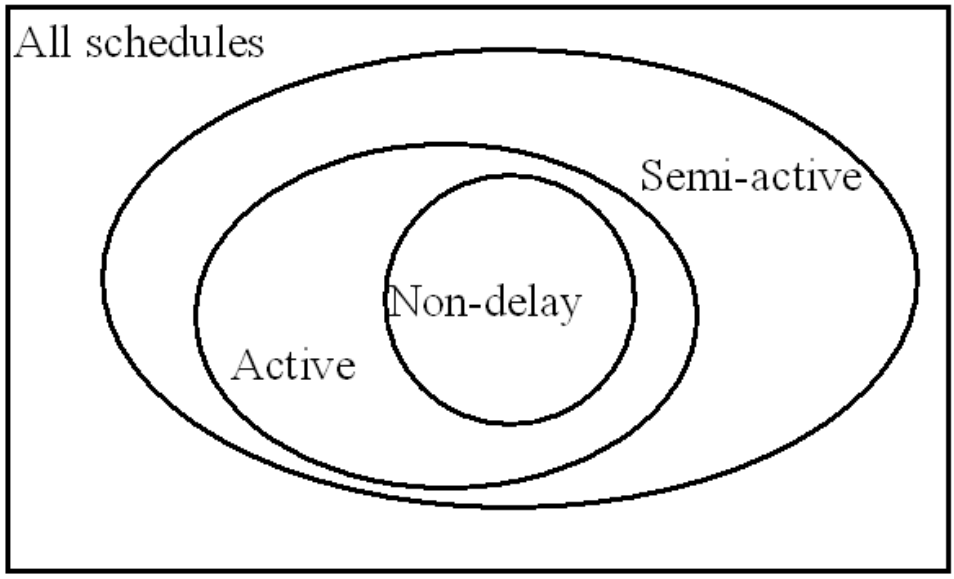
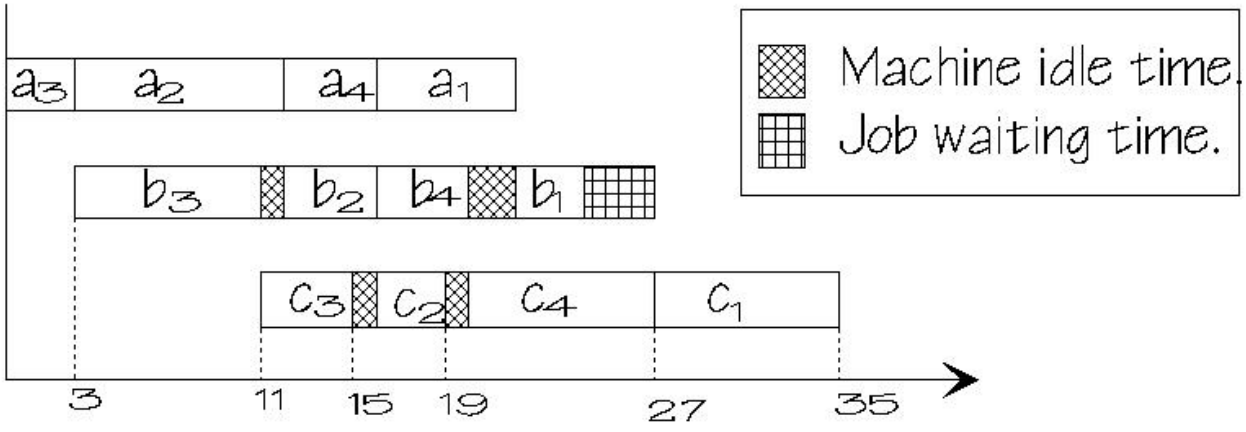
ALGORITHMS FOR SEQUENCING AND SCHEDULING





Algorithms for Sequencing and Scheduling

Ibrahim M. Alharkan

Industrial Engineering Department
College of Engineering
King Saud University
Riyadh, Saudi Arabia

Contents

Preface

This book deals with the field of sequencing and scheduling algorithms. Sequencing and scheduling is a form of decision-making that plays a crucial role in manufacturing and service industries. Scheduling is one of the most mathematically involved and developed fields in Industrial Engineering and Operations Research. In fact, scheduling began to be taken seriously in manufacturing at the beginning of 20^{th} century with the work of Henry Gantt and other pioneers. However, the first scheduling algorithms were formulated in the mid of 20^{th} century. Since then there has been a growing interest in scheduling. A rich body of knowledge composed of scientific papers and books has been built by scheduling academicians and practitioners. From this rich literature, I selected the classical scheduling algorithms that are taught in most industrial operations scheduling courses. The book is not intended to cover most of the algorithms available in the scheduling literature since a complete treatment of this scientifically rich field is impossible in given time and effort constraints. Compared to existing textbooks, this book provides a more comprehensive and expanded coverage of these algorithms. The book can be recommended as a primary or supplementary text for undergraduate level courses in Industrial operations scheduling.

I intended this book to be used as a textbook in upper-division and graduate-level Scheduling courses. I was also aware that this book has been an important resource for scheduling professionals. To balance these two purposes, I have emphasized the empirical research basis of operations scheduling, I have stressed basic concepts and the operation scheduling considerations involved in the topics covered, and I have supplied references for those who wish to delve into particular area. I tried to maintain a scholarly approach to the field. Unfortunately, there are times when my presentation may be little technical especially when I am presenting information that would be more appropriate for the practicing Scheduling specialist than for student. But I hope the book will be one student want to keep as a valuable reference.

Despite the huge number of books available on the theory and algorithms for Sequencing and scheduling problems. This book is the result of the development of courses in scheduling theory and applications at King Saud University. The book serves

the teaching process plus puts together all theories and algorithm in one reference. The book deals primarily with machine scheduling and project resources allocation models. The book can be viewed as consisting of four major parts. The first part, Chapter 1, covers basics like an introduction to and classification of scheduling problems, methods of optimization that are relevant for the solution procedures. The second part, Chapters 2 through 5, covers classical scheduling algorithms for solving single machine problems, parallel machine problems, and shop scheduling problems. The third part, Chapters 7 and 6, covers the project scheduling problems for allocating resources. The fourth and final part, Chapters 8 and 9, is devoted to Met-Heuristics which have attacked scheduling problems such as Simulated Annealing, Tabu search, and Genetic Algorithms. Hopefully there will be in a couple of years a second edition in which the applications part will be expanded, showing a stronger connection with the more theoretical parts of the text. Hopefully, the book will contribute to the effort of enhancing the understanding of the mathematically complicated sequencing and scheduling Algorithms.

Ibrahim Alharkan

Acknowledgments

I would like to express my deep appreciation for the Research Center at the College of Engineering, King Saud University for sponsoring the project of writing this book under the Research Grant number 429/22.

I would like also to extend my appreciation to my colleagues for proof reading the book and for giving me valuable suggestions and critiques. I particularly thank Prof. Badiru, and Eng. Rafi Qureshi for their continuous support and encouragement to finish this book.

Special acknowledgment and thanks go to Eng. Yusuf Usmani for his continual technical assistance in writing the book.

I thank my undergraduate and graduate students at King Saud University who read some chapters, solved some problems, and discovered some typos in earlier drafts of the book.

The final acknowledgment must go to my parents, my wife, and my kids for their understanding and patience while I was occupied with the writing of this book. In addition, I am very appreciative of their continual encouragement and continuous motivation while writing this book. I dedicate the book to them.

