4

Flow Shop Scheduling

CHAPTER CONTENTS

4.1 INTRODUCTION

A flow shop problem exists when all the jobs share the same processing order on all the machines. In flow shop, the technological constraints demand that the jobs pass between the machines in the same order. Hence, there is a natural processing order (sequence) of the machines characterized by the technological constraints for each and every job in flow shop. Frequently occurring practical scheduling problems focus on two important decisions:

- The sequential ordering of the jobs that will be processed serially by two or more machines
- The machine loading schedule which identifies the sequential arrangement of start and finish times on each machine for various jobs.

Managers usually prefer job sequence and associated machine loading schedules that permit total facility processing time, mean flow time, average tardiness, and average lateness to be minimized. The flow shop contains **m** different machines arranged in series on which a set of **n** jobs are to be processed. Each of the **n** jobs requires **m** operations and each operation is to be performed on a separate machine. The flow of the work is unidirectional; thus every job must be processed through each machine in a given prescribed order. In other words, if machines are numbered from *1,2,3....m*, then operations of job ***j*** will correspondingly be numbered (1,j), (2,j), (3,j),.... (m,j). In this context, each job has been assigned exactly **m** operations where as in real situations a job may have a fewer operations. Nevertheless, such a job will still be treated as processing **m** operations but with zero processing times correspondingly.

The general **n** jobs, **m** machine flow shop scheduling problem is quite formidable. Considering an arbitrary sequence of jobs on each machine, there are $\mathbf{(n!)^m}$ possible schedules which poses computational difficulties. Therefore, efforts in the past have been made by researchers to reduce this number of feasible schedules as much as possible without compromising on optimality condition. Literature on flow shop process indicates that it is not sufficient to consider only schedules in which the same job sequence occurs on each machine with a view to achieving optimality. On the other hand, it is not always essential to consider $\mathbf{(n!)^m}$ schedules in search for an optimum. The following two dominance properties will indicate the amount of reduction possible in flow shop problems.

Theorem 1

When scheduling to optimize any regular measure of performance in a static deterministic flow shop, it is sufficient to consider only those schedules in which the same job sequence exists on machine 1 and machine 2.

Theorem 2

When scheduling to optimize makespan in the static deterministic flow shop, it is sufficient to consider only those schedules in which the same

job sequence exists on machine 1 and 2, and the same job sequence on machines **m**-1 and **m**.

The implications of above dominance can be interpreted as follows:

- For any regular measure of performance, by virtue of the fact that the same job sequence on the first two machines is sufficient for achieving optimization, it is $\mathbf{(n!)^{m-1}}$ schedules that constitute a dominant set.
- For makespan problems, by virtue of the fact that the same job sequence on machine **m**-1 and **m** besides on machine 1 and 2 is sufficient for achieving optimization, it is $\mathbf{(n!)^{m-2}}$ schedules that constitute a dominant set for $\mathbf{m} > 2$.

  As defined earlier, a permutation schedule is that class of schedule which may be completely identified by single permutation of integers. For a flow shop process, the permutation schedule is therefore, a schedule with the same job sequence on all machines. Interpreting the above results yet in another way, it is observed that:
- In a two machine flow shop, permutation schedule is the optimum schedule with regard to any regular measure of performance.
- In a three machine flow shop, permutation schedule is the optimum schedule with respect to makespan criterion.

Unfortunately, this second dominance property is confined to makespan only. This neither extends to other measures of performance nor does it take into account large flow shops with $m > 3$. However, no stronger general results than the above two concerning permutation schedules are available in the literature.

4.2 MINIMIZATION OF MAKESPAN USING JOHNSON'S RULE FOR ($F_2 \parallel C_{max}$) PROBLEM

The flow shop contains **n** jobs simultaneously available at time zero and to be processed by two machines arranged in series with unlimited storage in between them. The processing times of all jobs are known with certainty. It is required to schedule the **n** jobs on the machines so as to minimize makespan (C_{max}). This problem is solved by Johnson's non-preemptive rule for optimizing the makespan in the general two machine static flow shop. This is the most important result for the flow shop problem which has now become a standard in theory of scheduling. The Johnson's rule for scheduling jobs in two machine flow shop is given below:

In an optimal schedule, job ***i*** precedes job ***j*** if:

$$\min \{p_{i1} , p_{j2}\} < \min \{p_{j1} , p_{i2}\}$$

Where as,

p_{i1} is the processing time of job ***i*** on machine 1 and p_{i2} is the processing time of job i on machine 2. Similarly, p_{j1} and p_{j2} are processing times of job j on machine 1 and 2 respectively.

The steps of Johnson's algorithm for constructing an optimal schedule may be summarized as follows:

Let,

p_{1j} = processing time of job j on machine 1.

p_{2j} = processing time of job j on machine 2.

Johnson's Algorithm

Step 1: Form set-I containing all the jobs with $p_{1j} < p_{2j}$

Step 2: Form set-II containing all the jobs with $p_{1j} > p_{2j}$

The jobs with $p_{1j} = p_{2j}$ may be put in either set.

Step 3: Form the sequence as follows:

a) The jobs in set-I go first in the sequence and they go in increasing order of p_{1j} (SPT)

b) The jobs in set-II follow in decreasing order of p_{2j} (LPT). Ties are broken arbitrarily.

This type of schedule is referred to as SPT (1)-LPT (2) schedule.

Example 4.1

Consider the following data presents an instance of $F_2 \parallel C_{max}$ problem. Find optimal value of makespan using Johnson's rule.

Job (j)	j_1	j_2	j_3	j_4	j_5
p_{1j}	5	2	3	6	7
p_{2j}	1	4	3	5	2

Solution:

Step 1:

Of all jobs; $1 \le j \le 5$, only job j_2 has $p_{1j} < p_{2j}$ which belong to Set-I = { j_2 }

Step 2:

Jobs j_1, j_4 and j_5 have $p_{1j} > p_{2j}$ which belong to Set-II = { j_1, j_4 , j_5 }

Job j_3 has $p_{1j} = p_{2j}$, so put it in any set; say set-I. Set-I = { j_2 , j_3 }

Step 3:

a) Arrange sequence of jobs in set-I according to SPT. Set-I contains j_2 and j_3 as members. Process time of job 2 on machine M_1 is p_{12}=2. Similarly process time of job 3 on machine M_1 is p_{13}=3. Sequencing jobs j_2 and j_3 according to SPT;

Set-I = { j_2 , j_3 }

b) Arrange sequence of jobs in set-II according to LPT. Process times of jobs in set-II are; p_{21} = 1, p_{24} = 4 and, p_{25} = 2. Hence, revised sequence is;

Set-II = { j_4, j_5, j_1 }

Optimal sequence; Set-I + Set-II = { j_2, j_3, j_4, j_5, j_1}

The schedule for the optimal sequence is presented in graphical form using directed graph and Gantt chart. Directed graph also presents the critical path. All the processes on machine M_1 are on critical path. Gantt chart shows idle times on machine M_2.

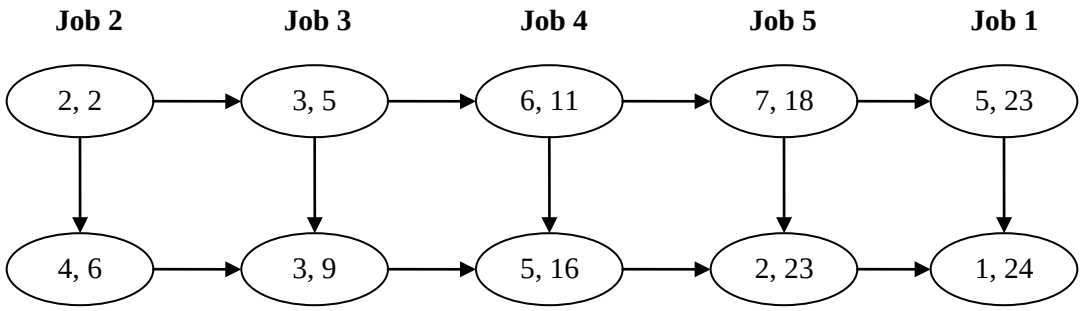

Figure 4.1 Directed Graph For Optimal Sequence { j_2, j_3, j_4, j_5, j_1}

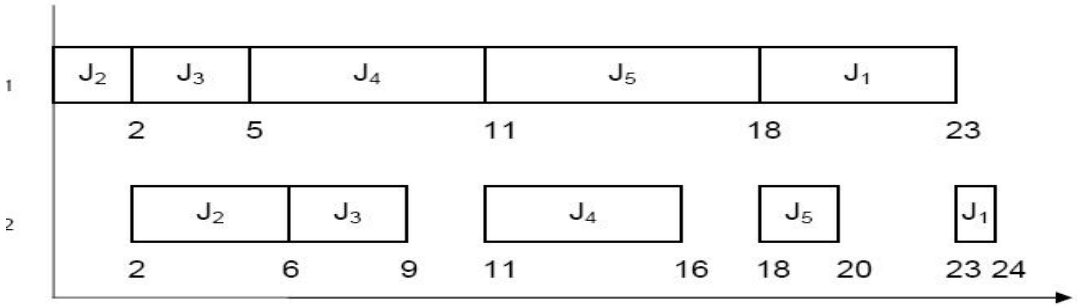

Figure 4.2 Gantt chart For optimal sequence { j_2, j_3, j_4, j_5, j_1}

4.3 MINIMIZATION OF MAKESPAN FOR (F_3 || C_{max}) PROBLEM

This is the same flow shop problem as defined in two-machine case except that now there are three machines arranged in series for processing of n jobs in a prescribed order. Also, by virtue of dominance property number 2 (Theorem 2), permutation schedule still constitutes a dominant set for optimizing makespan. However, Johnson's 2-machine algorithm can not be extended to general 3-machine flow shop. Nevertheless, under special conditions, generalization is possible. In this regard, if either of the following condition satisfies, the Johnson's 2 machine algorithm may be extended to the 3-machine flow shop to achieve optimum makespan.

