random rules.
2. There is little choice between rules based in some reasonable way upon job slack (idle time available before due-date).
3. The shortest imminent operation (SIO) rule is less effective than a job slack based rule.
4. The alternate operation and look ahead heuristics are effective, both individually and collectively.
5. There lies a real difference in results obtained with the use of different priority rules. A poor rule augmented by the heuristics is better than a good rule without them.

Priority rule consists of assigning a scalar value to each of the waiting jobs. The job having the minimum scalar value is scheduled first. In case of a tie, the job with smaller job number is scheduled. The priority rules used for job assignments in the experiments conducted by Conway et. al. (25), Eilon and Hodgson (31), Gere (34), and Rowe (93) are summarized as follows. The priority is:

1. random. (25), (31), (34), and (93).
2. first come first served. (25), (31), (34), and (93).

3. inversely related to due-date. (25), (31), and (93).

4. inversely related to the remaining slack-time. (25) and (34). This rule means, priority is given to jobs for which the time remaining to due-date less, the remaining processing time is a minimum.

5. inversely related to the job slack ratio. (34). Job slack ratio is the ratio of free time available before the due-date to the total time remaining until the due-date.

6. inversely related to the modified job slack ratio (34). This takes into account machine loading by adding the expected delay time to each operation time.

7. shortest imminent operation (SIO). (25), (31), (34), and (93).

8. directly related to the processing time on the next operation. (25), (31), and (93). Maximum priority is given to the job having longest operation.

9. directly related to the number of operations left. (25).

10. inversely related to the number of operations left. (25).

11. inversely related to the sum of processing times for all remaining operations. (25).

12. directly related to the sum of processing times for all processing times for all remaining operations. (25).

13. directly related to dollar value of job. (25).

14. directly related to the subsequent move. (25).

Maximum priority is given to the job which, will go on the next machine which has the shortest queue, in terms of least processing time.

As described earlier, Gere (34) has experimented on some heuristics. These are actually tailor made approaches. If the immediate situation calls for action extraneous to the priority rule, then exception to the general rule is taken. This, then becomes the heuristic. The heuristics employed by Gere are as follows:

1. Alternate Operation: Scheduling job according to some rule it is checked if some other job becomes critical (slack becoming negative). If it does, schedule is changed, otherwise the previous one is followed.

2. Look Ahead: After scheduling, it is checked if there is a critical (late or nearly late) job due to reach the particular machine at some future time, yet before the scheduled job is completed. If so, that job is scheduled. Effect of this job on other jobs is checked. Depending on this effect, the schedule is retained or replaced.

3. Insert: Once "look ahead" job is scheduled and there is idle time, and if there is a job in the queue whose operation can be completed in this idle time, it is scheduled.

4. Subset of Critical Jobs: In this, a subset of critical jobs is selected. These jobs are scheduled according to a priority rule and remaining jobs are scheduled around these jobs. An advantage of this heuristic is that it points out conflicts within the subset.

5. Re-do with Adjusted Due Dates: When a schedule is completed and if at least one job is late, decrease the due-date of each

late job by the time it was late and lay out the schedule again. This will tighten the jobs previously late. It is not advisable to repeat this process more than twice or thrice as it might degenerate.

6. Flexibility: Flexibility in scheduling means an operation could be "squeezed into" the available time without much delaying other jobs. If time-transcending mode is employed, then opportunities of squeezing in do appear. But if time-progression scheduling is employed, then there is no need for flexible scheduling.

7. Manipulation: This refers to shifting to "tailor-made" approach, attacking the problem as a unique instead of following the rules the second time. This heuristic has not been tested but Gere has recommended as future step in research.

In order to make comparisons among the various priority rules and heuristics, the following criteria have been recorded in simulation:

1. Job waiting time.
2. Queue length (the number of jobs waiting to be processed at the time when a machine is free to accept next job).
3. Flow time (the time taken from the arrival of a job to its completion).
4. Delay factor (a ratio of flow time to processing time).
5. Facility idle time.
6. Missed due-date time.

7. Total number of jobs late.

It appears that especially for dynamic scheduling situations, heuristics is a powerful tool.

CHAPTER V

SUMMARY AND CONCLUSIONS

The machine scheduling problem is one of the most challenging problems posed in operations research. The problem arises whenever a number of jobs has to be processed on various machines in order to achieve an objective. The problem is of special interest because of the large number of computational effort required in its solution.

Attempts have been made by researchers to develop efficient procedure to find an optimal sequence of jobs from a considerably large set of feasible sequences. The aim has been, to curtail the number of sequences in searching for the optimal. All algorithms developed so far, are based on several assumptions. The most common measure of effectiveness has been the schedule time.

In order to study the merits of various techniques with regard to the computational effort and optimality of the solution, a sample problem has been solved by some of the available techniques. Table 5.1 shows a summary of research made in solving the machine scheduling problem.

The most simple procedure to solve a flow shop problem of J jobs and two machines is the Direct Technique. It guarantees an optimal solution. A six jobs and two machines flow shop problem has been solved by this technique, which is also feasible for a special case of J jobs and three machines flow shop problem.

Table 5.1

Summary of Research on Machine Scheduling Problem*

Problem Size	Problem Type	Technique	Author
Jx2	Flow Shop	Direct	Johnson (63),(64)
JxM	Flow Shop	Extended Direct	Smith and Dudek (105) Dudek and Ghare (29) Dudek and Teuton (30)
JxM	Flow Shop	Branch-and-Bound	Lomnicki (74) Brown and Lomnicki (20) Ignall and Schrage (56) McMahon and Burton (76)
JxM	Job Shop+	Lower Bound	Brooks and White (19) White (113)
JxM	Job Shop+	Decomposition	Ashour (6),(7)
2xM	Job Shop+	Boolean Algebra	Akers and Friedman (5)
JxM	Job Shop+	Integer-Linear Programing	Bowman (18) Wagner and others (42),(108),(111) Manne (75)
2xM	Job Shop+	Graphical	Sasieni and others (99) Hardgrave and Nemhauser (46)
Jx1	Job Shop+	Dynamic Programing	Held and Karp (47)
2xM	Job Shop+	Graphical-Dynamic Programing	Szwarc (109)
JxM	Job Shop+	Heuristics, Priorities and Simulation	Eilon and Hodgson (31) Conway and others (25) Gere (34) Rowe (93)

* Those covered in this report.

\+ These techniques are also feasible for flow shop problems.

Extended Direct Technique also guarantees an optimal solution for flow shop problems of J jobs and M machines. In solving a six jobs and three machines problem, it has been observed that the technique requires excessive amount of computation.