1

Scheduling Theory

CHAPTER CONTENTS

The scheduling process most often arises in a situation where resource availabilities are essentially fixed by the long term commitments of a prior planning decision. With this background in mind, we can describe the steps by which scheduling decisions are reached as the systems approach. In this regard, the four primary stages include: formulation, analysis, synthesis and evaluation. In the first stage, basically, a problem is identified and the criteria that should guide decision making are determined. Analysis is the detailed process of examining the elements of a problem and their inter-relationships. This stage is aimed at identifying the decision variables and also at specifying the relationships among them and the constraints they must obey. Synthesis is the process of building alternative solutions to the problem. Its role is to characterize the feasible options that are available. Finally, evaluation is the process of comparing these feasible alternatives and selecting a desirable course of action.

1.1 INTRODUCTION

Scheduling is the task of determining when each operation is to start and finish. Since each operation is in possible competition with other operations for scarce resources of time and capacity, the job of scheduling is neither simple nor easy.

Scheduling is the allocation of resources over time to perform a collection of tasks. This rather general definition of the term does convey two different meanings. First scheduling is a decision making function. Second scheduling is a body of theory as it is a collection of principles, models, techniques and logical conclusions that provide insight into the scheduling function.

It may be worth mentioning here to distinguish between terms "Scheduling" and "Sequencing" which as a matter of fact, are both associated with the job shop process. Scheduling is defined as assigning each operation of each job a start time and a completion time on a time scale of machine within the precedence relations. However, under certainty, a schedule can not be determined because the arrival time of the job, operation processing time, as well as other attributes are not fully known. On the other hand sequencing means that for each machine in the shop, one has to establish the order in which the jobs waiting in the queue in front of that particular machine have to be processed.

The problem that motivated this study is as follows: suppose there are a number of jobs to be performed. Each job consists of a given sequence of operations which needs to be performed using a number of machines. All operations for each job must be performed in the order given by the sequence. Each operation demands the use of a particular machine for a given time. Each machine can process only one operation at a time. Therefore, given a cost function by which each sequence can be evaluated, the order of operations on each machine that minimizes the cost function needs to be found.

Sequencing and scheduling problems occur in different industries and circumstances. The following are some examples of different situations which need sequencing or scheduling:

i. parts waiting for processing in a manufacturing plant;

ii. aircraft waiting for landing clearance at an airport;

iii. computer programs running at a computing center;

iv. class scheduling in a school,

v. patients waiting in a Doctor's office;

vi. ships to be anchored in a harbor, and

vii. Thursday afternoon chores at home.

1.2 DEFINITIONS

Production sequencing and scheduling is one of the most important activities in production planning and control. Morton and Pentico discussed how important the sequencing and scheduling role is, stating that "it pervades all economic activity" (Morton and Pentico 1993, 5). Pinedo further discussed the importance of the sequencing and scheduling problem:

> *...Sequencing and scheduling are forms of decision-making which play a crucial role in manufacturing as well as in service industries. In the current competitive environment, effective sequencing and scheduling has become a necessity for survival in the marketplace. Companies have to meet shipping dates committed to the customers, as failure to do so may result in a significant loss of good will. They also have to schedule activities in such a way as to use the resources available in an efficient manner. (Pinedo 1995, xiii)*

The definition of sequencing among researchers is common. Sequencing is defined as the order in which the jobs (tasks) are processed through the machines (resources). Scheduling was defined by Baker as follows:

> *...Scheduling is the allocation of resources over time to perform a collection of tasks.... Scheduling is a decision-making function: it is the process of determining a schedule.... Scheduling is a body of theory: it is a collection of principles, models, techniques, and logical conclusions that provide insight into the scheduling function. (Baker 1974, 2)*

Also, Morton and Pentico defined scheduling as follows:

> *...Scheduling is the process of organizing, choosing, and timing resource usage to carry out all the activities necessary to produce*

> *the desired outputs at the desired times, while satisfying a large number of time and relationship constraints among the activities and the resources. (Morton and Pentico 1993, 5)*

Therefore, from the above two definitions, scheduling can be defined as a decision-making process that is concerned with the allocation of limited machines (resources) over time to perform a collection of jobs (tasks) in which one or several objectives have to be optimized.

The general definition of the sequencing problem can be stated as follows:

> There are m machines $\{M_1, M_2, ..., M_m\}$ available and n jobs $\{J_1, J_2,..., J_n\}$ to be processed. A subset of these machines is required to complete the processing of each job. The flow pattern (process plan) for some or all jobs may or may not be fixed. Each job should be processed through the machines in a particular order that satisfies the job's technological constraints. The processing of job i on machine j is called an operation denoted by O_{ij}. Associated with each operation is a processing time denoted by P_{ij}, and a setup time denoted by S_{ij}. Also, associated with each job is a weight, w_i, a ready (release or arrival) time, r_i, and a due date, d_i. Finally, each job has an allowance time to be in the shop, a_i.

Thus, the general problem is to generate a sequence that satisfies the following conditions:

i. all jobs are processed;
ii. all technological constraints are met for all jobs (feasibility condition), and
iii. all criteria that were selected are optimized.

1.3 LEVELS OF THE SEQUENCING AND SCHEDULING PROBLEM

Sequencing and scheduling are involved in planning and controlling the decision-making process of manufacturing and service industries in several stages. According to several researchers (Baker 1974; Browne, Harhen, and Shivnan 1988; Muchnik 1992; and Morton and Pentico 1993), sequencing and scheduling exist at several levels of the decision-making process. These levels are as follows:

1) **Long-term planning** which has a horizon of 2 to 5 years. Some examples are: plant layout, plant design, and plant expansion.

2) **Middle-term planning** such as production smoothing and logistics which can be done in a period of 1 to 2 years.

3) **Short-term planning** which is done every 3 or 6 months. Examples include: requirements plan, shop bidding, and due date setting.

4) **Predictive scheduling** which is performed in a range of 2 to 6 weeks. Job shop routing, assembly line balancing, and process batch sizing qualify as predictive.

5) **Reactive scheduling or control** which is performed every day or every three days. A few examples are: hot jobs, down machines, and late material.

1.4 SCHEDULING ENVIRONMENTS

According to Conway, Maxwell, and Miller (1967), sequencing and scheduling environments are classified according to four types of information: the jobs and operations to be processed; the number and types of machines that comprise the shop; the disciplines that restrict the manner in which assignment can be made, and the criteria by which a schedule will be evaluated. The sequencing and scheduling environments are as follows:

1) **Single machine shop**: one machine and n jobs to be processed.

2) **Flow shop**: there are m machines in series and jobs can be processed in one of the following ways:
   a) Permutational: jobs are processed by a series of m machines in exactly the same order, or
   b) Non-permutational: jobs are processed by a series of m machines not in the same order.

3) **Job shop**: each job has its flow pattern and a subset of these jobs can visit each machine twice or more often. Multiple entries and exits.

4) **Assembly job shop**: a job shop with jobs that have at least two component items and at least one assembly operation.

5) **Hybrid job shop**: the precedence ordering of the operations of some jobs is the same.

6) **Hybrid assembly job shop**: combines the features of both the assembly and hybrid job shop.

7) **Open shop**: there are m machines and there is no restriction in the routing of each job through the machines. In other words, there is no specified flow pattern for any job.

8) **Closed shop**: it is a job shop; however, all production orders are generated as a result of inventory replenishment decisions. In other words, the production is not affected by the customer order.