Either,

$$\min\left(p_{1j}\right) \geq \max\left(p_{2j}\right)$$

or

$$\min(p_{3j}) \geq \max(p_{2j})$$

In other words, machine 2 is completely dominated by either the first or the third machine so that no bottleneck could possibly occur on the second machine. Subject to the above conditions, the optimal scheduling rule is applicable to 3-machine flow shop

The working procedure is the same as that described in the two machines case except that the three machines flow shop is reduced to two dummy machine M_1' and M_2' such that processing times of job *j* on machines M_1' and M_2' are $(p_{j1} + p_{j2})$ and $(p_{j2} + p_{j3})$ respectively. Johnson's algorithm is then applied to these two dummy machines to find the optimal job sequence.

Example 4.2

Consider an instance of the F3 || C_{max} problem in the following Table.

Job (j)	Process time (M1)	Process time (M2)	Process time (M3)
1	8	2	4
2	5	4	5
3	6	1	3
4	7	3	2

Find optimal sequence.

Solution:

Check for minimum value of process time on machines M_1 and M_3. These times are 5 and 2 respectively. Check maximum time on machine M_2 which is 4. Since min { p_{1j} } >= max { p_{2j}}, the problem can be converted to surrogate 2-machine problem. The problem data for two surrogate machines $M1'$ and $M2'$ is given in the following table.

Job (j)	Process time (M_1')	Process time (M_2')
1	10	6
2	9	9
3	7	4
4	10	5

Applying Johnson's Rule;

Set-I = { j_2 }, Set-II = { j_1, j_4, j_3 }

Optimal sequence = { j_2, j_1, j_4, j_3 }

Application of Johnson's algorithm to three machine flow shop problem has been tested by various authors. Burns and Rooker showed that under the conditions

p_{j2} > min (p_{i1}, p_{i3}) for each job ***j***=1,.....n, Johnson's algorithm produces optimal schedule. Jackson presented a case where all jobs use a common first and third machine for operation one and three respectively, but for second operation; the machine differs with each job. Under the conditions he observed that Johnson's algorithm produces optimal makespan for the 3-machine flow shop problem.

For general flow shops where the condition min $\{p_{1j}\}$ >= max $\{p_{2j}\}$ or min $\{p_{3j}\}$ >= max $\{p_{2j}\}$ is relaxed, Johnson's algorithm does not necessarily produce optimum makespan. However, it does provide good starting schedule, which can be further improved towards optimality through employing various techniques. In this context, Giglio and Wagner tested the algorithm for the series of the problems whereby the average makespan of 20 different cases under Johnson's Rule came out to be the 131.7 as compared to 127.9 for the optimal schedules. Furthermore, in 9 cases the true optimal results were obtained and another 8 results could be made optimum by interchanging the sequence of two adjacent jobs. Therefore, apparently Johnson's algorithm seems to produce good starting solutions, which even if not optimal, possesses the potential of heading towards optimality with reduced efforts.

4.4 MINIMIZATION OF MAKESPAN FOR (F_m || C_{max}) PROBLEMS WHEN m > 3

This is a general flow shop scheduling problem where **n** jobs are to be scheduled on **m** machines with all the **n** jobs available for processing at time zero. Processing times are deterministic. The problem is an extension of the 3-machine flow shop but there are no efficient exact solution procedures known. The problem, looking very simple at the outset, transforms into very complex and formidable one as the number of the machines exceed 3. Reason being the inherent combinatorial aspects associated with the general flow shop scheduling. Except for Johnson's algorithm for the optimizing **makespan** criterion in a 2 machines static flow shop, no constructive algorithms exists that take account of optimizing other measures of performance or tackle the larger flow shops with regard to any measure of performance. The secret of this lack of success has been exposed through relatively recent finding that non-preemptive scheduling for flow shop problems is **NP-complete** with regard to minimizing makespan or the mean flow time (Garey et al. [1976]). Similarly with preemptive scheduling, Gonzalez and Sahni [1978] proved **NP-completeness** for the makespan criterion. Therefore, even after 36 years of the pioneering work of Johnson, no optimal strategies could be developed for flow shop with **m** > 3.

4.5 PERMUTATION SCHEDULES

Research on large flow shops puts great emphasis on permutation schedules. This is because of two reasons. First, the permutation schedules constitute a significantly smaller set containing **n!** sequences where as non permutation schedules consist of $\mathbf{(n!)^m}$ sequences. Second, despite the fact that for m > 3 permutation schedules are not dominant, it is not unreasonable to believe that the best permutation schedules, even if not necessarily optimal, can not be too far away from true optimum. Hence, the benefit gained through permutation schedule in terms of significant reduction in number of sequences in a large flow shop is of much more value than going for a possible true optimum solution at the expense of increased computational effort and money.

4.6 HEURISTICS FOR MINIMIZATION OF MAKESPAN ($F_m \,||\, C_{max}$) PROBLEMS

4.6.1 Palmer's Heuristic

This heuristic comprises two steps as follows.

Step 1: For n job and m machine static flow shop problem, compute slope A_j for j^{th} job as follows;

$$A_j = -\sum_{i=1}^{m} \{m - (2i-1)\} p_{ij}$$

Step 2: Order the jobs in the sequence based on descending (decreasing) order of A_j values.

Example 4.3

Solve $F_3 || C_{max}$ problem for the data shown in Table using Palmer's heuristic.

Machines	j_1	j_2	j_3	j_4
M_1	6	8	3	4
M_2	5	1	5	4
M_3	4	4	4	2

Solution

$$A_j = -\sum_{i=1}^{m}\{m-(2i-1)\}p_{ij} = -\sum_{i=1}^{3}\{3-(2i-1)\}p_{ij}\ ,$$

Table 4.1 Calculation for the job's slope.

	i=1	i=2	i=3	
	3-(2-1) = 2	3-(2x2-1) = 0	3-(2x3-1) = -2	
Job j	p_{1j}	p_{2j}	p_{3j}	A_j
1	6	5	4	-4
2	8	1	4	-8
3	3	5	4	2
4	4	4	2	-4

Arranging slope values in descending order; there are two sequences;

Sequence 1 = { j_3 , j_1, j_4 , j_2 } Sequence 2 = { j_3, j_4 , j_1 , j_2 }

Directed graph for sequence 1 is;

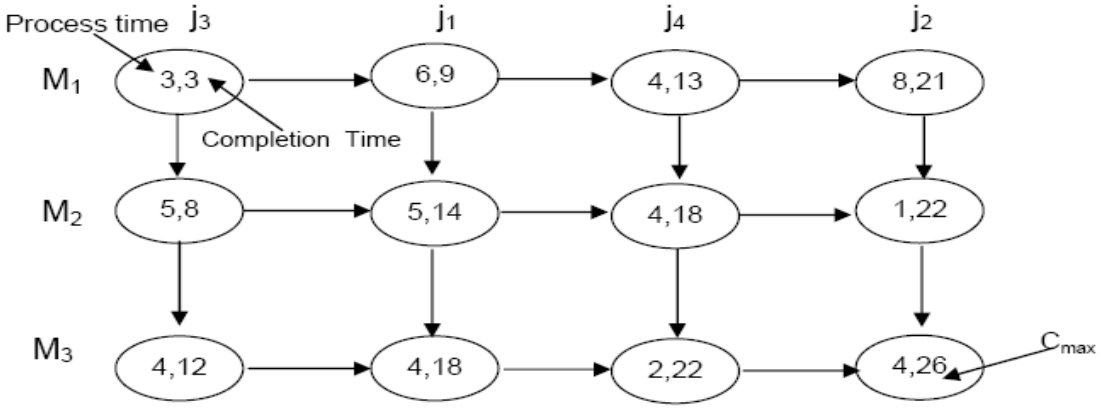

Figure 3.3 Directed graph for seq. { j_3 , j_1, j_4 , j_2 }

Directed graph for sequence 2 is;