Branch-and-Bound Technique for flow shop problems of J jobs and M machines is computationally efficient when compared to the previous one. However, it has been found that an optimal solution is not guaranteed. This is because, in solving the sample problem, it has been observed that branching at the node having the least lower bound does not generate an optimal solution. It has been reported that the concept of dominated nodes and reversed approach reduces the search by 13 and 33 percent, respectively.

The Lower Bound Technique when applied to flow shop problems of J jobs and M machines is simpler than the Branch-and-Bound Technique. However, an optimal solution is also not guaranteed.

The Boolean Algebra Technique is efficient for job shop problem. It generates a set of feasible sequences each of which may be optimal for some set of processing times. The technique is limited to job shop problems having two jobs and M machines.

The Decomposition Technique reduces the problem of constructing and evaluating feasible sequences to a limited number of arrangements. It produces at least a near-optimal solution.

Results of experiments conducted by various researchers are summarized in Table 5.2. Since the computers used are not the same, their speeds, for comparison purpose, are listed in

Table 5.2

Experimental Results of Various Researchers

Investigator	Technique	Number of Problems Solved	Problem Size (JxM)	Mean+ Computing Time
Smith and Dudek (105)	Extended Direct (IBM 1620-II)	15	3x3	2.4
		20	4x3	8.9
		20	5x3	35.4
		20	6x3	118.0
		15	7x3	210.0
		4	8x3	1222.0
		15	3x5	4.4
		17	4x5	20.1
		18	5x5	90.3
		15	6x5	564.0
		3	7x5	2500.0
Brown and Lomnicki (20)	Branch-and-Bound (ICT 1301)	100	6x3	1.15
		50	7x3	2.79
		50	8x3	7.00
		50	10x3	112.00
		100	6x5	1.72
		100	6x7	2.45
Ignall and Schrage (56)	Branch-and-Bound (CDC 1604)	50	4x3	0.55
		50	5x3	0.61
		50	6x3	0.94
		50	7x3	1.19
		50	8x3	4.2
		50	9x3	5.7
		50	10x3	8.7

+ In seconds.

Table 5.2 (cont'd.)

Investigator	Technique	Number of Problems Solved	Problem Size (JxM)	Mean+ Computing Time
McMahon and Burton	Branch-and-Bound (CDC 3600)	16	6x3	0.079
		16	7x3	0.135
		16	8x3	0.505
		16	9x3	5.593
		16	10x3	7.578
Ashour (6)	Decomposition with K = J/2 (IBM-7044)	107	6x3	3.7
		1	6x3*	12.6
		1	6x4	6.8
		1	6x5	8.5
		1	6x10	12.4
		11	8x3	81.7
		1	8x3*	147.0
		1	8x4	96.1
		1	12x4	482.7
		1	20x3	290.2
		1	20x5	598.7
		1	40x3	413.7

+ In seconds.

* Job Shop Problems.

Appendix B, Table B.1.

Integer-linear programing formulations do not appear attractive. The main reasons are: first, the formulations involve a large number of constraints and variables even for small size problems. Second, the integer-linear programing algorithm is not computationally efficient.

The Graphical Technique for job shop problems of two jobs and M machines is a simple technique. However, as the number of machines increases, the solution moves away from the optimal. It is not possible to check whether the solution is optimal or not. This difficulty is eliminated in the following technique.

The combination of Graphical and Dynamic Programing Techniques guarantees an optimal solution. However, this technique is also feasible for job shop problem of two jobs and M machines. When applied to large problems, it is not guaranteed that the solution is optimal.

The above discussions relate to static situation studied in this report. Mostly, the situations encountered in industry are dynamic in nature. Moreover the problem becomes complicated because of the machine breakdown, possibility of alternative routes for jobs, and probabilistic nature of processing times. The most promising approach for such situations appears to be simulation, and utilizing priority rules, heuristics and/or combinations of them. Researchers have found that combination of priority and heuristic rules produce better results than pure Monte Carlo simulations.

APPENDIX A

In this appendix, proofs of the lemmas and theorems of the Direct Technique presented in subsection 2.1 are given.

Lemma 1: The sequence on either machine can be made the same as that of the other machine without loss of time.

This means that for an optimal sequence, it is sufficient to consider the case in which the jobs are processed in the same order through both the machines.

Proof: The J jobs can be arranged in J! ways on machine 1. A problem of three jobs and two machines is considered. Let the sequence be {3 2 1} for machine 1 and {1 2 3} for machine 2. Then it is clear that jobs 1 and 3 can be interchanged without loss of time as shown below:

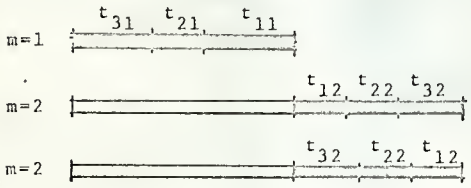

Figure A.1 Gantt Chart for a Job Shop Problem of Size (3x2)

It is observed that the sequences are the same for both the machines. Similarly it follows that if needed, one can make sucessive interchanges without loss of time in order to make

sequences the same on both the machines.

Since both jobs have the same ordering the schedule time will consist of idletime and processing times of all jobs on machine 2. An expression for total idle time on machine 2 is derived.

Let I_{jm} = idle time on machine m immediately prior to processing job j on it.

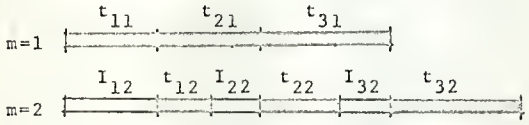

Figure A.2 Gantt Chart for a Flow Shop Problem of Size (3x2)

In the Figure A.2,

$$I_{12} = t_{11}$$

$$I_{22} = \max\left[\sum_{j=1}^{2} t_{j1} - \sum_{j=1}^{1} t_{j2} - \sum_{j=1}^{1} I_{j2}\ ,\ 0\right]$$

Hence,

$$\sum_{j=1}^{2} I_{j2} = I_{12} + I_{22}$$

$$= I_{12} + \max\left[\sum_{j=1}^{2} t_{j1} - \sum_{j=1}^{1} t_{j2} - \sum_{j=1}^{1} I_{j2}\ ,\ 0\right]$$

$$= \max \left[I_{12} + \sum_{j=1}^{2} t_{j1} - \sum_{j=1}^{1} t_{j2} - I_{12} \, , \; 0 + I_{12} \right]$$

$$= \max \left[\sum_{j=1}^{2} t_{j1} - \sum_{j=1}^{1} t_{j2} \, , \; t_{11} \right] .$$

$$I_{32} = \max. \left[\sum_{j=1}^{3} t_{j1} - \sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{2} I_{j2} \, , \; 0 \right]$$