1.5 ASSUMPTIONS

A variety of assumptions is made in sequencing and scheduling problems The nature of these assumptions depends on the sequencing environment. The following list contains typical assumptions generally applied to scheduling problem with variations depending on the situation.

1) the set of the jobs and the set of the machines are known and fixed;

2) all jobs and all machines are available at the same time and are independent;

3) all jobs and machines remain available during an unlimited period;

4) the processing time for each job on all machines is fixed, has a known probability distribution function, and sequence independent;

5) setup times are included in processing times;

6) a basic batch size is fixed for all jobs;

7) all jobs and all machines are equally weighted;

8) no preemption is allowed;

9) a definite due date is assigned to each job;

10) each job is processed by all the machines assigned to it;

11) each machine processes all the jobs assigned to it,

12) the process plan for each job is known and fixed.

1.6 CATEGORIES OF SCHEDULING PROBLEMS

When none, one, or more of the assumptions used is/are relaxed, then the sequencing and scheduling problem is categorized into one of the following:

a) **Deterministic** sequencing and scheduling problems: when all elements of the problem, such as the state of the arrival of the jobs to the shop, due-dates of jobs, ordering, processing times and availability of machines, do not include stochastic factors and are determined in advance.

b) **Static** sequencing and scheduling problems: the same as deterministic problems except that the nature of the job arrival is different. The set of jobs over time does not change, and it is available beforehand.

c) **Dynamic** sequencing and scheduling problems: the set of jobs changes over time and jobs arrive at different times.

d) **Stochastic** sequencing and scheduling problems: at least one of the problem elements includes a stochastic factor.

1.7 SCHEDULING FRAME WORK AND NOTATIONS

A scheduling problem is described by the following notational form:

$$\alpha \mid \beta \mid \gamma$$

α : notation describes the machine/scheduling environment.

β : notation is used to explain the processing characteristics and constraints.

γ : notation contains information on the objective function to be attained.

The following tables provide more information on the three notations.

Table 1.1 Notations for Common Machine Environment (α)

Environment Name	Symbol / Notation	Description
Single Machine	1	One machine
Identical Machines in Parallel	P_m	P : Parallel machines "m" : number of machines
Parallel machines with different speeds	Q_m	Q : Parallel machines with different speeds "m" : number of machines
Flow Shop	F_m	F : Flow shop "m" : number of machines
Job Shop	J_m	J : Job shop "m" : number of machines
Open Shop	O_m	O: Open shop "m" : number of machines

Table 1.2 Notations for Common Processing Characteristics / Constraints (β)

Term	Notation	Description
Release Date	r_j	This term if present indicates the dynamic shop; a job cannot start its processing on a machine prior to its r_j value.
Preemptions	*Prmp*	A job may interrupted during its processing due to arrival of a high priority job
Precedence constraints	*Prec*	When one job depends on the completion of another job, this implies precedence constraint
Breakdowns	*Brkdwn*	This implies that machines are not continuously available for processing
Recirculation	*Recrc*	When a job visits a machine more than once
Permutation	*Prmu*	The processing order of all jobs on one machine is maintained throughout the shop

Table 1.3 Notations for Common scheduling objective functions (γ).

Makespan	C_{max}
Maximum Lateness	L_{max}
Total weighted completion time	$\Sigma\omega_j C_j$
Total weighted Tardiness	$\Sigma\omega_j T_j$
Total completion times	ΣC_j

1.8 DECISION-MAKING GOALS

According to Baker (1974), there are three common types of decision-making goals in sequencing and scheduling problems: efficient utilization of resources; rapid response to demands, and close conformance to prescribed deadlines.

- Efficient utilization of resources. Schedule activities so as to maintain high utilization of labor, equipment and space.
- Rapid response to demands. Scheduling should allow jobs to be processed at rapid rate resulting in low levels of work-in-process inventory.
- Close conformance to meet deadlines. Scheduling should ensure that due dates are met with every time through shorter lead times.

The three common goals can be achieved by associating the criteria mentioned above with each of the three goals as follows:

a) Efficient utilization of machines (resources):

Minimize C_{max} or $\bar{I}$, or maximize $\bar{N}_p$ or $\bar{U}$.

b) Rapid response to demands:

Minimize $\sum_{i=1}^{n} C_i$; $\sum_{i=1}^{n} F_i$; $\sum_{i=1}^{n} L_i$; $\sum_{i=1}^{n}\sum_{j=1}^{m} W_{ij}$; $\bar{C}$; $\bar{F}$; $\bar{L}$, $\bar{N}_w$, or $\bar{W}$.

c) Close conformance to prescribed deadlines:

Minimize L_{max}; T_{max}; NT; $\sum_{i=1}^{n} T_i$; $\bar{T}$, or $\sum_{i=1}^{n} w_i T_i$.

1.9 CRITERIA OF THE SEQUENCING AND SCHEDULING

According to Rinnooy Kan (1976) and French (1982), the criteria for sequencing and scheduling problems are classified according to three measures: completion times; due dates, and inventory and machine utilization. With each of the three measures, the following criteria can be associated, as shown in Tables 1, 2 and 3.

In the sequencing and scheduling literature, there are other criteria such as a combination of two or more of the above mentioned criteria. Also, there are other criteria in the sequencing and scheduling literature that were not mentioned above. For additional criteria, the reader can refer to Conway, Maxwell, and Miller (1967); Baker (1974); Rinnooy (1976); Bellman, Esogbue, and Nabeshima (1982); French (1982); Morton and Pentico (1993); and Pinedo (1995).

1.9.1 Criteria based on completion times

1)	Completion time of job i	C_i
2)	The total completion time	$\sum_{i=1}^{n} C_i$
3)	The total weighted completion time	$\sum_{i=1}^{n} w_i C_i$
4)	The total weighted waiting time	$\sum_{i=1}^{n} w_i \sum_{j=1}^{m} W_{ij}$
5)	Flow time of job i	$F_i = C_i - r_i$
6)	Maximum completion time (the schedule time. or makespan)	$C_{max} = \max_{1,\ldots,n} \{C_i\}$.
7)	The total flow time	$\sum_{i=1}^{n} F_i$
8)	The total weighted flow time	$\sum_{i=1}^{n} w_i F_i$
9)	Average flow time	$\overline{F}$
10)	Maximum flow time	F_{max}
11)	Waiting time of job I	$Wi = F_i - \sum_{j=1}^{m} P_{ij}$.

12)	The total waiting time	$\sum_{i=1}^{n} \sum_{j=1}^{m} W_{ij}$
13)	Average completion time	$\overline{C}$
14)	Average waiting time	$\overline{W}$

1.9.2 Criteria based on inventory and machine utilization

1)	Average number of jobs waiting for machines	$\overline{N}_w$
2)	Average number of unfinished jobs	$\overline{N}_u$
3)	Average number of jobs completed	$\overline{N}_c$
4)	Average number of jobs actually being processed	$\overline{N}_p$
5)	Average number of machines idle	$\overline{I}$
6)	Maximum machine idle time	I_{max}
7)	Average utilization	$\overline{U} = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} / m.C_{max}$

1.9.3 Criteria based on due-dates.

1)	Lateness of job i	$L_i = C_i - d_i$
2)	The total lateness	$\sum_{i=1}^{n} L_i$
3)	The total weighted lateness	$\sum_{i=1}^{n} w_i L_i$
4)	Average lateness	$\overline{L}$