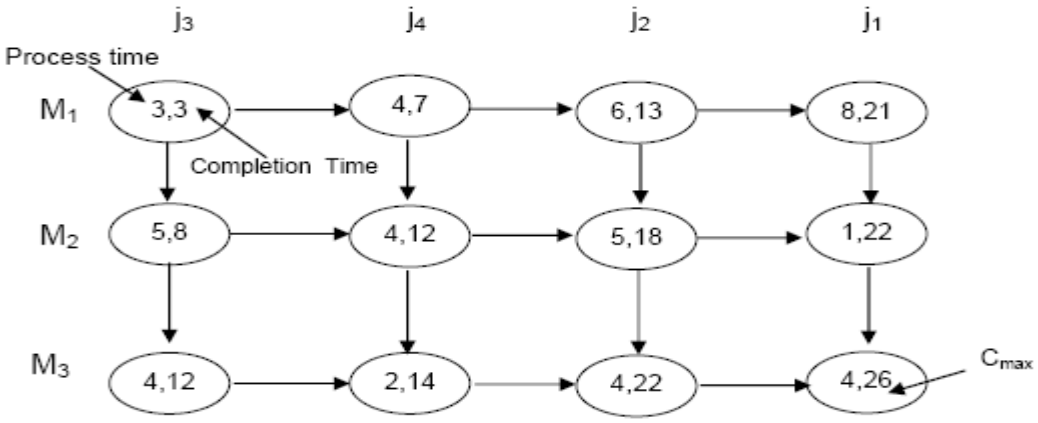

Figure 3.4 Directed graph for sequence { j_3 , j_4, j_1 , j_2 }

Conclusion: Note the C_{max}=26 for both sequences

4.6.2 Campbell, Dudek, and Smith (CDS) Algorithm

The algorithm converts a given n-job m-machine problem (m > 2) into p number of 2-machine n-job surrogate problems, where p = m-1. Each surrogate problem is solved using the Johnson rule. The value of C_{max} for each surrogate problem is found using Johnson rule. The sequence of the surrogate problem yielding minimum value of C_{max} after applying Johnson's rule is selected for scheduling jobs on the machines.

First step in CDS algorithm is to formulate surrogate problems from the original problem. Consider a 3-machine 4-job problem as below. The 3-machine will have two surrogate $F_2 \mid\mid C_{max}$ problems.

Table 4.2 Data for 3-machine 4-job Problem.

Jobs	**M_1**	**M_2**	**M_3**
j_1	P_{11}	P_{21}	P_{31}
j_2	P_{12}	P_{22}	P_{32}
j_3	P_{13}	P_{23}	P_{33}
j_4	P_{14}	P_{24}	P_{34}

4.6.2.1 First Surrogate Problem

In first $F_2 \mid\mid C_{max}$ surrogate problem, machine 1 data will comprise 1st column of original problem. Similarly, machine 2 data will comprise 3rd column of the original problem as shown under.

Table 4.4 Data for surrogate machines M_1' and M_2'

Jobs	$M_1' = M_1$	$M_2' = M_3$
j_1	P_{11}	P_{31}
j_2	P_{12}	P_{32}
j_3	P_{13}	P_{33}
j_4	P_{14}	P_{34}

4.6.2.2 Second Surrogate Problem

In second $F_2 \mid\mid C_{max}$ surrogate problem, machine M_1 data will comprise summation of 1st and 2nd columns of the original problem. Similarly, machine M_2 data will comprise summation of 2nd and 3rd columns of the original problem.

Table 4.5 Data for surrogate machines M_1' and M_2'

Jobs	$M_1' = M_1+M_2$	$M_2' = M_2+M_3$
j_1	$P_{11} + P_{21}$	$P_{21} + P_{31}$
j_2	$P_{12} + P_{22}$	$P_{22} + P_{32}$
j_3	$P_{13} + P_{23}$	$P_{23} + P_{33}$
j_4	$P_{14} + P_{24}$	$P_{24} + P_{34}$

This implies that the surrogate problems data will be in generated as follows:

For k = 1, ……, m-1 and j = 1, ……, n then,

$$M_1' = \sum_{i}^{k} P_{ij} \quad \text{and} \quad M_2' = \sum_{i=m-k+1}^{m} P_{ij}$$

Where:

M_1' = the processing time for the first machine

M_2' = the processing time for the second machine

Example 4.4

Solve $F_3 \mid\mid C_{max}$ problem for the data shown in Table using CDS heuristic.

	j_1	j_2	j_3	j_4
M_1	6	8	3	4
M_2	5	1	5	4
M_3	4	4	4	2

Solution:

Since there are three machines in the original problem, two (m-1 =2) surrogate $F_2 \mid\mid C_{max}$ problems will be formed.

i. Surrogate Problem 1

Consider Machine M_1 as surrogate machine 1 (M_1') and Machine M_3 as surrogate machine 2 (M_2') as shown in Table below.

Table 4.6 First 2-machine Surrogate problem data using CDS Heuristic.

	j_1	j_2	j_3	j_4
$M_1' = M_1$	6	8	3	4
$M_2' = M_3$	4	4	4	2

Applying Johnson Rule, Set I = $\{j_3\}$, set II = $\{j_1, j_2, j_4\}$ or set II = $\{j_2, j_1, j_4\}$. Hence there are two possible sequences:

Sequence 1 = $\{j_3, j_1, j_2, j_4\}$ and, is given in Table 4.7.

Table 4.7 First sequence obtained and job's processing times.

	j_3	j_1	j_2	j_4
$M_1' = M_1$	3	6	8	4
$M_2' = M_3$	4	4	4	2

Using directed graph, the C_{max} calculations are shown in Figure 3.5

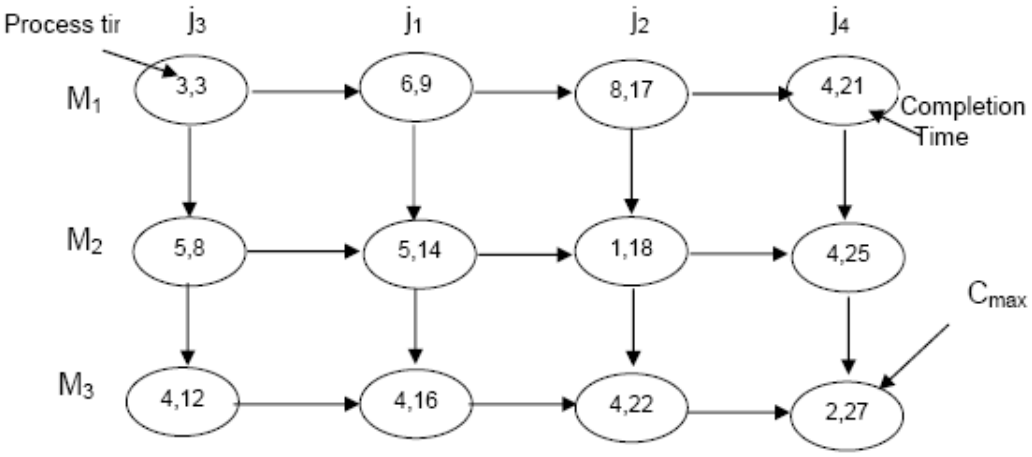

Figure 4.5 Directed Graph For Sequence/Schedule $\{j_3, j_1, j_2, j_4\}$

Sequence 2 = {j_3, j_2, j_1, j_4} and, is given in the Table 4.8.

Table 4.8 Second sequence obtained and job's processing time.