Therefore,

$$\sum_{j=1}^{3} I_{j2} = I_{12} + I_{22} + I_{32}$$

$$= \max \left[\sum_{j=1}^{2} I_{j2} + \sum_{j=1}^{3} t_{j1} - \sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{2} I_{j2} \, , \; 0 + \sum_{j=1}^{2} I_{j2} \right]$$

$$= \max \left[\sum_{j=1}^{3} t_{j1} - \sum_{j=1}^{2} t_{j2} \, , \; \sum_{j=1}^{2} I_{j2} \right]$$

$$= \max \left\{ \sum_{j=1}^{3} t_{j1} - \sum_{j=1}^{2} t_{j2}, \; \max \left[\sum_{j=1}^{2} t_{j1} - \sum_{j=1}^{1} t_{j2}, \; t_{11} \right] \right\}$$

$$= \max \left[\sum_{j=1}^{3} t_{j1} - \sum_{j=1}^{2} t_{j2}, \; \sum_{j=1}^{2} t_{j1} - \sum_{j=1}^{1} t_{j2}, \; \sum_{j=1}^{1} t_{j1} \right]$$

Extending this to the case of J jobs,

$$\sum_{j=1}^{J} I_{j2} = \max \left[\sum_{j=1}^{J} t_{j1} - \sum_{j=1}^{J-1} t_{j2}, \; \sum_{j=1}^{J-1} t_{j1} - \sum_{j=1}^{J-2} t_{j2}, \; \sum_{j=1}^{J-2} t_{j1} - \sum_{j=1}^{J-3} t_{j2} \, , \right.$$

$$\left. \ldots, \; \sum_{j=1}^{2} t_{j1} - \sum_{j=1}^{1} t_{j2}, \; \sum_{j=1}^{1} t_{j1} \right]$$

If,

$$D_J = \sum_{j=1}^{J} t_{j1} - \sum_{j=1}^{J-1} t_{j2} \quad ,$$

Total time idle on machine 2 may be expressed as,

$$\sum_{j=1}^{J} I_{j2} = \max [D_J, D_{J-1}, \ldots, D_1]$$

and the schedule time

$$T = \sum_{j=1}^{J} t_{j2} + \sum_{j=1}^{J} I_{j2}$$

$$= \sum_{j=1}^{J} t_{j2} + \max [D_J, D_{J-1}, \ldots, D_1]$$

It is observed that the total idle time on machine 2 has been expressed in terms of the processing times. The schedule time T has two components. The first, $\sum_{j=1}^{J} t_{j2}$ is constant. Hence it follows that the optimal sequence will be one, that has minimum of idle time on machine 2. Consider two sequences S and S', the latter formed by interchanging job j_x and job j_{x+1} in the former.

$$S = 1, 2, \ldots, j_{x-1}, j_x, j_{x+1}, \ldots, J,$$

$$S' = 1, 2, \ldots, j_{x-1}, j_{x+1}, j_x, \ldots, J.$$

Let

$$F(S) = \max_{1 \le x \le J} [D_{j_x}] ,$$

and

$$F(S') = \max_{1 \le x \le J} [D'_{j_x}]$$

where

$$D'_{j_x} = \sum_{x=1}^{J} t'_{j_x 1} - \sum_{x=1}^{J-1} t'_{j_x 2} , \qquad t'_{j_x 1} = t_{j_x 1} ,$$

$$t'_{j_x 2} = t_{j_x 2} , \text{ for } x \neq x, x+1$$

For $x = x, x+1$,

$$t'_{j_x 1} = t_{j_{x+1} 1} ,$$

$$t'_{j_x 2} = t_{j_{x+1} 2} ,$$

$$t'_{j_{x+1} 1} = t_{j1} ,$$

and

$$t'_{j_{x+1} 2} = t_{j2} .$$

Hence

$$D'_{j_x} = D_{j_x} \qquad \text{for } x \neq x, x+1 .$$

Thus

$F(S') = F(S)$ unless possibly if

$$\max[D_{j_x}, D_{j_{x+1}}] \neq \max[D'_{j_x}, D_{j_{x+1}}]$$

Theorem 1: An optimal policy is given by the following rule. The job j_x precedes the job j_{x+1} if

$$\max[D_{j_x}, D_{j_{x+1}}] < \max[D'_{j_x}, D_{j_{x+1}}] \qquad (A.1)$$

If there is an equality either sequence is optimal provided it is consistent with all the definite preferences (refer case IV in lemma 2).

By subtracting $\sum_{x=1}^{x+1} t_{j_x 1} - \sum_{x=1}^{x-1} t_{j_x 2}$ from each term in the inequality (A.1),

$$\max[-t_{j_x 2}, -t_{j_{x+1} 1}] < \max[-t_{j_{x+1} 2}, -t_{j_x 1}]$$

or

$$\min[t_{j_x 1}, t_{j_{x+1} 2}] < \min[t_{j_{x+1} 1}, t_{j_x 2}] \qquad (A.2)$$

The inequality (A.2) is transitive leading to a unique sequence S^* except for indifferent elements. Hence $F(S^*) \leq F(S_0)$ where S_0 is any other sequence.

Lemma 2: The inequality (A.2) is transitive.

The proof depends on the condition that if

$$\min[t_{11}, t_{22}] \leq \min[t_{21}, t_{12}],$$

and

$$\min [t_{21}, t_{32}] \leq \min [t_{31}, t_{22}] \quad ,$$

then

$$\min [t_{11}, t_{32}] \leq \min [t_{31}, t_{12}] \quad .$$

Proof:

Case I: If

$$t_{11} \leq t_{22}, t_{21}, t_{12},$$

and

$$t_{21} \leq t_{32}, t_{31}, t_{22}$$

then

$$t_{11} \leq t_{21} \leq t_{31},$$

$$t_{11} < t_{12}$$

so that

$$t_{11} \leq \min [t_{31}, t_{12}].$$

Case II: If

$$t_{22} \leq t_{11}, t_{21}, t_{12},$$

and

$$t_{32} \leq t_{21}, t_{31}, t_{22},$$

then

$$t_{32} \leq t_{22} \leq t_{12}, \qquad t_{32} \leq t_{31},$$

so that

$$t_{32} \leq \min [t_{31}, t_{12}].$$

Case III: If

$$t_{11} \leq t_{22}, t_{12}, t_{21}$$

and

$$t_{32} \leq t_{21}, t_{31}, t_{23},$$

then

$$t_{11} \leq t_{12},$$

$$t_{32} \leq t_{31}$$

so that

$$\min [t_{11}, t_{32}] \leq \min [t_{31}, t_{12}] .$$

Case IV: If

$$t_{22} \leq t_{11}, t_{21}, t_{12},$$

and

$$t_{21} \leq t_{32}, t_{31}, t_{22},$$

then

$$t_{21} = t_{22}$$

and job 2 will be indifferent to jobs 1 and 3. In this, case I may or may not precede case III. But, there is no contradiction to transitivity as long as jobs 1 and 3 are scheduled first and job 2, anywhere.