5)	Maximum lateness	$L_{max} = \max_{1,\ldots,n} \{Li\}$
6)	Tardiness of job I	$T_i = \max_{1,\ldots,n} \{0, L_i\}$
7)	Earliness of job I	$E_i = \max_{1,\ldots,n} \{0, -Li\}$
8)	Maximum Earliness	$E_{max} = \max_{1,\ldots,n} \{E_i\}$
9)	The total tardiness	$\sum_{i=1}^{n} T_i$
10)	The total weighted tardiness	$\sum_{i=1}^{n} w_i T_i$
11)	Average tardiness	$\overline{T}$
12)	Maximum tardiness	$T_{max} = \max_{1,\ldots,n} \{T_i\}$
13)	Number of jobs tardy	$N_T = \sum_{i=1}^{n} \delta(T_i)$, $\delta(T_i) = 1$ if $T_i > 0$ and $\delta(T_i) = 0$ if $T_i \leq 0$

The criterions of more significance are;

- *Makespan:* It is the total amount of time required to completely process all the jobs.
- *Lateness of the Jobs:* It is defined as the difference between completion time and due date of the job.
- *Tardiness of the Jobs:* It is the maximum value of lateness of the job.
- *Mean Flow Time:* It is the average time spent by a job in the shop and comprises of processing time, waiting time and transfer time.

1.10 METHODS OF SOLUTION

Several methods have been developed to solve and model sequencing and scheduling problems that belong to any of the four categories (deterministic, static, dynamic, and stochastic). These methods of solution can be classified as follows:

1) Efficient optimal methods such as Johnson's algorithm to solve a flow shop problem with two machines and n jobs (Johnson 1954).
2) Enumerative methods (implicit and explicit or complete) such as Brown and Lomnicki's branch and bound algorithm (Brown and Lomnicki 1966).
3) Heuristic methods such as Campbell, Dudek, and Smith's algorithm to solve m machines and n jobs flow shop problems (Campbell, Dudek, and Smith 1970).
4) Mathematical models (Integer Programming) such as Wagner's form to solve the permutation flow shop problem with n jobs and m machines (Wagner 1959).
5) Heuristic search techniques: Simulated Annealing, Genetic Algorithms, Tabu Search, and Artificial Intelligence.
6) Simulation models.
7) Analytical models (such as Jackson's open queuing network model, Jackson 1957a).

1.11 DISPATCHING RULES

Over the last four decades, the sequencing and scheduling problem has been solved using dispatching rules (also called scheduling rules, sequencing rules, decision rules, or priority rules). These dispatching rules are used to determine the priority of each job. The priority of a job is determined as a function of job parameters, machine parameters, or shop characteristics. When the priority of each job is determined, jobs are sorted and then the job with the highest priority is selected to be processed first.

Baker (1974, 216-217) and Morton and Pentico (1993, 373) classified dispatching rules as follows:

⇒ **Local rules** are concerned with the local available information.

⇒ **Global rules** are used to dispatch jobs using all information available on the shop floor.

⇒ **Static rules** do not change over time, and ignore the status of the job shop floor.

⇒ **Dynamic rules** are time dependent, and change according to the status of the job shop floor.

⇒ **Forecast rules** are used to give priority to jobs according to what the job is going to come across in the future, and according to the situation at the local machine.

Several dispatching rules have been reported by many researchers. The following are some of the dispatching rules that have been developed, investigated, and implemented by several researchers and practitioners:

1) **SPT or SEPT**: Shortest Processing Time or Shortest Expected Processing Time. The job with the smallest operation processing time is processed first. The SPT rule has several versions.
   - **SRPT**: Total Shortest Remaining Processing Time.
   - **TSPT**: Truncated SPT. The job with the smallest operation processing time is processed first, but if there is a job with an operation waiting time larger than W, that job is processed first, W is arbitrarily chosen.
   - **WSPT**: Weighted Shortest Processing Time. The job with the smallest ratio is processed first. The ratio is computed by dividing the operation processing time of the job by its weight.
   - **LWR**: Least Work Remaining in terms of the number of operations.
   - **TWORK**: Total Work in terms of processing time.
   - **AJF-SPT**: Assembly jobs first with SPT rule. If there are assembly and non-assembly products waiting for a specific machine, then the assembly products are selected first. The SPT rule is used to select one of them.

2) **LPT or LEPT**: Longest Processing Time or Longest Expected Processing Time. The job with the largest operation processing time is processed first. There are other versions of LPT.
   - **TLPT**: Total LPT.
   - **LRPT**: Total Longest Remaining Processing Time.
   - **MWR**: Most Work Remaining in terms of the number of operations.

3) **EDD**: Earliest Due Date. The job with the smallest due date is processed first. There are three versions of EDD rule.
   - **ODD**: Operation Due Date. The operation with the smallest due date is processed first.
   - **MDD**: Modified Due Date. From the set of jobs waiting for a specific machine, jobs are assigned a new due date, and EDD is performed on this set. The new due dates are assigned in one of two ways. In the first, a job

with negative slack is assigned a due date that is equal to the current time plus the processing time. In the second, a job with positive slack is assigned its original due date.

- **MODD**: Modified Operation Due Date. From the set of operations waiting for a specific machine, operations are assigned a new due date, and ODD is performed on this set. This means the new operation due dates are assigned using the two ways used in the MDD, but instead of using EDD, the ODD is used.

4) **JST**: Job Slack Time. The job with minimum slack is processed first. The job slack time is computed as the difference between the job due date, the work remaining, and the current time. The JST rule has five versions.

- **OST or S/OPN**: Operation Slack Time. The job with the smallest operation slack is processed first. The OST is determined by dividing the JST by the number of job operations remaining.
- **A/OPN**: Allowance over remaining number of operation. The job with the smallest ratio is processed first.
- **S/A**: Slack time over Allowance: The job with the smallest ratio is processed first.
- **WPT+WOST**: Weighted Processing Time plus Weighted Operation Slack Time. The job with the smallest value is processed first.
- **S/RPT**: Slack over Remaining work Time. S/RPT is computed as job slack divided by the remaining work time.

5) **CR**: Critical Ratio. The job with the smallest ratio is processed first. The CR is determined by dividing job's allowance by the remaining work time. The CR has one version.

- **OCR**: Operation Critical Ratio. The operation with the smallest ratio is processed first. The OCR is determined by dividing operation's allowance by the operation process time.

6) **RANDOM**: Service in Random Order. A job is randomly selected from the set of jobs which are queued at the machine. RANDOM has one version.

- **Biased-RANDOM**: Service in Biased Random Order. When RANDOM rule is applied, jobs are equally likely to be selected from the set of jobs waiting. However, in the Biased-RANDOM rule, jobs are not equally likely to be selected. The selection process is biased according to one or more of the other dispatching rules such SPT or EDD. To apply the Biased-RANDOM to a set of jobs waiting, a dispatching rule is selected first (say SPT). Then, the set of jobs waiting are sorted according to the dispatching

rule selected (i.e., SPT). Next, jobs in the ordered list are assigned selection probabilities which are usually computed according to geometric distribution. The job in the first position will be given the largest selection probability and the job in the last position will be given the smallest selection probability. By doing so, the jobs early in the ordered list of jobs are more likely to be selected, while jobs late in the ordered list of jobs are less likely to be selected.

7) **FCFS or SORT**: First Come, First Served or Smallest Ready Time. The job which arrives first at the machine will be served first. There is one version of FCFS rule.
   - **FASFS or SRT**: First At Shop, First Served or Smallest Release Time. A job arriving first at the shop is given priority to go first in all machines.