	J_3	j_2	j_1	j_4
$M_1' = M_1$	3	8	6	4
$M_2' = M_3$	4	4	4	2

Using directed graph, the C_{max} calculations are shown in Figure 3.6

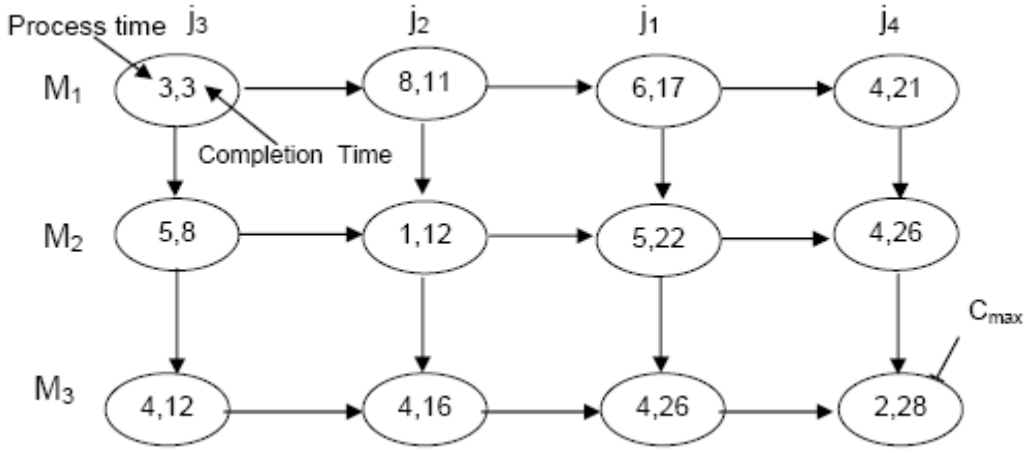

Figure 4.6 Directed graph for sequence/schedule {j_3, j_2, j_1, j_4}

ii. Surrogate Problem 2

From the problem data in Table, formulate 2-machine problem as under;

Table 4.9 Data for surrogate problem 2

	j_1	j_2	j_3	j_4
$M_1'=M_1+M_2$	11	9	8	8
$M_2'=M_2+M_3$	9	5	9	6

Applying Johnson rule; Set–I = {j_3}, and, Set-II = {j_1, j_4, j_2}. The Johnson sequence is, therefore, {j_3, j_1, j_4, j_2}. The computation of C_{max} is shown in Table 4.10

Table 4.10 C_{max} calculations using tabular method for sequence: {j_3, j_1, j_4, j_2}

Machine	j_3	j_1	j_4	j_2	C_3	C_1	C_4	C_2	C_{max}
M_1	3	6	4	8	3	9	13	21	
M_2	5	5	4	1	8	14	18	22	
M_3	4	4	2	4	12	18	20	26	26

The Gantt chart for schedule is shown in Figure 4.7

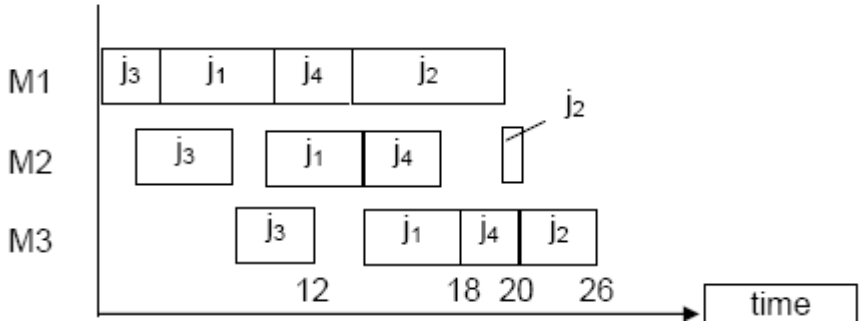

Figure 4.7 Gantt chart for sequence {j_3, j_1, j_4, j_2}.

The schedule {j_3 , j_1 , j_4 , j_2} is also presented by directed graph as shown in Figure 4.8

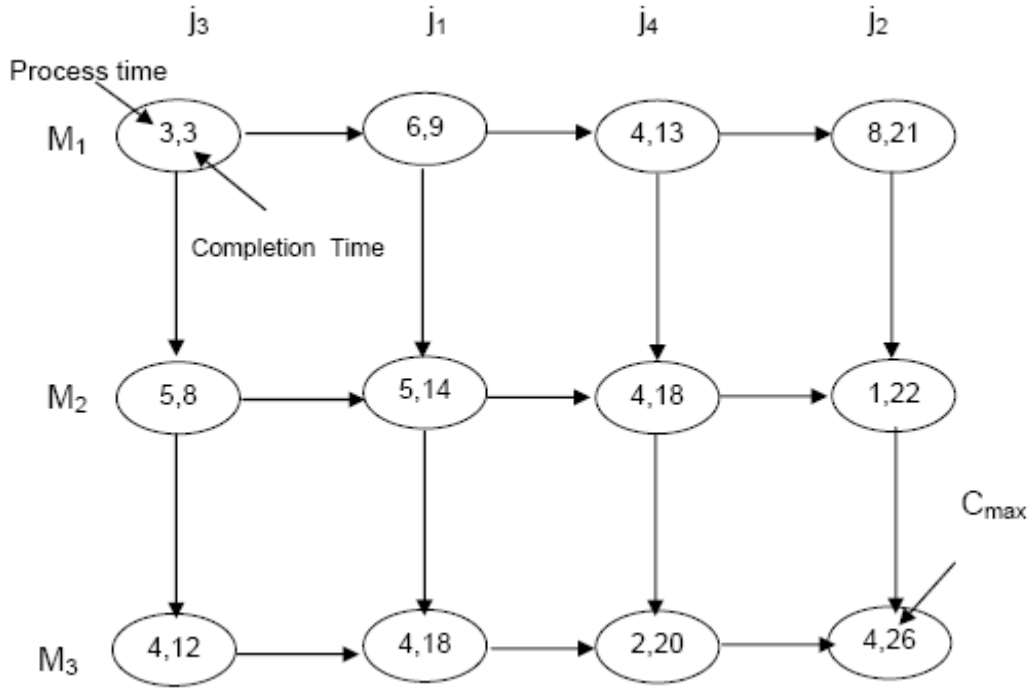

Figure 4.8 Directed graph for schedule {j_3, j_1, j_4, j_2}

Conclusion: Minimum C_{max} value is 26 using sequence: {j_3, j_1, j_4, j_2}

4.6.3 Nawaz, Enscor, and Ham (NEH) Algorithm

Nawaz, Enscor and Ham (NEH) algorithm constructs jobs sequence in iterative manner. Two jobs having largest values of total process times (called total work content) are arranged in a partial sequence one by one. The partial sequence having small value of C_{max} is selected for subsequent iteration. Then, next job from the work content list is picked. This job is alternately placed at all possible locations

in the partial sequence. This job is permanently placed at the location where it yields lowest C_{max} value for the partial schedule. In a similar fashion, next job from the work content list is picked, and placed one by one at all possible locations in the partial sequence to find C_{max} value of the partial sequence. This job is permanently placed at the location where partial sequence has minimum C_{max} value. The process is continued till all jobs from the content list are placed in the partial sequence.

NEH algorithm is formally described as under;

Step (1)

Find Total work content (T_j) for each job using expression

$$T_j = \sum_{i=1}^{i=m} p_{ij}$$

Step (2)

Arrange jobs in a work content list according to decreasing values of T_j

Step (3)

Select first two jobs from the list, and form two partial sequences by interchanging the place of the two jobs. Compute C_{max} values of the partial sequences. Out of the two partial sequences, discard the partial sequence having larger value of C_{max}. Call the partial sequence with lower value of C_{max} as *incumbent* sequence

Step (4)

Pick next job from the work content list, and place it at all locations in the *incumbent* sequence. Calculate the value of C_{max} for all the sequences.

Step (5)

Retain the sequence with minimum value of C_{max} as incumbent sequence and, discard all the other sequences.

Step (6)

If there is no job left in the work content list to be added to incumbent sequence, STOP. Otherwise go to step (4).

Example 4.5

Solve $F_3 || C_{max}$ problem for the data shown below using NEH algorithm.

	j_1	j_2	j_3	j_4
M_1	6	8	3	4
M_2	5	1	5	4
M_3	4	4	4	2

Solution:

For four jobs, the T_j values are shown in the Table 4.11

Table 4.11 Calculation for T_j values

	j_1	j_2	j_3	j_4
M1	6	8	3	4
M2	5	1	5	4
M3	4	4	4	2
T_j	**15**	**13**	**12**	**10**

The ordered list of jobs according to decreasing T_j values is; {j_1 , j_2 , j_3 , j_4}

Iteration 1

Since jobs j_1 and j_2 have highest values of T_j, select these two jobs to form partial schedule. The calculations of C_{max} value for partial schedule (j_1 , j_2 , * , *) are shown below in Table 4.12. Note C_{max} = 19 for the partial schedule (j_1 , j_2 , * , *).

Table 1 C_{max} calculations for partial schedule S_{12**}: (j_1, j_2, *, *)

	j_1	j_2	C_1	C_2	C_{max}
M_1	6	8	6	14	
M_2	5	1	11	15	
M_3	4	4	15	19	19

The calculations of C_{max} value for partial schedule (j_2 , j_1 ,* ,*) are shown below in Table 4.13. Note C_{max} = 23 for the partial schedule (j_2 , j_1 ,* ,*).

Table 4.13 C_{max} calculations for partial schedule S_{21**}: (j_2 , j_1 , * , *)

	j_2	j_1	C_1	C_2	C_{max}
M_1	8	6	8	14	
M_2	1	5	9	19	
M_3	4	4	13	23	23

The makespan (C_{max}) values for the partial schedules are;

Table 24.14 Comparison between the two partial sequences

Schedule	C_{max}
S_{21**}: (j_2 , j_1 , *, *)	23
S_{12**}: (j_1 , j_2 , * ,*)	19

Since value of C_{max} is smaller for partial sequence S_{12**}: (j_1 , j_2 , * , *), we further investigate this partial schedule. So partial sequence S_{21**}: (j_2 , j_1 , * , *) is fathomed and not investigated any more.