Lemma 3: An optimal sequence can be reached if the same machine ordering is assumed for all the jobs.

Proof: By lemma 1, the sequence of jobs on the first and the third machines can be made the same as that of the second. Therefore, the first two machines have the same job orders and the last two machines have the same job orders. Hence all the three machines can have the same job ordering without loss of time.

Now, as all the three machines have the same job sequence the schedule time again is a function of the total idle time of the last machine. An expression for this is developed.

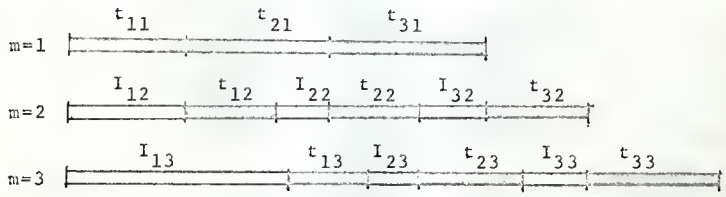

Figure A.3 Gantt Chart for a Flow Shop Problem of Size (3x3)

In Figure A.3,

$$I_{13} = I_{12} + t_{12}$$

$$= t_{11} + t_{12}$$

$$I_{23} = \max \left[\sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{1} t_{j3} + \sum_{j=1}^{2} I_{j2} - \sum_{j=1}^{1} I_{j3}, 0 \right]$$

$$\sum_{j=1}^{2} I_{j3} = I_{13} + I_{23}$$

$$= I_{13} + \max\left[\sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{1} t_{j3} + \sum_{j=1}^{2} I_{j2} - \sum_{j=1}^{1} I_{j3},\ 0\right]$$

$$= \max\left[I_{13} + \sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{1} t_{j3} + \sum_{j=1}^{2} I_{j2} - \sum_{j=1}^{1} I_{j3},\ I_{13}\right]$$

$$= \max\left[\sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{1} t_{j3} + \sum_{j=1}^{2} I_{j2},\ I_{13}\right]$$

$$I_{33} = \max\left[\sum_{j=1}^{3} t_{j2} - \sum_{j=1}^{2} t_{j3} + \sum_{j=1}^{3} I_{j2} - \sum_{j=1}^{2} I_{j3},\ 0\right]$$

Hence,

$$\sum_{j=1}^{3} I_{j3} = \sum_{j=1}^{2} I_{j3} + I_{33}$$

$$= \sum_{j=1}^{2} I_{j3} + \max\left[\sum_{j=1}^{3} t_{j2} - \sum_{j=1}^{2} t_{j3} + \sum_{j=1}^{3} I_{j2} - \sum_{j=1}^{2} I_{j3},\ 0\right]$$

$$= \max\left[\sum_{j=1}^{2} I_{j3} + \sum_{j=1}^{3} t_{j2} - \sum_{j=1}^{2} t_{j3} + \sum_{j=1}^{3} I_{j2} - \sum_{j=1}^{2} I_{j3},\ \sum_{j=1}^{2} I_{j3}\right]$$

$$= \max\left[\sum_{j=1}^{3} t_{j2} - \sum_{j=1}^{2} t_{j3} + \sum_{j=1}^{3} I_{j2},\ \sum_{j=1}^{2} I_{j3}\right]$$

$$= \max\left[\sum_{j=1}^{3} t_{j2} - \sum_{j=1}^{2} t_{j3} + \sum_{j=1}^{3} I_{j2},\ \sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{1} t_{j3} + \sum_{j=1}^{2} I_{j2},\ I_{13}\right]$$

This is for three jobs. Extending for J jobs,

$$\sum_{j=1}^{J} I_{j3} = \max\Big[\sum_{j=1}^{J} t_{j2} - \sum_{j=1}^{J-1} t_{j3} + \sum_{j=1}^{J-1} I_{j2},\ \sum_{j=1}^{J-1} t_{j2} - \sum_{j=1}^{J-2} t_{j3} + \sum_{j=1}^{J-2} I_{j2},$$

$$\ldots,\ \sum_{j=1}^{2} t_{j2} - \sum_{j=1}^{1} t_{j3} + \sum_{j=1}^{2} I_{j2},\ I_{13}\Big]$$

If,

$$E_J = \sum_{j=1}^{J} t_{j2} - \sum_{j=1}^{J-1} t_{j3}$$

and

$$D_J = \sum_{j=1}^{J} t_{j1} - \sum_{j=1}^{J-1} t_{j2},$$

the total idle time on machine 3 becomes

$$\sum_{j=1}^{J} I_{j3} = \max_{1 \le j \le J} [E_j + D_j]$$

and the schedule time,

$$T = \sum_{j=1}^{J} t_{j3} + \sum_{j=1}^{J} I_{j3}$$

$$= \sum_{j=1}^{J} t_{j3} + \max_{1 \le j \le J} [E_j + D_j]$$

As in the two-job case, considering two sequences S and S' (S' is formed by interchanging jobs j_x and j_{x+1} in S), the E's

and D's are unchanged except possibly those with subscripts j_x and j_{x+1}.

On comparing

$$\max[E_{j_{x+1}} + D_{j_x},\ 1 \le j_x \le j_{x+1};\ E_{j_x} + D_{j_x},\ 1 \le j_x \le j_x]$$

with

$$\max[E'_{j_{x+1}} + D'_{j_x},\ 1 \le j_x \le j_{x+1};\ E'_{j_x} + D'_{j_x},\ 1 \le j_x \le j_x]$$

It is observed that these terms no longer involve just the subscripts j_x and j_{x+1}, and hence the decision is not independent of what precedes the interchanged elements.

Special case when $\min[t_{j_x 1} \ge \max[t_{j_x 2}]$: Unlike in the previous case there are fewer terms to compare. Hence job j_x precedes job j_{x+1} if

$$\max[E_{j_{x+1}} + D_{j_x},\ E_{j_x} + D_{j_x}] < \max[E'_{j_{x+1}} + D'_{j_{x+1}},\ E'_{j_x} + D'_{j_x}] \quad (A.3)$$

In case of an equality ordering of the indifferent jobs is made consistent with the ordering given by definite inequalities. Then, by subtracting

$$\sum_{x=1}^{x+1} t_{j_x 1} - \sum_{x=1}^{x-1} t_{j_x 2} + \sum_{x=1}^{x+1} t_{j_x 2} - \sum_{x=1}^{x-1} t_{j_x 3}$$ from both sides of

incquality (A.3)

$$\max[-t_{j_x 2} - t_{j_x 3}, -t_{j_{x+1} 2} - t_{j_{x+1} 1}]$$

$$< \max[-t_{j_{x+1} 2} - t_{j_{x+1} 3}, -t_{j_x 2} - t_{j_x 1}]$$

or

$$\min [t_{j_x 1} + t_{j_x 2}, t_{j_{x+1} 3} + t_{j_{x+1} 2}]$$

$$< \min [t_{j_{x+1} 1} + t_{j_{x+1} 2}, t_{j_x 3} + t_{j_x 2}] \qquad (A.4)$$

Lemma 4: The inequality (A.4) is transitive.