8) **LCFS**: Last Come, First Served. The job which arrives last will be served first.

9) **LFJ**: Least Flexible Job. The job with the least flexibility is processed first.

10) **FOFO**: First Off, First On. The job with the operation that could be completed earliest will be processed first even if this operation is not yet in the queue. In this case, the machine will be idle until the operation arrives.

11) **LAWINQ**: Least Anticipated Work In Next Queue. From the set of jobs waiting for a specific machine, a job will be selected that will encounter the smallest queue at the next machine in its route.

12) **COVERT**: Cost OVER Time. COVERT is a composite rule that puts the job with the largest COVERT ratio in first position. The COVERT ratio is computed by dividing an anticipated tardiness for the associated job and its operation processing time. The COVERT rule has two versions.
   - **ATC**: Apparent Tardiness Cost. The ATC introduces the effect of job weight and it uses a different function to estimate the tardiness associated with each job. ATC gives priority to a job with the largest ATC value.
   - **ATEC**: Apparent Tardiness and Earliness Cost. ATEC is a generalization of both COVERT and ATC. It includes a different function to account for tardiness and earliness in its computations.

As mentioned earlier, the above dispatching rules are determined according to job parameters, machine parameters, and shop characteristics. The above rules can be classified into four classes. The first class consist of rules that deal with the

processing time (that is, rules 1 and 2). Rules 3, 4, and 5 are a class of rules which involve due dates. Class three consists of rules numbering 6, 7, 8, 9, 10, and 11 that involve shop and/or job characteristics. Finally, class four is formed by a combination of the other three classes and is known as rule 12.

1.11.1 Effect of dispatching rules:

Since the sequencing and scheduling problem can be viewed as a network of queues, the effect of the dispatching rules can be tested using queuing network theory. During last six decades, a series of investigations was done to continue studying the outcome of dispatching rules. Several important conclusions can be obtained from those series of studies:

1. The SPT rule minimizes the average flow time, average lateness, average number in queue, average tardiness, and percentage of jobs tardy. The SPT is insensitive to due date tightness.
2. COVERT rule is superior in minimizing the mean tardiness when compared to SPT and TSPT.
3. Job slack rules are more effective to minimize the tardiness.
4. The size of the shop is not a significant factor.
5. The FCFS rule achieves a small proportion of jobs tardy if the shop is not heavily loaded.
6. The OST minimized the percentage of jobs tardy and the conditional average tardiness.

The above conclusions have inspired researchers to study the effect of the dispatching rule in more complex and different job shop environments. Also, advancements in computer technology and software that can be used to simulate and study the job shop environment have helped researchers to do more work in this fruitful area. Thirteen studies had been performed during the seventies to investigate more difficult job shop environments. These attempts have been performed by Hottenstein (1970), Putnam et al. (1971), Ashour and Vaswani (1972), Elvers (1973, 1974), Holloway and Nelson (1974), Irastorza and Deane (1974), Eilon, Chowdhury, and Serghiou (1975), Hershauer and Ebert (1975), Berry and Finlay (1976), Eilon and Chowdhury (1976), Nelson, Holloway, and Wong (1977), Hurrion (1978), and Weeks (1979). Some of these studies will be discussed in the following paragraphs.

Hottenstein (1970) studied the process of speeding up the job delivery which is called expediting. One of two reasons can be used to accelerate jobs: 1) the due date of a job has been revised or 2) job slack has become negative. Under normal operating conditions, the SPT rule is used. When jobs belong to the expediting set of jobs, however, the jobs are processed according to either SPTEX or FCFSEX rules. The SPTEX rule gives priority to jobs according to the SPT rule, and the FCFSEX gives priority according to the FCFS rule. Hottenstein simulated hybrid job shops and pure flow shops using six machines. Two types of loads were used. Six performance measures were used: average number of jobs in the system; flow time; percentage of jobs tardy; percentage of early-request jobs shipped late;

average tardiness, and average tardiness for early-request jobs. Conclusions of this study can be summarized as follows: SPT and SPTEX rules performed almost the same under all performance measures and conditions, and the FCFSEX had the worst performance. Eilon, Chowdhury, and Serghiou (1975), performed a similar study and in their study, jobs were quickened according to an expediting criterion which was computed as the job slack combined with a control parameter (U). The control parameter was used to regulate the percentage of jobs that can be put in the set of expediting jobs. Then the SPT rule was applied to the jobs which were in the expediting set. The researchers named their general procedures the SPT* rule which was performed by first computing a classification index as follows: $F_i = JST_i - U$. Then, if $F_i \leq 0$, job i is put in the expediting set. Otherwise, job i is put in the normal set.

The effect of due date assignments was studied by Ashour and Vaswani (1972), Elvers (1973), Eilon and Chowdhury (1976), and Weeks (1979). A common conclusion of this series of studies is that dispatching rules that were due-date based performed better when due dates were assigned according to number of operations and work content of a job. In other words, these dispatching rules performed better when due dates were assigned according to expected flow time and job shop congestion. Also, the average tardiness was minimized by S/OPN rule.

Elvers (1974) investigated the effects of sixteen arrival distributions on ten dispatching rules using tardiness as the performance measure. The sixteen arrival distributions were three parameters for each of binomial distribution, bimodal distribution, discrete uniform distribution, left skew distribution, and right skew distribution, plus the Poisson distribution. Some of the dispatching rules used were: FCFS; FASFS; SRPT; SEPT; EDD; OST, and JST. Elvers simulated a job shop with eight machines, and concluded that the dispatching rules are not affected by the arrival distributions in the performance measure tested.

The effect of incorporating queueing waiting time in the calculations of both job slack time and critical ratio was investigated by Berry and Finlay (1976). They used flow time, job lateness, and work-in-process as the performance measures. They estimated the queue time using historical queue waiting times. Berry and Finlay simulated a job shop with ten machines and fifteen products. They concluded that the incorporation of queueing waiting time in the calculations of JST and CR rules did not improve the performance of these rules, which implies no improvement in the shop performance.

Using decision theory, Arumugam and Ramani (1980) compared five dispatching rules to be selected to minimize a combined criterion. This criterion consisted of work-in-process inventory and delivery performance. The five dispatching rules used were: lowest value time; highest value time; customer priority; SPT, and OST. They simulated a job shop with sixty-four machines, ninety-one workers, and nineteen products. Arumugam and Ramani simulated a job shop with various shop loads, and they concluded that the SPT dominated all dispatching rules in all job shop configurations they tested. Kanet and Zhou (1993) used decision theory to developed a dispatching rule which is called MEANP and they tested it against six other dispatching rules. The other dispatching rules used were: SPT; FCFS; ODD; COVERT; ATC, and MODD. They simulated a job shop with a single machine, and they concluded that the MEANP approach was better than all the dispatching

rules when both tardiness and flow time were the criteria.