Iteration 2

Now job j_3 is next in ordered list after jobs j_1 and j_2 with a T_w value of 12. Job j_3 can be placed at three sequence positions in partial sequence (j_1, j_2, *, *).

a) Before job j_1 as follows: New Partial Sequence , S_{312*}: (j_3, j_1, j_2, *)

b) After job j_1 as follows: New Partial Sequence , S_{132*}: (j_1, j_3, j_2, *)

c) After job j_2 as follows: New Partial Sequence , S_{123*}: (j_1, j_2, j_3, *)

Calculations of C_{max} for sequence, S_{123*}:(j_1, j_2, j_3, *) are shown below in the following table 4.15.

Table 4.15 C_{max} calculations for partial sequence (j_1 , j_2 , j_3 , *)

	j_1	j_2	j_3	C_1	C_2	C_3	C_{max}
M_1	6	8	3	6	14	17	
M_2	5	1	5	11	15	22	
M_3	4	4	4	15	19	26	26

The Gantt chart of the partial schedule S_{123*}:(j_1, j_2, j_3,*) is shown in Figure

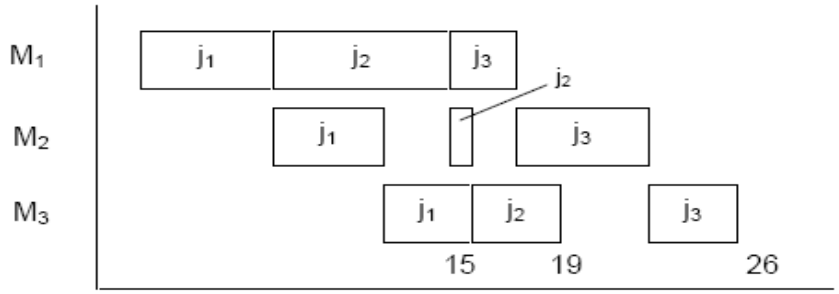

Figure 4.8 Gantt chart for the partial sequence S_{123*}: (j_1, j_2, j_3, *)

Note C_{max} = 26 for the partial schedule, S_{123*}:((j_1, j_2, j_3,*).

The calculations of C_{max} value for this schedule (j_3,j_1,j_2,*) are shown below in the following table 4.16.

Table 4.16 C_{max} calculations for partial schedule (j_3 , j_1 , j_2 , *)

	j_3	j_1	j_2	C_3	C_1	C_2	C_{max}
M_1	3	6	8	3	9	17	
M_2	5	5	1	8	14	18	
M_3	4	4	4	12	18	22	22

The calculations of C_{max} value for the schedule $S_{132*}$are shown in the following table 4.17.

Table 34.17 C_{max} calculations for partial schedule S_{132*}: (j_1, j_3, j_2, *)

	j_1	j_3	j_2	C_1	C_3	C_2	C_{max}
M_1	6	3	8	6	9	17	
M_2	5	5	1	11	16	18	
M_3	4	4	4	15	20	24	24

A comparison of the three schedules indicate that schedule S_{312*} : (j_3, j_1, j_2,*) results in minimum C_{max} value, as shown in below table 4.18:

Table 4.18 Comparison of the three partial sequences.

Partial Schedule	C_{max}
S_{123*} (j_1, j_2, j_3, *)	26
S_{312*} (j_3, j_1, j_2, *)	22
S_{132*} (j_1, j_3, j_2,*)	24

Iteration 3

Job j_4 is the last job in the ordered list. Using minimum C_{max} value partial schedule from iteration 2, generate four sequences by inserting job j_4 at four possible locations in partial sequence (j_3, j_1, j_2, *) as follows:

a) Before job j_3 as follows: New Sequence, S_{4312}: (j_4, j_3, j_1, j_2)
b) After job j_3 as follows: New Sequence, S_{3412}: (j_3, j_4, j_1, j_2)
c) After job j_1 as follows: New Sequence, S_{3142}: (j_3, j_1, j_4, j_2)
d) After job j_2 as follows: New Sequence, S_{3124}: (j_3, j_1, j_2, j_4)

The calculations of C_{max} value for this schedule (j_4, j_3, j_1, j_2) are shown below in the following table 4.19

Table 4.19 C_{max} calculations for schedule (j_4, j_3, j_1, j_2)

	j_4	j_3	j_1	j_2	C_4	C_3	C_1	C_2	C_{max}
M_1	4	3	6	8	4	7	13	21	
M_2	4	5	5	1	8	13	18	22	
M_3	2	4	4	4	10	17	22	26	26

The calculations of C_{max} value for this schedule (j_3, j_4, j_1, j_2) are shown below in the following table 4.20

Table 4.20 C_{max} calculations for schedule (j_3, j_4, j_1, j_2)

	j_3	j_4	j_1	j_2	C_3	C_4	C_1	C_2	C_{max}
M_1	3	4	6	8	3	7	13	21	
M_2	5	4	5	1	8	12	18	22	
M_3	4	2	4	4	12	14	22	26	26

The calculations of C_{max} value for this schedule (j_3, j_1, j_4, j_2) are shown below in the following table 4.21

Table 4.21 C_{max} calculations for schedule (j_3, j_1, j_4, j_2)

	j_3	j_1	j_4	j_2	C_3	C_1	C_4	C_2	C_{max}
M_1	3	6	4	8	3	9	13	21	
M_2	5	5	4	1	8	14	18	22	
M_3	4	4	2	4	12	18	20	26	26

The calculations of C_{max} value for this schedule (j_3, j_1, j_2, j_4) are shown below in the following table 4.22

Table 4.22 C_{max} calculations for schedule (j_3, j_1, j_2, j_4)

	j_3	j_1	j_2	j_4	C_3	C_1	C_2	C_4	C_{max}
M_1	3	6	8	4	3	9	17	21	
M_2	5	5	1	4	8	14	18	25	
M_3	4	4	4	2	12	18	19	27	27

The comparison of C_{max} values for the four schedules is presented below in table 4.23

Table 4.23 Comparison of the four partial sequences

Schedule	C_{max}
S_{3124} : (j_3, j_1, j_2, j_4)	27
S_{3124} : (j_4, j_3, j_1, j_2)	26
S_{3412} : (j_3, j_4, j_1, j_2)	26
S_{3142} : (j_3, j_1, j_4, j_2)	26

The NEH method yields three alternate schedules with a minimum makespan of 26. Clearly, NEH provides more elaborate results as compared to CDS or Slope heuristic.

The total enumeration tree for NEH method is shown in the figure 4.9 below.