Proof: This is the same as for lemma 2. The above results may be stated as the following theorem.

Theorem 2: An optimal sequence is given by the following rule. The job j_x precedes the job j_{x+1} if

$$\min [t_{j_x 1} + t_{j_x 2}, t_{j_{x+1} 3} + t_{j_{x+1} 2}]$$

$$< \min [t_{j_{x+1} 1} + t_{j_{x+1} 2}, t_{j_x 3} + t_{j_x 2}]$$

In case of an equality either job is permissible.

APPENDIX B

Computer Speeds

In this appendix the speeds of various computers for add, multiply and divide operations are given.* These computers have been used by researchers in their experimentations. The following table might help in comparing various techniques on the basis of computer time required.

*Taken from "The Reference Day Book 1967", The Library of Computer and Information Sciences, pp. 26-34.

Table B.1

Computer Speeds†

Technique	Investigator	Computer Model	Time per Operation Instruction Add	Multiply	Divide
Direct	Smith and Dudek	IBM 1620-II	140μ	1210μ	3230μ
Branch-and-Bound	Brown and Lomnicki	ICT 1301*			
Branch-and-Bound	Ignall and Schrage	CDC 1604	7.2μ	25.2-63.9μ	65.2μ
Branch-and-Bound	McMahon and Burton	CDC 3600	2μ	2.12-6.5μ	2.12-14.9μ
Decomposition	Ashour	IBM 7044	5μ	22.5-37.5μ	7.5-50μ

†The computer speed is in microsecond ($\mu=10^{-6}$ seconds).

*ICT 1301 is approximately half as fast as IBM 7044, see Ashour (6), pp. 107.

APPENDIX C

Notation

The notation used by various researchers is different. For reference, Table C-1 lists the various notation which has been used. It is hoped that the notation used throughout this report is more clear.

Table C.1

Notation Used by Researchers

	Johnson (63)	Smith & Dudek (105)	Brown & Lomnicki (20)	Ignall & Schrage (56)	McMahon & Burton (76)	Giffler & Thompson (39)	Brooks & White (19)	Hardgrave & Nemhauser (46)
Machine Designation	A,B	A,B	M_j	A,B	A,B	F_j	f_i	m
Job Designation	i	j	J_i	i	i	c_i	c_i	i
Number of Machines		M	m	M		n	M	M
Number of Jobs	n	N	n	n	n	m	N	N
Operation Designation						C_iF_j		
Starting Time								
Processing Time	A_i,B_i	A_j,B_j	a_i,b_i	a_i,b_i	a_i,b_i	t_i	f_{ij}	p_{im}
Idle Time	X_i	IB_j						
Schedule Time		T						
Sequence of Machines for Job						C_{nx1}		
Sequence of Jobs Through Machines	S	S				F_{mx1}	S_{mn}	
Total Sequence								

Table C.1 (continued)

	Bowman (18)	Wagner (111)	Manne (75)	Szwarc (109)	Heller (49)	Sisson (102)	Mitten (80)	This Report
Machine Designation	A,B	k		j	m	m	I,II	m
Job Designation	x,y	i	j	i	j	j	i	j
Number of Machines		m		m	M	M		M
Number of Jobs		n	n	n	J	J	n	J
Operation Designation	$x_{A:t}$	$x_{ij}^{(k)}$			mj_k	mj_k		$j_x m_y$
Starting Time		$h_{ij}^{(k)}$	x_j			h_{mj}	C	S_{jm}
Processing Time		$T_{ij}^{(k)}$	a_j	t_{ij}	t_{mj}	t_{mj}	A_i, B_i	t_{jm}
Idle Time								I_{jm}
Schedule Time	T	h*		L	t(p)	T		T
Sequence of Machines for Job					$\mathfrak{J}_j$	J_j		M_j^*
Sequence of Jobs Through Machines					$\mathfrak{M}_m$	$\mathfrak{M}_m$		S_m^*
Total Sequence					S_{JM}	S	S	S

APPENDIX D

Glossary of Terms

Terms	Definition	Equivalent Terms
Machine		Facility, equipment
Job		Item, part, commodity, lot
Flow Shop	Type of problem where the jobs have the same machine ordering	without passing
Job Shop	Type of problem where job has a different machine ordering	with passing
Processing time matrix	A matrix which gives the processing times of jobs on various machines	
Machine ordering matrix	A matrix which gives the required machine ordering for all jobs	Routing, technological ordering
Job sequencing matrix	A matrix which gives sequence of jobs through various machines	Sequence, schedule, make span
Operation	a job is processed on a machine	Task

APPENDIX D (continued)

Terms	Definition	Equivalent Terms
Sequencing	Required machine ordering of operations on each job	Routing, technological ordering
Schedule time	Time taken to complete all jobs on all machines	Elapsed time, work duration
Node	Element of scheduling tree which consists of a partial on complete sequence	Vertex, tuple
Gantt-chart	Geometric representation of a sequence	Program, sequence, schedule
Numeric	Problems with numerical processing times	
Non-numeric	Problems without numerical processing times or given one unit to each processing time	
Active feasible	A feasible schedule where no machine is idle for length of time sufficient to process an idle job completely, and each operation starts as soon as machine and job are available	
Tardiness	Lateness	
Earliness	Negative lateness	

CLASSIFIED BIBLIOGRAPHY

Combinatorial Analysis

(3), (5), (9), (13), (15), (19), (20), (29), (30), (35), (39), (40), (49), (50), (56), (57), (58), (59), (63), (64), (65), (71), (73), (74), (76), (79), (80), (84), (87), (92), (105), (106), (113).

Integer-Linear Programing

(18), (22), (27), (28), (42), (43), (44), (45), (66), (67), (75), (108), (110), (111).

Dynamic Programing and Graphical-Dynamic Programing

(13), (14), (47), (70), (89), (90), (96), (109).

Graphical

(4), (46), (99).

Decomposition

(6), (7).

Schedule Algebras and Graph-Theoretic

(36), (38), (51), (52), (53), (82).