One of the most important elements that affect the performance of dispatching rules are the due date setting rules. The effects of due date setting rules on the dispatching rules have been investigated by several researchers. These attempts have been made by Baker and Bertrand (1981, 1982), Miyazaki (1981), Baker and Kanet (1983), Baker (1984), Ragatz and Mabert (1984), Kanet and Christy (1989), Udo (1993), Vig and Dooley (1993), and Chang (1994). Several conclusions came out of these studies:

1. When job flow time estimates are used to predict due dates, the due dates produced are more robust and accurate to uncontrollable job shop (Miyazaki 1981 and Vig and Dooley 1993).
2. The relative performance of the dispatching rules was affected by the tightness of the due dates (Baker and Bertrand 1981).
3. For practicability, the best due date setting rule is the total work content (TWK) rule which provides the best results for tardiness performance measures (Baker and Bertrand 1981 and 1982, Baker 1984, and Kanet and Christy 1989). According to Kanet and Christy (1989), the TWK reduces work-in-process. TWK = kP, where k is the due date factor and P is the total work required.
4. According to Baker (1984), the second best due date setting rule is the number of operations (NOP) rule which is computed as follows: NOP = km, where k is the due date factor and m is the number of operations required by the job.
5. There is no advantage to using the slack-based dispatching rules over the simple allowance-based rule (Baker 1984).
6. When assigning due dates, both job characteristics and shop status information should be included (Ragatz and Mabert 1984, Udo 1993, Chang 1994).
7. In estimating a job due date, information about machine center congestion and the routing of the job is more useful than knowing general information about the job shop conditions (Ragatz and Mabert 1984).
8. When estimating due dates, the use of more details provides only marginal improvement in the performance of the due date setting rules (Ragatz and Mabert 1984).
9. Due dates that are assigned according to analytical analysis are favorable (Baker and Bertrand 1981).

Dar-El and Wysk (1982) investigated the effect of the release mechanism on six dispatching rules using tardiness as a performance measure. The release of jobs to the shop floor is controlled by delaying jobs according to two actions known as flushed and un-flushed. An un-flushed action was taken when no more jobs were allowed to enter the system. A flushed action is taken when all remaining jobs are completed, and all machines in the job shop are empty. The following were the dispatching rules used: SPT; FCFS; LCFS; EDD; OST, and LAWINQ. The researchers simulated a job shop with four machines which had three types of load (70%, 77%, and 85%). Dar-El and Wysk concluded that the best rules

that should be selected to manage a job shop with such behavior were SPT and LAWINQ.

The effect of dispatching rules when incorporating machine breakdowns was investigated by Muhlemann, Lockett, and Farn (1982). They tested twelve dispatching rules using seven performance measures. The twelve dispatching tested were: RANDOM; FCFS; EDD; SPT; LWR; SPT*; S/OPN; ODD; LCFS; CR; OST, and a composite rule which was developed by Farn (1979). The seven performance measures were: lateness; makespan; conditional lateness; percentage of jobs late; average queue time; mean tardiness, and average ration of flow time to process time. Their job shop had twelve machines and processed twelve products. They tested four cases of breakdowns where each had different arrival times and repair times. Also, Muhlemann, Lockett, and Farn included a rescheduling factor in their experiment. This factor was handled by having two sets of jobs waiting for any machine. The first set had the initial jobs, and the second set had the newly arrived jobs. The rescheduling was done in a certain frequency to include the newly arrived jobs in the initial set. From the results obtained, Muhlemann, Lockett, and Farn concluded that, in general, the SPT rule was the best when rescheduling was infrequent. However, the SPT* and the composite rules were far better than the SPT when rescheduling was performed frequently. The mean tardiness was minimized by JST, SPT, and EDD rules. The CR rule minimized the conditional mean lateness. Frequent rescheduling resulted in better performance for the shop.

Elvers and Taube (1983) studied the effects of efficiencies and inefficiencies of machines and workers on five dispatching rules (SPT, EDD, JST, OST, and FCFS) using percentage of jobs completed on time as a criterion. In other words, they studied the effect of workers' learning and loss of knowledge in terms of the processing times. To represent workers' learning and loss of knowledge using the processing times, the processing times were fluctuated accordingly. In their experiment, Elvers and Taube compared two cases which they called stochastic and deterministic. The stochastic case is with efficiencies and inefficiencies of machines and workers. The deterministic case is without efficiencies and inefficiencies of machines and workers. The study simulated a job shop with eight machines and six types of loads which ranged from 84.5% to 97.9% of capacity. From their results, it is clear that when the job shop is heavily loaded, SPT was superior. However, when the job shop load was under 91.6%, EDD, JST, OST, and FCFS were superior to SPT. Finally, they concluded that the incorporation of efficiencies and inefficiencies in terms of the processing did affect the performance of the dispatching rules in most situations.

Russell, Dar-El, and Taylor (1987) simulated an open job shop to test three alternative formulations of COVERT rule and ten other dispatching rules to test the effect of due date tightness. The ten dispatching rules were: FCFS; EDD; JST; S/OPN; SPT; MDD; MODD; ATC, two versions of TSPT. Eight performance measures were used: average flow time; average tardiness; average conditional tardiness; average lateness; root mean square of tardiness; root mean square of conditional tardiness; percent tardy job, and maximum tardiness. The job shop simulated, as designed by Baker (1984), consisted of four machines which had a 90% utilization level. From their results, it is clear that the SPT rule was superior in minimizing the average flow time, average lateness, and percent of job tardy. The lowest value for average conditional tardiness, root mean square of tardiness, and root mean square of conditional tardiness was achieved by COVERT rule. The MODD was superior in minimizing the average tardiness and TSPT was superior in minimizing maximum tardiness.

For loose due dates (20% tardy), MODD was superior in minimizing all performance measures except for the average flow time which was minimized by SPT. The SPT was superior in minimizing the average flow time, the maximum tardiness, and average lateness when due dates were moderate (40% tardy). Also, under tight due dates, COVERT was superior in minimizing the average conditional tardiness, the root mean square tardiness, and the root mean square conditional tardiness. The MODD was superior in minimizing the average tardiness.

Sawaqed (1987) performed a study where he investigated a hybrid assembly job shop with bottleneck machine. He investigated the effect of the position of the bottleneck machine on various performance measures. Sawaqed tried to answer several questions in his study. However, the two most important questions are:

> ...Does the location of bottleneck machines influence the relative performance of dispatching rules? Is it sufficient to manage a job shop by managing its bottleneck machines? (Sawaqed 1987, ix)

To answer these two questions Sawaqed simulated a hybrid assembly job shop with nine machines, nine products, six criteria, and six dispatching rules. The load for non-bottlenecks was 75% and for the bottleneck it was 90%. Out of the nine products, there were four assembly products. The six criteria were: average flow time; average tardiness; average lateness; average staging time; percentage of tardy, and maximum tardiness. Six dispatching rules were used (FASFS, FCFS, SPT, EDD, AJF-SPT, and SRPT). The results of this investigation concluded that the location of the bottleneck machine does not affect the relative performance of the superior dispatching rules. For example, SPT will be superior wherever the bottleneck is.

Next, Sawaqed performed another experiment to investigate the effect of managing the job shop by managing its bottleneck machines. The bottleneck machines were identified by first identifying the average utilization level of all nine machines, then the machine with over 85% utilization level was identified as the bottleneck machine. In his experiment there were three bottleneck machines with utilization level of 97%, 86%, and 95%. In terms of dispatching rules, Sawaqed developed and used four management policies to schedule jobs on bottleneck and non-bottleneck machines. These policies were: 1) EDD for both; 2) SPT for non-bottlenecks and EDD for bottlenecks; 3) EDD for non-bottlenecks and SPT for bottlenecks, and 4) SPT for both. Then Sawaqed (1987, x) concluded that “the most crucial element in managing a job shop is the management of its bottleneck machines."