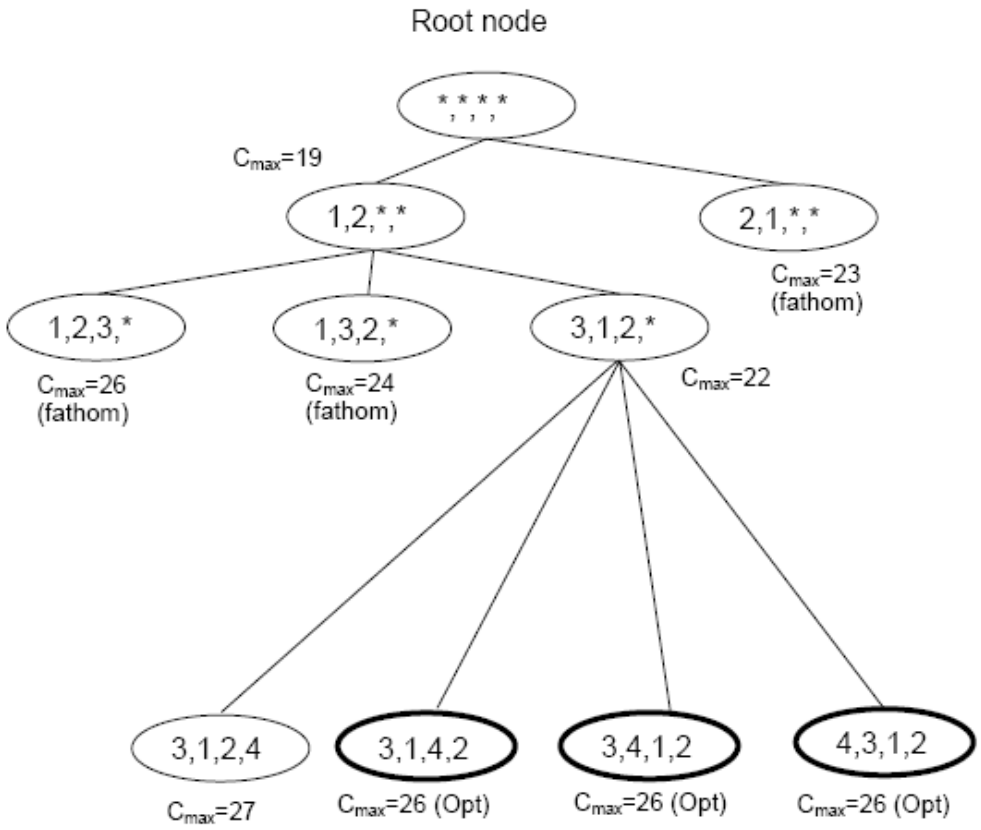

Figure 4.9 Enumeration tree for four-job problem using NEH algorithm**.**

4.7 BRANCH AND BOUND ALGORITHM

Branch and Bound algorithms have been proposed originally by Ignall and Schrage [1965] and Lomnicki [1965]. The application of the method to scheduling is based on the permutations schedules, which have a closed resemblance to a tree structure. The tree starts from an initial node and the initial or the first node corresponds to a situation where no jobs have been scheduled. This node has **n** branches as there are **n** possible jobs that can occupy first place in the sequence. From each of these **n** nodes, there are (**n**-1) branches corresponding to the (**n**-1) possible jobs that can be placed second in the sequence and so on. Since there are **n!** possible sequences, the tree has a total of 1 + n + n(n-1) +...+ n! nodes with each node representing a partial schedule.

As is obvious from above, the total number of nodes in the sequence is very large even for small number of jobs. Therefore, the Branch & Bound algorithm works on the principle of reducing the total number of nodes in search for an optimal solution. This is accomplished through:

- Presenting a branching rule to decide as on which node to branch from,
- Presenting an elimination rule which enables to discard a certain node and all nodes that emanate from it, from further consideration. Elimination of a certain node means that its partial sequence is dominated by some other partial sequence.

The Branch & Bound procedure starts from the initial node along the **n** nodes. Each time a new node is created, lower bound on makespan is calculated. Node corresponding to least lower bound is the one from where further branching is performed. Besides, dominance checks are made for discarding a node from further consideration. Many researchers have been working on developing sharper bounds and more powerful dominance conditions so as to shorten search for optimality. In this regard, Legeweg et al proposes a lower bound based on Johnson's two machines problem combined with an elimination criterion of Szwarc [1971] to solve quickly problems upto 50 jobs and 3 machines but the bounds become less reliable and solution times increase drastically as the number of machines exceed 3.

For any partial sequence S_k represented by a node on branch and bound tree, a lower bound (LB) for partial sequence S_k is calculated as follows:

$LB(k) = \max (A_1, A_2, A_3)$

Where,

$$A_1 = C_1(k) + \sum_{j \in U} p_{1j} + \min \sum_{j \in U} (p_{2j} + p_{3j})$$

$$A_2 = C_2(k) + \sum_{j \in U} p_{2j} + \min\{p_{3j}\}$$

$$A_3 = C_3(k) + \sum_{j \in U} p_{3j}$$

$C_1(k)$, $C_2(k)$ and $C_3(k)$ are completion times on machines M_1, M_2 and M_3 respectively for partial sequence S_k.

U = Set of unscheduled jobs; jobs not in the partial sequence S_k.

P_{1j}, P_{2j}, P_{3j} are process times of j^{th} job on machines M1, M2 and M3 respectively.

Example 4.6

Consider following data as an instance of F3 || C_{max} problem.

Jobs	M_1	M_2	M_3
1	4	6	2
2	3	7	1
3	6	2	5

Apply branch and bound method to find minimum C_{max} value.

Solution:

Starting from root node (*,*,*), create three nodes as follows:

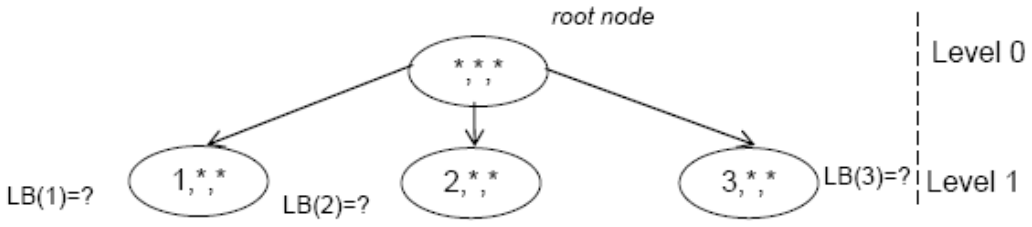

Figure 4.10 Branch and Bound Tree with Level 1 Nodes

Level 1 Computations

Calculate LB(1) for partial sequence (1,*,*) as follows:

First , find $C_1(k)$, $C_2(k)$ and $C_3(k)$ as shown below in figure 4.11

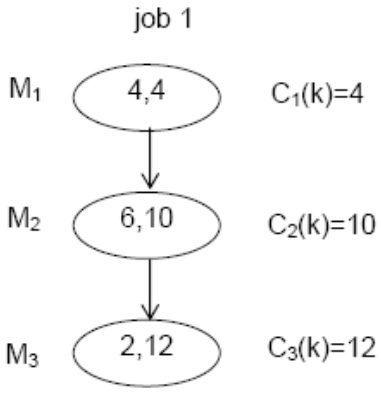

Figure 4.11 Directed graph for the partial sequence (1, *, *)

Note, set U = {j2, j3}. The calculations are shown below.

Table 4.23 Calculation of lower bound for partial Sequence (1,* ,*)

Jobs	**M_1**	**M_2**	**M_3**	**$p_{2j}+p_{3j}$**	**p_{3j}**
1	4	6	2		
2	3	7	1	8	1
3	6	2	5	7	5
	Σp_{1j}=	Σp_{2j}=	Σp_{3j}=	min	min
	9	9	6	7	1

$$A_1 = C_1(k) + \sum_{j\in U} p_{1j} + \min \sum_{j\in U} (p_{2j} + p_{3j}) = 4 + 9 + 7 = 20$$

$$A_2 = C_2(k) + \sum_{j\in U} p_{2j} + \min\{p_{3j}\} = 10 + 9 + 1 = 20$$

$$A_3 = C_3(k) + \sum_{j\in U} p_{3j} = 12 + 6 = 18$$

$$LB(1) = \max(A_1, A_2, A_3) = \max(20,20,18) = 20$$

Similarly, calculate LB(2) for partial sequence (2,*,*) as follows:
First, find $C_1(k)$, $C_2(k)$ and $C_3(k)$ as shown below in figure 4.12

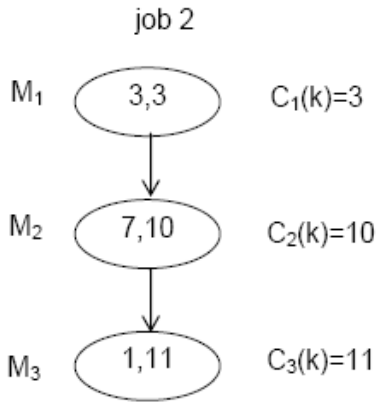

Figure 4.12 Directed graph for the partial sequence (2, *, *).

Note, set U = { j1 , j3}. The calculations are shown below.

Table 4.24 Calculation of lower bound for partial Sequence (2, *, *)

Jobs	**M_1**	**M_2**	**M_3**	**$p_{2j}+p_{3j}$**	**p_{3j}**
1	4	6	2	8	2
2	3	7	1		
3	6	2	5	7	5
	$\Sigma p_{1j}=$	$\Sigma p_{2j}=$	$\Sigma p_{3j}=$	min	min
	10	8	7	7	2

$$A_1 = C_1(k) + \sum_{j\in U} p_{1j} + \min \sum_{j\in U} (p_{2j} + p_{3j}) = 3 + 10 + 7 = 20$$

$$A_2 = C_2(k) + \sum_{j\in U} p_{2j} + \min\{p_{3j}\} = 10 + 8 + 2 = 20$$

$$A_3 = C_3(k) + \sum_{j\in U} p_{3j} = 11 + 7 = 18$$

$$LB(2) = \max(A_1, A_2, A_3) = \max(20,20,18) = 20$$

Similarly, calculate LB(3) for partial sequence (3,*,*) as follows:
First, find $C_1(k)$, $C_2(k)$ and $C_3(k)$ as shown in figure below.

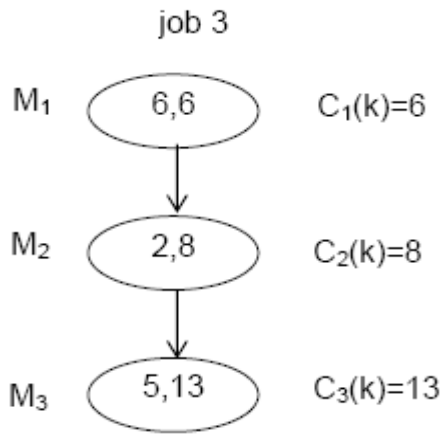

Figure 4.13 Directed graph for the partial sequence (3, *, *).