Priorities, Heuristics and Simulation

(1), (8), (10), (21), (24), (25), (31), (33), (34), (37), (41), (48), (51), (55), (60), (61), (62), (68), (69), (72), (77), (88), (93), (98), (100), (101), (102).

General

(2), (11), (12), (16), (23), (26), (32), (54), (78), (82), (91), (99), (103), (104), (107), (112).

BIBLIOGRAPHY

1. Ackerman, S., "Even-Flow. A Scheduling Method for Reducing Lateness in Job Shops", Management Technology, Vol. 3, No. 1, May 1963, pp. 20-32.

2. Ackoff, R., Arnoff, E. and Churchman, C., Progress in Operations Research, John Wiley & Sons, New York, 1961.

3. Agin, N., "Optimum Seeking by Branch-and-Bound", Management Science, Vol. 13, Dec. 1966, pp. 176-185.

4. Akers, S., "A Graphical Approach to Production Scheduling Problem", Operations Research, Vol. 4, Mar. 1956, pp. 244-245.

5. _____ and Friedman, J., "A Non-Numerical Approach to Production Scheduling Problem", Operations Research, Vol. 3, Nov. 1955, pp. 429-442.

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

7. _____, "A Decomposition Approach for the Machine Scheduling Problem", The International Journal of Production Research, Vol. 6, No. 2, 1967.

8. Baker, C. and Dzielinski, B., "Simulation of a Simplified Job Shop", Management Science, Vol. 6, Jan. 1960, pp. 311-323.

9. Banerjee, B., "Single Facility Sequencing with Random Execution Times", Operations Research, Vol. 13, May-June 1965, pp. 358-364.

10. Barnes, W., "The Application of Computer Simulation to Production Scheduling Research", 16th National Meeting of the Operations Research Society of America, 1962.

11. Beenhakker, H., "Development of Alternate Criteria for Optimality in the Machine Sequencing Problem", Ph.D. Thesis, Purdue University.

12. Bellman, R., "Some Mathematical Aspects of Scheduling Theory", Journal of the Society for Industrial and Applied Mathematics, Vol. 4, Sept. 1956, pp. 168-205.

13. _____, "Combinatorial Processes and Dynamic Programming", RAND P-1284, Feb. 1958.

14. _____ and Dreyfus, S., Applied Dynamic Programming, Princeton University, Princeton, New Jersey, 1962.

15. _____ and Gross, O., "Some Combinatorial Problems Arising in the Theory of Multi Stage Processes", Journal of the Society for Industrial and Applied Mathematics, Vol. 2, Sept. 1954, pp. 175-183.

16. Beyer, W., Handbook of Tables for Probability and Statistics, The Chemical Rubber Company, Cleveland, Ohio, 1966.

17. Blake, K. and Stopakis, W., "Some Theoretical Results on the Job Shop Scheduling Problem", Report M-1533-1, United Aircraft Corp. Research Dept., East Hartford, Conn., July, 1959.

18. Bowman, E., "The Scheduling-Sequencing Problem", Operations Research, Vol. 7, Sept.-Oct. 1959, pp. 621-624.

19. Brooks, G. and White, C., "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem", The Journal of Industrial Engineering, Vol. 16, Jan. 1965, pp. 34-40.

20. Brown, A. and Lomnicki, Z., "Some Applications of the Branch-and-Bound Algorithm to the Machine Scheduling Problem", Operational Research Quarterly, Vol. 17, June 1966, pp. 173-186.

21. Burstall, R., "A Heuristic Method for a Job-Scheduling Problem", Operational Research Quarterly, Vol. 17, Sept. 1966, pp. 291-304.

22. Carlson, R. and Nemhauser, G., "Scheduling to Minimize Interaction Cost", Operations Research, Vol. 14, Jan.-Feb. 1966, pp. 52-58.

23. Churchman, C., Ackoff, R. and Arnoff, E., Introduction to Operations Research, John Wiley & Sons, New York, 1961.

24. Conway, R., "An Experimental Investigation of Priority Dispatching", The Journal of Industrial Engineering, Vol. 11, June 1960, pp. 221-230.

25. _____, Johnson, B. and Maxwell, W., "A Queue Network Simulator for the IBM 650 and Burroughs 220," Comm. Assn. for Computing Machinery, Vol. 2, No. 12, Dec. 1959.

26. _____, Maxwell, W. and Miller, L., Theory of Scheduling, Addison-Wesley Publishing Company, 1967.

27. Dantzig, G., "A Machine Shop Scheduling Model", Management Science, Vol. 6, Jan. 1960, pp. 191-196.

28. _____, "On the Shortest Route Through a Network", Management Science, Vol. 6, Jan. 1960, pp. 187-190.

29. Dudek, R. and Ghare, P., "Make-Span Sequencing on M-Machines", The Journal of Industrial Engineering, Vol. 18, Jan. 1967, pp. 131-134.

30. _____ and Teuton, Jr., "Development of M-Stage Decision Rule for Scheduling n Jobs Through M Machines", Operations Research, Vol. 12, May-June 1964, pp. 471-497.

31. Eilon, S. and Hodgson, R., "Job Shops Scheduling With Due Dates", The International Journal of Production Research, Vol. 6, No. 1, 1967.

32. Feller, W., An Introduction to Probability Theory and Its Application, Vol. 1, Second Edition, John Wiley & Sons, New York, 1962, Theorem 2, p. 35.

33. Fisher, H. and Thompson, G., "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules", Chapter 15 in reference 82,

34. Gere, W., "Heuristics in Job Shop Scheduling", Management Science, Vol. 13, Nov. 1966, pp. 167-190.

35. Giffler, B., "Mathematical Solution of Explosion and Scheduling Problems", IBM Research Report RC-118, Yorktown Heights, New York, June 1959.

36. _____, "Mathematical Solution of Production Planning and Scheduling Problems, IBM ASDD Technical Report 09.026, White Plains, New York, Oct. 1960.

37. _____, SIMPRO 1: An IBM 704-7090 "Simulation Program for Planning Scheduling and Monitoring Production Systems", IBM ASDD Technical Report, Dec. 1961.

38. _____, "Schedule Algebras and Their Use in Formulating General Systems Simulation", Chapter 4 in reference 82.

39. _____ and Thompson, G., "Algorithms for Solving Production Scheduling Problems", IBM Research Report RC-118, Yorktown Heights, New York, June 1959.

40. _____ and _____, "Algorithms for Solving Production Scheduling Problems", Operations Research, Vol. 8, July-Aug. 1960, pp. 487-503.

41. _____, _____ and Van Ness, V., "Numerical Experience with the Linear and Monte Carlo Algorithms for Solving Production Scheduling Problems", Chapter 3 in reference 82.