Schultz (1989) developed a new rule that combined SPT with tardiness-based rules which was named CEXSPT rule. Schultz tested the CEXSPT and six other dispatching rules by simulating an open job shop that was designed by Russell, Dar-El, and Taylor (1987). The six dispatching rules used were: MODD; COVERT; SPT; ODD; S/OPN, and OCR. Four performance measures were used which were: average flow time; average tardiness; average conditional tardiness, and proportion of job tardy. Schultz concluded that the SPT was superior in minimizing the average flow time, CEXSPT was superior in minimizing average tardiness, and COVERT was superior in minimizing average conditional tardiness. Both MODD and SPT were superior in minimizing the proportion of job tardy.

Vepsalainen and Morton (1987) developed and tested the effect of the ATC rule

which considered the influence of multiple machines by using look-ahead parameters. They compared the ATC rule with five dispatching rules using three performance measures. The five dispatching rules were: FCFS; EDD; OST; WSPT, and COVERT. The four performance measures were: the normalized weighted tardiness; percentage of jobs tardy; the work-in-process, and the work-in-system. Vepsalainen and Morton simulated three types of job shops with ten machines and five shop loads (80%, 85%, 90%, 95% and 97%). Vepsalainen and Morton generalized their conclusions for the three types of job shops because of similar patterns. For all utilization and under tight due dates, they ranked the dispatching rules to minimize the weighted tardiness as follows: ATC; COVERT; WSPT; OST; EDD, then FCFS. In all utilization levels and when the due dates are loose, the ATC was ranked first to minimize the weighted tardiness, COVERT was second. When due dates are loose and the utilization is low (<90%), the OST was ranked third, but, with high utilization (>90%), the WSPT rule was ranked third. The ATC rule was the best under all utilization levels and due dates types to minimize the percentage of jobs tardy. When due dates were tight and utilization was low (<85%), COVERT performed better than WSPT, but WSPT was better when the utilization level was higher than 85%. Also, when due dates were loose and utilization was lower than 95%, COVERT performed better than WSPT, and when the utilization was higher than 95%, WSPT performed better. The WIP was minimized by the ATC and EDD rules when the utilization was high (≥ 90%) and the due dates were loose. However, when the utilization was lower than 90%, the WSPT rule was the first to minimize WIP, then the ATC and the EDD rules. In all shop loads and under tight due dates, the EDD was the best rule to minimize the WIP. The EDD and OST rules were the best for WIS under tight due dates and all utilization levels. The EDD rule was superior in all utilization levels when due dates were loose. However, when the utilization was lower than 85%, the ATC rule was ranked second, but when the utilization was higher than 85%, the OST rule was ranked second.

The computations of the ATC and COVERT rules required the computation of the expected waiting time for each operation of each job under consideration. Vepsalainen and Morton (1987) used a unique method to compute the expected waiting time which was a multiplier of the processing time of a specific job under consideration ($W=aP_{ij}$, where a: is the multiplier and P_{ij} is the processing time of operation j for job i). Therefore, Vepsalainen and Morton (1988) continued their research and investigated the effect of different estimates of the expected waiting times on ATC and COVERT. They tested three methods to estimate the waiting time, using the models of the previous study. These three methods are: multiple of processing time (STD); priority-based (PRIO), and lead-time iteration (ITER). They found that an accurate estimate of the waiting time helped the ATC and the COVERT rules to reduce the tardiness. With respect to minimizing tardiness, the ATC/ITER combination was the best minimizer and COVERT/PRIO was the second.

Anderson and Nyirenda (1990) developed two new methods to compute the operation due date. These two methods were: CR+SPT and S/RPT+SPT. Using the CR+SPT, the due date for operation j of job i is computed as follows: ODD = max ($OCR*P_{ij}$, P_{ij}). Also, the S/RPT+SPT computed an operation due date as follows: ODD = max($S/RPT*P_{ij}$, P_{ij}). Anderson and Nyirenda simulated an open job shop with eight machines to compare the performance of these two methods when they were used to computed the operation due date

in the MODD rule. Also, they compared the performance of the MODD with four other dispatching rules. These rules were: SPT; CEXSPT, and two versions of COVERT. Four performance measures were used: mean flow time; mean tardiness; proportion of tardy jobs, and conditional mean tardiness. The shop load was kept at 90% utilization level. The results of this study indicated that the SPT rule was superior in minimizing the average flow time in all due dates types, and also superior in minimizing the percentage of jobs tardy when due dates were tight. When the due dates were very tight, the MODD rule was superior in minimizing the mean tardiness. The S/RPT+SPT rule was the best to minimize the mean tardiness when due dates were moderate, and superior in minimizing the percentage of jobs tardy when due dates were loose. The CR+SPT rule was better than S/RPT+SPT rule in minimizing the average tardiness when due dates were loose.

Raghu and Rajendran (1993) developed a new dispatching rule that is sensitive to the machine utilization level, job processing time, and operation due date. Raghu and Rajendran tested their rule against six dispatching rules (SPT, EDD, MOD, ATC, S/RPT+SPT, and CR+SPT). Four performance measures were used: average flow time; average tardiness; percentage of jobs tardy, and root mean square of average tardiness. They simulated an open job shop with twelve machines and two shop loads (86% and 95%). The results of this study indicated that at 85% utilization level and in all cases of due dates and processing times, RR and SPT rules performed equally and they were the best to minimize the average flow time. However, when the utilization level was 95%, the RR rule was ranked first and SPT was second. For both the average tardiness and root mean square of average tardiness, the RR rule was superior in combinations tested. The S/RPT+SPT and the CR+SPT rules were ranked second to minimize the average tardiness. The EDD was ranked second with respect to root mean square average tardiness. For percentage of jobs tardy, the SPT rule was ranked first, the S/RPT+SPT rule was ranked second, and then the RR rule was ranked third.

A similar study to Dar-El and Wysk (1982) was recently performed by Rohleder and Scudder (1993b). In this study, four job release mechanisms were tested to minimize earliness and tardiness simultaneously. The four release rules were immediate release (IR), modified infinite loading (MIL), modified Ow and Morton (MOM), and operation early or tardy release (OETR). The release time in the IR rule was the arrival time of the job. The MIL rule derived its release time by using the attributes of jobs and the shop congestion. The MOM rule obtained its release time by using the job's due date, processing times, and early and tardy costs. The OETR rule used overall information of the job, and produced release times for each operation of each product at all machines. The OETR rule forced machines to have two queue types, which were active and inactive. The active queue kept jobs that had been released, and the inactive queue held jobs that had not yet been released. The active and inactive behavior performed by the OETR rule simulated the construction of a delay schedule and the other three release rules used a non-delay schedule. Four dispatching rules were used: FCFS; EDD; weighted COVERT, and modified ATEC. Rohleder and Scudder simulated an open job shop with six machines with three levels of utilization (70%, 80%, and 90%). The results of this study indicated that in terms of dispatching rules, the modified ATEC was superior in all cases tested. The OETR rule was the best at all utilization levels and due date types. When utilization was high, the IR was ranked second, but, as utilization levels decreased, MOM competed with IR. In terms of importance between dispatching rules

and release rules, they conclude that dispatching rules were effective in reducing early or tardy costs when the utilization was high and due dates were tight. However, release rules were effective when the utilization level was low and the due dates were loose.