Note, set U = { j1 , j2}. The calculations are shown below.

Table 4.25 Calculation of lower bound for partial Sequence (3, *, *)

Jobs	M_1	M_2	M_3	$p_{2j}+p_{3j}$	p_{3j}
1	4	6	2	8	2
2	3	7	1	8	1
3	6	2	5		
	Σp_{1j}=	Σp2j=	Σp3j=	min	min
	7	13	3	8	1

$$A_1 = C_1(k) + \sum_{j\in U} p_{1j} + \min \sum_{j\in U} (p_{2j} + p_{3j}) = 6 + 7 + 8 = 21$$

$$A_2 = C_2(k) + \sum_{j\in U} p_{2j} + \min\{p_{3j}\} = 8 + 13 + 1 = 22$$

$$A_3 = C_3(k) + \sum_{j\in U} p_{3j} = 13 + 3 = 16$$

$$LB(3) = \max(A_1, A_2, A_3) = \max(21,22,16) = 22$$

The values of lower bound for first level nodes are entered for respective sequence. Since nodes 1 and 2 have equal values of lower bound, branch to lower level nodes from these nodes as shown in Figure 4.14. Thus, fathom node 3 for further branching.

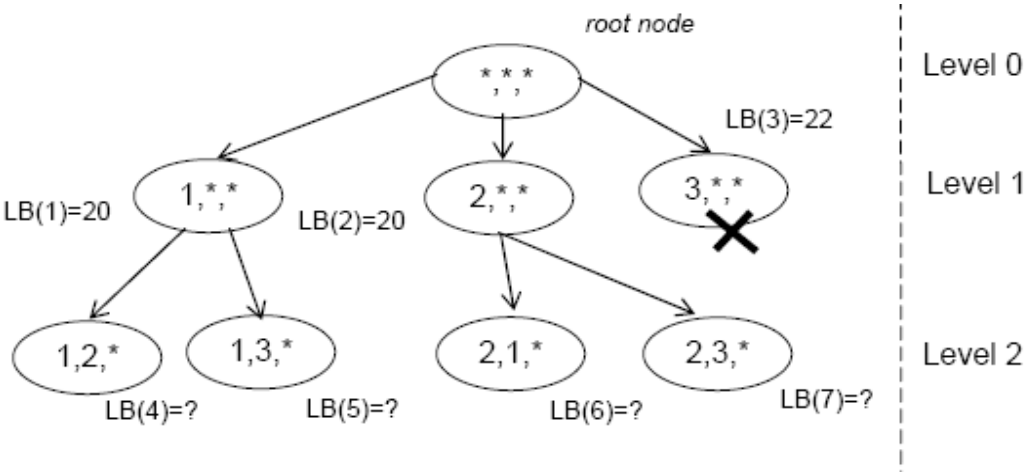

Figure 4.14 Branch and bound tree with level two nodes.

Level 2 Computations

Calculate LB for partial sequence (1,2,*) as follows:

First, find $C_1(k)$, $C_2(k)$ and $C_3(k)$ as shown in Figure 4.15 below.

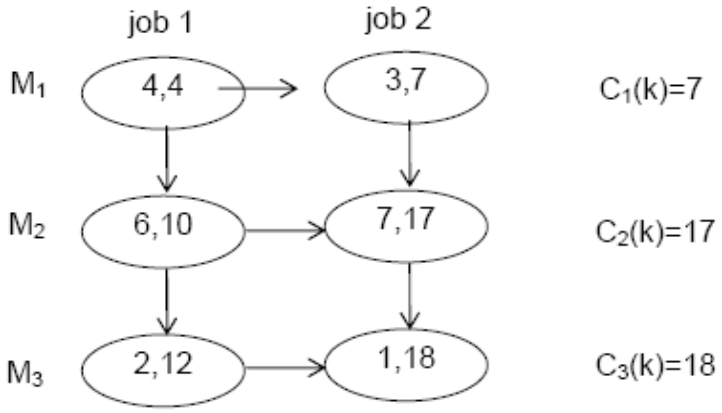

Figure 4.15 Directed graph for partial sequence.

Note, set U = {j_3}. The calculations are shown below.

Table 4.26 Calculation of lower bound for partial Sequence (1, 2, *)

Jobs	**M_1**	**M_2**	**M_3**	**$p_{2j}+p_{3j}$**	**p_{3j}**
1	4	6	2		
2	3	7	1		
3	6	2	5	7	5
	Σp_{1j}=	Σp_{2j}=	Σp_{3j}=	min	min
	6	2	5	7	5

$$A_1 = C_1(k) + \sum_{j \in U} p_{1j} + \min \sum_{j \in U} (p_{2j} + p_{3j}) = 7 + 6 + 7 = 20$$

$$A_2 = C_2(k) + \sum_{j \in U} p_{2j} + \min\{p_{3j}\} = 17 + 2 + 5 = 24$$

$$A_3 = C_3(k) + \sum_{j \in U} p_{3j} = 18 + 5 = 23$$

$$LB(4) = \max(A_1, A_2, A_3) = \max(20,24,23) = 24$$

Similarly, calculate LB(5) for partial sequence (1,3,*) as follows:
First, find $C_1(k)$, $C_2(k)$ and $C_3(k)$ as shown in Figure 4.16 below.

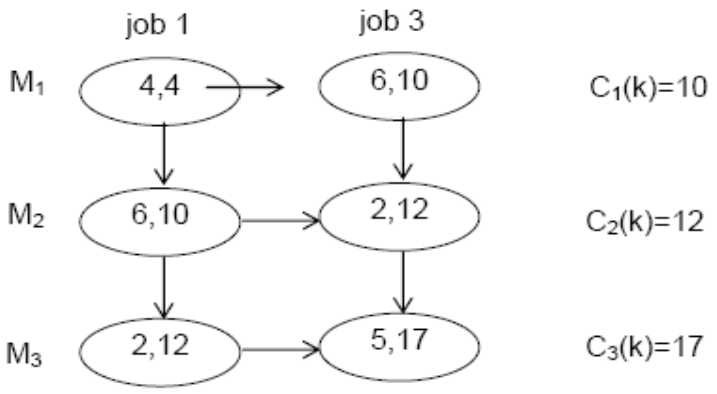

Figure 4.16 Directed graph for partial sequence (1, 3, *).

Note, set U = { j2}. The calculations are shown below.

Table 4.27 Calculation of lower bound for partial Sequence (1, 3, *)

Jobs	**M_1**	**M_2**	**M_3**	**$p_{2j}+p_{3j}$**	**p_{3j}**
1	4	6	2		
2	3	7	1	8	1
3	6	2	5		
	Σp_{1j}=	Σp_{2j}=	Σp_{3j}=	min	min
	3	7	1	8	1

$$A_1 = C_1(k) + \sum_{j \in U} p_{1j} + \min \sum_{j \in U} (p_{2j} + p_{3j}) = 10 + 3 + 8 = 21$$

$$A_2 = C_2(k) + \sum_{j \in U} p_{2j} + \min\{p_{3j}\} = 12 + 7 + 1 = 20$$

$$A_3 = C_3(k) + \sum_{j \in U} p_{3j} = 17 + 1 = 18$$

$$LB(4) = \max(A_1, A_2, A_3) = \max(21,20,18) = 21$$

Similarly, calculate LB(6) for partial sequence (2,1,*) as follows:
First, find $C_1(k)$, $C_2(k)$ and $C_3(k)$ as shown in Figure 4.17 below.

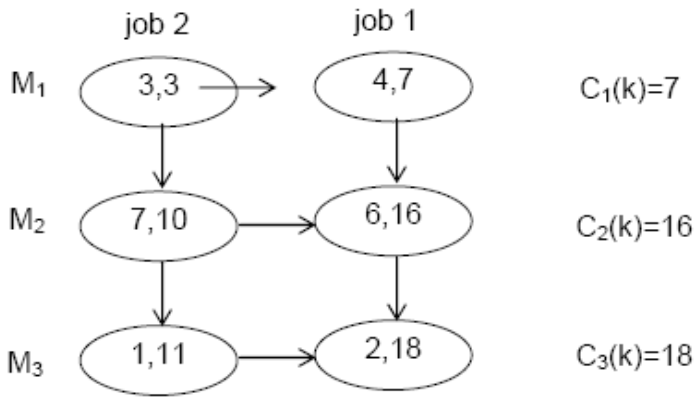

Figure 4.17 Directed graph for the partial sequence (2, 1, *).

Note, set U = { j3}. The calculations are shown below.