42. Giglio, R. and Wagner, H., "Approximate Solutions to the Three-Machine Scheduling Problems", Operations Research, Vol. 12, Mar.-Apr. 1964, pp. 305-324.

43. Gomory, R., "An Algorithm for Integer Solutions to Linear Programs", Princeton-IBM Mathematics Research Project, Technical Report No. 1, Nov. 17, 1958.

44. _____, "All-Integer Integer Programming Algorithm", Chapter 13 in reference 71.

45. _____, "Mixed Integer Programming Algorithm", Rand Report P-1885, June 23, 1960.

46. Hardgrave, W. and Nemhauser, G., "A Geometric Model and a Graphical Algorithm for a Sequencing Problem", Operations Research, Vol. 11, Nov.-Dec. 1963, pp. 889-900.

47. Held, M. and Karp, R., "A Dynamic Programming Approach to Sequencing Problems", Journal of the Society for Industrial and Applied Mathematics, Vol. 10, Mar. 1962, pp. 196-210.

48. _____, _____ and Shareshian, R., "Scheduling with Arbitrary Profit Functions", Data Systems Division, Mathematics and Application Dept., IBM Corporation, New York.

49. Heller, J., "Combinatorial, Probabilistic and Statistical Aspects of an M x J Scheduling Problem", Report NYO-2540, Atomic Energy Commission Computed and Applied Mathematics Center, Institute of Mathematical Science, New York University, New York, Feb. 1959.

50. _____, "Combinatorial Properties of Machine Shop Scheduling", Report NYO-2879, Atomic Energy Commission Computed and Applied Mathematics Center, Institute of Mathematical Science, New York University, New York, July 1959.

51. _____, "Some Numerical Experiments for an M x J Flow Shop and Its Decision-Theoretical Aspects", Operations Research, Vol. 8, Mar.-Apr. 1960, pp. 178-184.

52. _____, "Some Problems in Linear Graph Theory that Arise in the Analysis of the Sequencing of Jobs through Machines", Report NYO-9847, Atomic Energy Commission Computed and Applied Mathematics Center, Institute of Mathematical Science, New York University, New York, Oct. 1960.

53. _____ and Logemann, G., "An Algorithm for the Construction and Evaluation of Possible Schedules", Management Science, Vol. 8, Jan. 1962, pp. 168-183.

54. Hogg, R. and Craig, A., Introduction to Mathematical Statistics, Second Edition, The Macmillan Company, New York, 1965, p. 180.

55. IBM Group, "The Job Simulator: An IBM 704 Program", IBM Mathematics and Application Dept., IBM Corporation, New York, 1960.

56. Ignall, E. and Schrage, L., "Application of the Branch and Bound Technique to Some Flow Shop Scheduling Problems", Operations Research, Vol. 13, May-June 1965, pp. 400-412.

57. Jackson, J., "Notes on Some Scheduling Problems", Management Sciences Research Project, Research Report No. 35, University of California, Los Angeles, Calif., Oct. 1954.

58. _____, "Scheduling a Production Line to Minimize Maximum Tardiness", Management Science Research Project, Research Report No. 43, University of California, Los Angeles, Calif., 1955.

59. _____, "An Extension of Johnson's Results on Job Lot Scheduling", Naval Research Logistics Quarterly, Vol. 3, Sept. 1956, pp. 201-203.

60. _____, "Simulation Research on Job Shop Production", Naval Research Logistics Quarterly, Vol. 4, Dec. 1957, pp. 287-295.

61. _____, "Machine Shop Simulation Using SWAC: A Progress Report", Management SCiences Research Project, Discussion Paper No. 67, University of California, Los Angeles, Apr. 1958.

62. _____ and Kurantani, Y., "Production Scheduling Research : A Monte Carlo Approach", Management Sciences Research Project, Research Paper No. 61, University of California, Los Angeles, Calif., May 1957.

63. Johnson, S., "Optimal Two- and Three-Stage Production Schedules with Set-Up Times Included", Naval Research Logistics Quarterly, Vol. 1, Mar. 1954, pp. 61-68.

64. _____, "Discussion", Management Science, Vol. 5, Apr. 1959, pp. 299-303.

65. Karush, W., "A Counter-Example to a Proposed Algorithm for Optimal Sequencing of Jobs", Operations Research, Vol. 13, Mar.-Apr. 1965, pp. 323-325.

66. _____ and Moody, L., "Determination of Feasible Shipping Schedules for a Job Shop", Operations Research, Vol. 6, Jan.-Feb. 1958, pp. 35-55.

67. _____ and Vazsonui, A., "Mathematical Programming and Service Scheduling", Management Science, Vol. 3, Jan. 1957, pp. 140-148.

68. Kurantani, Y. and Nelson, R., "A Pre-Computational Report on Job-Shop Simulation Research", Journal of the Operations Research Society of Japan, Vol. 2, Mar. 1960, pp. 145-183.

69. _____ and McKenny, J., "A Preliminary Report on Job Shop Simulations Research", Management Sciences Research Project, Research Report No. 65, University of California, Los Angeles, Calif., Mar. 1958.

70. Lawler, E., "On Scheduling Problems with Deferral Costs," Management Science, Vol. 11, No. 2, Nov. 1966, pp. 280-288.

71. _____ and Wood, D., "Branch-and-Bound Methods: A Survey", Operations Research, Vol. 14, July-Aug. 1966, pp. 699-719.

72. LeGrande, Earl, "The Development of a Factory Simulation System Using Actual Operating Data", Management Technology, Vol. 3, No. 1, May 1963, pp. 1-19.

73. Little, J., Murty, K., Swecney, D. and Karel, C., "An Algorithm for the Traveling Salesman Problem", Operations Research, Vol. 11, Nov.-Dec. 1963, pp. 972-989.

74. Lomnicki, Z., "A Branch-and-Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem", Operational Research Quarterly, Vol. 16, Mar. 1965, pp. 89-100.

75. Manne, A., "On the Job-Shop Scheduling Problem", Operations Research, Vol. 8, Mar.-Apr. 1960, pp. 219-223.

76. McMahon, G. and Burton, P., "Flowshop Scheduling with Branch-and-Bound Method", Operations Research, Vol. 15, May-June 1967, pp. 473-481.

77. McNaughton, R., "Scheduling with Deadlines and Loss Functions", Management Science, Vol. 5, Jan. 1959, pp. 1-12.

78. Mellor, P., "A Review of Job Shop Scheduling", Operational Research Quarterly, Vol. 17, June 1966, pp. 161-171.

79. Mitten, L., "Sequencing n Jobs on Two Machines with Arbitrary Time Lags", Management Science, Vol. 5, Apr. 1959, pp. 293-298.

80. _____, "A Scheduling Problem", The Journal of Industrial Engineering, Vol. 10, Mar. 1959, pp. 131-135.

81. Muth, J., "The Effect of Uncertainty in Job Times on Optimal Schedules", Chapter 18 in reference 82.

82. _____ and Thompson, G., Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 1963.

83. Nelson, R., "Priority Functions Methods for Job-Lot Scheduling", Management Sciences Research Project, UCLA, Discussion Paper No. 51, Feb. 24, 1955.

84. Norman, A., "Optimum Seeking with Branch and Bound", Management Science, Vol. 13, Dec. 1966, pp. 176-185.

85. Oldfather, P., Ginsberg, A. and Markowitz, H., Programming by Questionaire, IM 5129PR and IM 4460PR, The Rand Corporation, Santa Monica, Calif., 1966.

86. Page, E., "An Approach to Scheduling Jobs on Machines", Journal of Royal Statistics Society B, Vol. 23, 1961, pp. 484-492.

87. Palmer, D., "Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time - A Quiek Method of Obtaining a Near Optimum", Operational Researeh Quarterly, Vol. 16, Mar. 1965, pp. 101-107.

88. Presby, J. and Wolfson, M., "An Algorithm for Solving Job Sequeneing Problems", Management Seienee, Vol. 13, No. 8, April, 1967, pp. 454-464.

89. Reinitz, R., "An Integrated Job Shop Seheduling Problem", Ph.D. Thesis, Case Institute of Technology, Cleveland, Ohio, 1961.

90. _____, "On the Job-Shop Seheduling Problem", Chapter 5 in referenee 82.

91. Root, J., "Seheduling with Dead Lines and Loss Funetions on k Parallel Machines", Management Scienee, Vol. 11, No. 3, Jan. 1965, pp. 460-475.

92. Roy, B., "Cheminement et Connexite dans les graphes - Applieations aux problemes d'ordonnaneement", METRA, Serie Speeiale No. 1, 1962, Soeiete d'economie et de mathematiques appliquees, Paris.

93. Rowe, A., "Toward a Theory of Scheduling", The Journal of Industrial Engineering, Vol. 11, Mar. 1960, pp. 125-136.

94. _____, "Sequential Decision Rules in Production Seheduling", General Electric Company, Oet. 1958.

95. _____ and Jaekson, J., "Research Problems in Production Routing and Seheduling", Researeh Report No. 46, University of California, Los Angeles, Calif., Oet. 1956.

96. Salah, E., "The One Maehine Sequeneing Problem with Delay Costs", The Journal of Industrial Engineering, Vol. 19, Feb. 1968, pp. 105-108.

97. Salveson, M., "A Problem in Optimal Maehine Loading", Management Seienee, Vol. 2, Apr. 1956, pp. 232-260.

98. _____, "A Computational Teehnique for the Seheduling Problem", The Journal of Industrial Engineering, Vol. 13, Jan. 1962, pp. 30-41.

99. Sasieni, M., Yaspan, A. and Friedman, L., Operations Researeh: Methods and Problems, John Wiley & Sons, New York, 1959.

100. Sherman, G., "The Use of a Computer for Scheduling Students", Speech before the annual meeting of the Operations Research Society of America, Washington, D.C., May 14-15, 1959.

101. Simon, H. and Newell, A., "Heuristic Problem Solving: The Next Advance in Operations Research", Operations Research, Vol. 6, No. 1, Jan.-Feb. 1958, pp. 1-10.

102. Sisson, R., "Machine Shop Simulation Using SWAC: Part II of a Proposal", Management Sciences Research Project, Research Paper No. 58, May 1956.

103. _____, "Sequencing Theory", Chapter 7 in reference 2.

104. _____, "Method of Sequencing in Job-Shop - A Review", Operations Research, Vol. 7, Jan.-Feb. 1959, pp. 10-29.

105. Smith, R. and Dudek, R., "A General Algorithm for Solution of the n-Job, M-Machine Sequencing Problem of the Flow Shop", Operations Research, Vol. 15, Jan.-Feb. 1967, pp. 71-82.

106. Smith, W., "Various Optimizers for Single-Stage Production", Naval Research Logistics Quarterly, Vol. 3, March and June 1956, pp. 59-66.

107. _____, "Application of a Posteriori Probability", Management Sciences Research Project, Research Report No. 56, University of California, Los Angeles, Calif., Sept. 1958.

108. Story, A. and Wagner, H., "Computational Experience with Integer Programming for Job-Shop Scheduling", Chapter 14 in reference 82.

109. Szware, W., "Solution of the Akers-Friedman Scheduling Problem", Operations Research, Vol. 8, Nov.-Dec. 1960, pp. 782-788.

110. Thompson, G., "Recent Development in the Job Shop Scheduling Problem", Naval Research Logistics Quarterly, Vol. 7, Dec. 1960, pp. 585-589.

111. Wagner, H., "An Integer Linear Programming Model for Machine Scheduling", Naval Research Logistics Quarterly, Vol. 6, June 1959, pp. 131-140.

112. Wald, A. and Wolfowitz, J., "An Exact Test for Randomness in the Non-Parametric Case Based on Serial Correlation", Annals of Mathematical Statistics, Vol. 14, 1943, pp. 378-388.

113. White, C., "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem", Ph.D. Thesis, Purdue University, Lafayette, Indiana, 1963.

THE MACHINE SCHEDULING PROBLEM

by

ARUNKUMAR VASANTLAL PAREKH

B. E. (Mechanical), Sardar Vallabhbhai Vidyapeeth, India, 1965

AN ABSTRACT OF 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

1968

The machine scheduling problem involves the scheduling of J jobs on M machines such that a criterion is optimized. The criterion considered in this report is the minimization of the schedule time. The number of possible sequences is very large even, for small size problems. Thus, the solution by complete enumeration is not practical.

Several algorithms have been developed to solve this problem. The objective of these algorithms have been to confine the search to a subset of the complete set of feasible sequences and then evaluating them to select the sequence(s) which minimize the schedule time.

A flow shop problem of six jobs and two machines is solved by Direct Technique. Another flow shop problem having six jobs and three machines is solved by Extended Direct, Branch-and-Bound, and Lower Bound Techniques. This problem is also solved by Direct Technique, but after the processing times are changed to meet the specific restrictions imposed in this technique. Further, a job shop problem of two jobs and three machines is solved by Boolean Algebra, Graphical, and Graphical-Dynamic Programing Techniques. Three Integer-Linear Programing formulations of a sample flow shop problem are presented but none are solved, since it requires a prohibitive amount of computer time.

In solving the above sample problems, it is observed that these techniques are only feasible for very small size problems. Computational difficulties are involved in solving large size

problems. In using the above techniques, it is observed that the optimality can be attained but with excessive amount of computation time.