Table 4.28 Calculation of lower bound for partial Sequence (2, 1, *)

Jobs	**M_1**	**M_2**	**M_3**	**$p_{2j}+p_{3j}$**	**p_{3j}**
1	4	6	2		
2	3	7	1		
3	6	2	5	7	5
	$\Sigma p_{1j}=$	$\Sigma p_{2j}=$	$\Sigma p_{3j}=$	min	min
	6	2	5	7	5

$$A_1 = C_1(k) + \sum_{j\in U} p_{1j} + \min \sum_{j\in U} (p_{2j} + p_{3j}) = 7 + 6 + 7 = 20$$

$$A_2 = C_2(k) + \sum_{j\in U} p_{2j} + \min\{p_{3j}\} = 16 + 2 + 5 = 23$$

$$A_3 = C_3(k) + \sum_{j\in U} p_{3j} = 18 + 5 = 23$$

$$LB(6) = \max(A_1, A_2, A_3) = \max(20,23,23) = 23$$

Finally, calculate LB(7) for partial sequence (2,3,*) as follows:
First, find $C_1(k)$, $C_2(k)$ and $C_3(k)$ as shown in Figure 4.18 below.

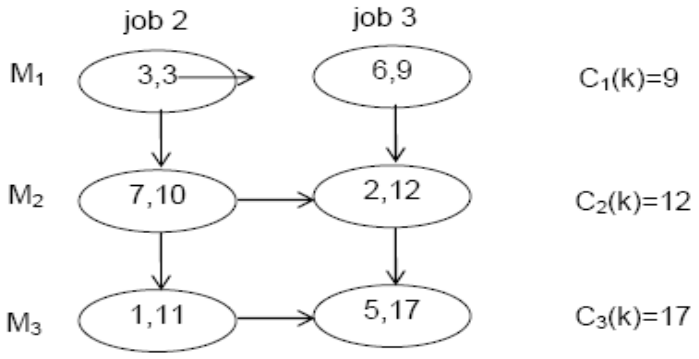

Figure 4.18 Directed graph for partial sequence (2, 3, *)

Note, set U = { j1}. The calculations are shown below.

Table 4.29 Calculation of lower bound for partial Sequence (2, 3, *)

Jobs	**M_1**	**M_2**	**M_3**	**$p_{2j}+p_{3j}$**	**p_{3j}**
1	4	6	2	8	2
2	3	7	1		
3	6	2	5		
	Σp_{1j}=	Σp_{2j}=	Σp_{3j}=	min	min
	4	6	2	8	2

$$A_1 = C_1(k) + \sum_{j \in U} p_{1j} + \min \sum_{j \in U} (p_{2j} + p_{3j}) = 9 + 4 + 8 = 21$$

$$A_2 = C_2(k) + \sum_{j \in U} p_{2j} + \min\{p_{3j}\} = 12 + 6 + 2 = 20$$

$$A_3 = C_3(k) + \sum_{j \in U} p_{3j} = 17 + 2 = 19$$

$$LB(7) = \max(A_1, A_2, A_3) = \max(21,20,19) = 21$$

The values of lower bounds for nodes 4, 5, 6 and 7 are shown in Figure 4.19 below.

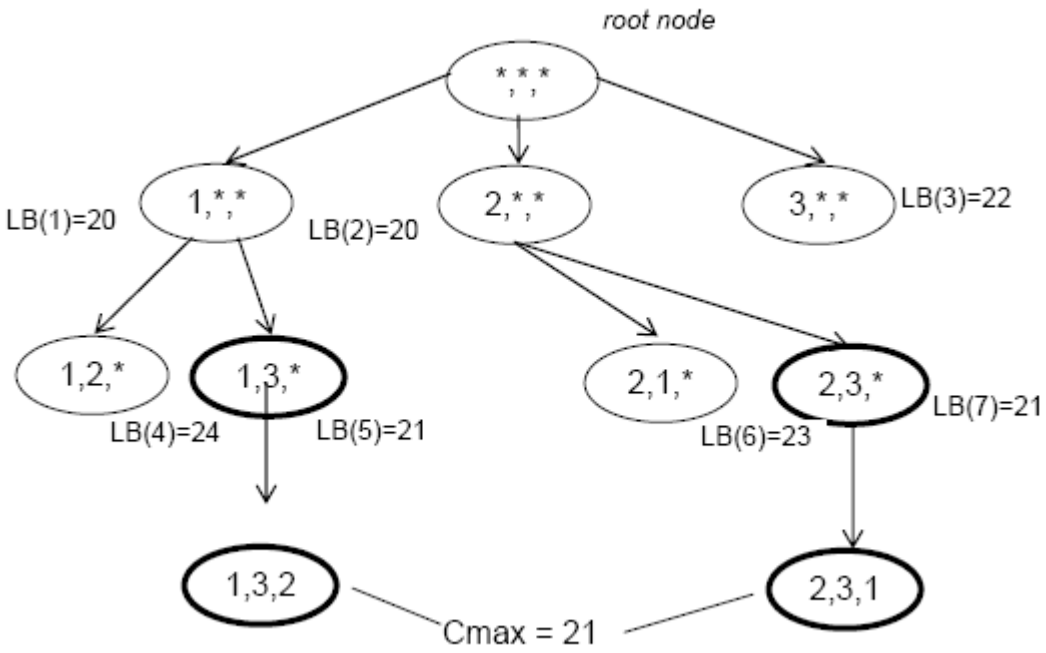

Figure 1 The complete branch and bound tree with an optimal solution.

The sequences for nodes 5 and 7 provide optimal solution for the problem under consideration.

EXERCISES

4.1 Consider the schedule 3-2-1-4 with the make span 24 units of time for the following $F_3 \mid\mid C_{max}$ problem given the following data:

Job	1	2	3	4
M_1	3	2	1	8
M_2	5	1	8	7
M_3	7	4	2	2

Construct the Gantt chart for the problem. Show mathematically that the schedule is optimal.

4.2 Consider a 3-machine 4-job problem as below.

Job	1	2	3	4
M_1	4	7	3	1
M_2	9	6	3	8
M_3	7	4	8	5

Solve problem using;

(i) CDS Algorithm, (ii) NEH Algorithm

4.3 Consider $F_3 \mid\mid C_{max}$ Problem given the following data:

Job	1	2	3	4	5	6
M_1	5	6	30	2	3	4
M_2	8	30	4	5	10	1
M_3	20	6	5	3	4	4

Use the branch and bound method to find the optimal makespan for this problem. (Hint: LB = max (A_1, A_2, A_3)).

4.4 The data pertaining to $F_3 \mid\mid C_{max}$ problem is shown in Table below.

Job	1	2	3	4
M_1	6	8	3	4
M_2	4	1	2	3
M_3	5	6	4	7

a) Apply Johnson' Rule and find optimal solution.

b) Apply CDS heuristic and find C_{max}

c) If due dates of the jobs are as follows:

Job	1	2	3	4
Due Date	17	13	11	18

Find tardiness of the jobs using Johnson's Rule and CDS heuristics. Which of the two methods provides minimum value of L_{max}?

4.5 Consider $F_3 \mid\mid C_{max}$ with the following data:

	Job					
Machines	1	2	3	4	5	6
M_1	2	23	25	5	15	10
M_2	29	3	20	7	11	2
M_3	19	8	11	14	7	4

Use the following methods to find the best sequence:

i. Compbell, Dudek, and Smith (CDS) approach,

ii. Nawaz, Enscore, and Ham (NEH) approach.

Show all of your work. Then, compare the results obtained by the two approaches.

4.6 Consider $F_4 \mid\mid C_{max}$ with the following data:

Job	1	2	3	4	5
1	1	10	17	12	11
2	13	12	9	17	3
3	6	18	13	2	5
4	2	18	4	6	16

Use the Compbell, Dudek, and Smith (CDS) approach to find the best sequence. Also, draw all Gantt Charts for the CDS schedule.

4.7 Consider $F_3 \mid\mid C_{max}$ with the following data:

Job	1	2	3	4	5
M_1	1	10	17	12	11
M_2	13	12	9	17	3
M_3	6	18	13	2	5

Find the optimal makespan using the branch and bound. Also, draw the final Gantt Charts for the Branch and bound schedule and draw the tree for the Branch and bound.

4.8 Consider $F_3 \mid\mid C_{max}$ with the following data:

Job	1	2	3	4
M1	6	8	3	4
M2	3	1	3	3
M3	4	4	4	2

Find the best solution for the above problem

4.9 Consider $F_3 \mid\mid C_{max}$ problem with the following data:

Job	1	2	3	4
M/c 1	5	6	30	2
M/c 2	8	30	4	5
M/c 3	20	6	5	3

a) Use the branch and bound method to find the optimal sequence that minimizes the makespan.
b) Draw the Gantt Charts for the optimal sequence.

c) Assume a common due date for all jobs which is the average total work content among jobs, then, compute the makespan, total tardiness, and maximum lateness.