Bahouth and Foote (1994) developed and implemented three dispatching rules to manage two bottleneck machines in a hybrid assembly job shop with one assembly machine. The three dispatching rules were developed by using Johnson's flow shop algorithm. The three developed rules were:

1. JNP: Johnson No Priority rule. Parts were scheduled or rescheduled in all machines according to the sequence obtained by Johnson's algorithm which was applied on the two bottlenecks.
2. JHP: Johnson Half Priority rule. Parts were scheduled or rescheduled according to JNP, but the first priority was given to a part on which only one operation was performed. If ties occurred among the jobs that were prioritized by JHP, then JNP was used. JHP was only applied before the assembly operation.
3. JFP: Johnson Full Priority rule. Parts were scheduled or rescheduled according to JNP, but the first priority was given to a part on which the maximum number of operations was performed. If ties occurred among the jobs that were prioritized by JFP, then JNP was used. JFP was applied at any machine.

Bahouth and Foote simulated a job shop with nine machines where two of them were bottlenecks. The total flow time was used as the performance measure. They studied the effect of five factors. These factors were: the interarrival times; percentage deviation between the assumed process time and the actual process time; the difference in average processing time between the two bottlenecks; the dispatching rules, and location of the bottlenecks. The performance of the three dispatching rules was compared with the performance of a superior rule, which was SPT. For the two bottlenecks, six locations were selected. These locations were: 1) the first two stages; 2) the first stage and the second-to-last stage; 3) the first and last stages; 4) the second and last stages; 5) the last two stages, and 6) the second stage and the second-to-last stage. The results of this study indicated that for the time between job creations, the JNP rule performed better than the other rules. Also, they found that for the difference in the average process time between the two bottlenecks, the SPT rule was superior when the two bottlenecks were located in the first two stages. The JNP rule performed better than the other rules when the two bottlenecks were located in the last two stages. The SPT rule performance deteriorated when there was more than one non-bottleneck machine between the two bottlenecks. Finally, Bahouth and Foote concluded the following:

> ...Flow shop sequencing rules can be applied to manage job shops: When a job shop has two bottleneck machines, a modified version of the Two-Machine Flow Shop Johnson rule can be used... The above results can only be applied to cases when the two bottleneck machines are not on parallel branches of the product structure, and when jobs use the two bottleneck machines in the same sequence. (Bahouth and Foote 1994, 2476)

EXERCISES

1.1 Give at least two practical examples of following environments.
   a. Single Machine shop
   b. Flow shop.
   c. Job shop.
   d. Assembly job shop.
   e. Hybrid job shop.
   f. Open shop.
   g. Closed shop

1.2 What is the difference between stochastic and deterministic sequencing and scheduling problems?

1.3 What are the three common types of decision-making goals in sequencing and scheduling problems?

1.4 How could we achieve efficient utilization of machine with respect to sequencing and scheduling criteria?

1.5 What are different methods of solving the sequencing & scheduling problem?

1.6 Explain the effects of following dispatching rules.
   a. SPT
   b. LPT
   c. EDD
   d. WSPT
   e. CR
   f. FCFS

1.7 Al-harkan factory makes large precision gears used in heavy equipment. These are made by drop forging and they have two identical drop forges. Currently, there are five gears that must be finished by tomorrow morning. The factory only has one shift of workers but will pay them overtime to finish the gears. All workers in the shift stay until all jobs are completed. The times to process the five gears are 3, 7, 2, 15, and 4 hours respectively.
   a. Define the scheduling problem environment that satisfies the objective of finishing gears in shortest time!
   b. Suppose that Al-harkan factory receives payment as soon as the gears are delivered. Define the scheduling problem environment that satisfies the objective of receiving payment as soon as the gears are finished!

2

Single Machine Scheduling

CHAPTER CONTENTS

2.1 INTRODUCTION

Single machine scheduling has attained most attention in theoretical scheduling studies. The understanding of the theories paves way for analyzing and better designing multi- machine systems. The following assumptions generally apply to build single machine scheduling models.

1. Machine is continuously available during scheduling period.
2. The machine processes jobs one at a time.
3. The process time of each job on the machine is accurately known and, it does not depend upon prior jobs.
4. The process time includes both set up time and actual machining time.
5. The other job related information is known before hand. This information may include due date of the job (d_j) and release time of the job (r_j).
6. In non-preemptive scheduling, jobs finish processing without interruption. In preemptive scheduling, jobs may be removed from the machine without finishing the operation.

2.2 SCHEDULING MATHEMATICS

The scheduling parameters for a typical job, say job j, are defined as follows:

- p_j = the processing time Job j
- S_j = the start time of job j
- W_j = the waiting time of job j
- W_j = the waiting time of job j
- D_j = the due date of job j
- Ej = the earliness job j
- r_j = the release time job j
- C_j = the completion time of job j
- F_j = the flow time of job j
- L_j = the lateness of job j
- T_j = the tardiness of job j

2.2.1 Gantt Chart

Gantt chart is a popular way of graphically presenting a schedule of jobs on machines. X-axis of the chart represents time and, rectangular block on y-axis represents machine. A horizontal bar shows each job's start and finish time on a particular machine. The job number is inscribed in a rectangle. The length of the rectangle is scaled to represent job's process time. The start and finish time of a job are indicated at the starting and terminating vertical sides of the job rectangle. Bars representing machines also indicate idle intervals on the machine. The following Gantt chart presents single machine with n jobs.

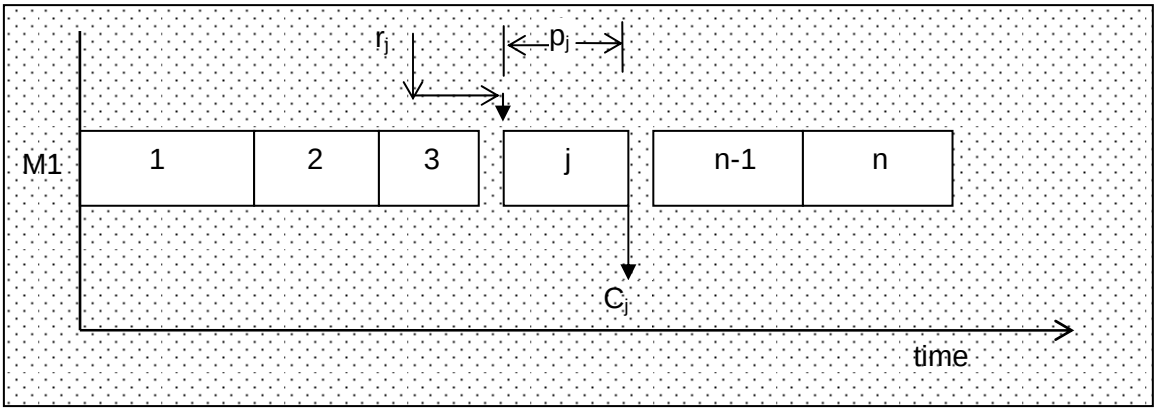


Figure 2.1 Gantt chart.

From the Gantt chart, waiting time for job j will be: $W_j = C_j - r_j - p_j$ Similarly, the lateness (L_j) of the job j will be; $L_j = C_j - d_j$. Job tardiness T_j is defined as positive lateness. In mathematical terms, tardiness (T_j) is expressed as follows:

$$T_j = L_j, \quad \text{if } L_j > 0$$
$$= 0, \quad \text{otherwise}$$
$$T_j = \max(L_j, 0)$$

Similarly, job earliness is negative lateness. In mathematical terms, E_j is expressed as

$$E_j = L_j, \quad \text{if } L_j < 0$$
$$= 0, \quad \text{otherwise}$$
$$E_j = \max(-L_j, 0)$$

Job Flow time (F_j) can be expressed in two ways:

$$F_j = C_j - r_j$$
$$= W_j + p_j$$

From the relationship between r_j, W_j and p_j, expression for C_j can be deduced;

$$C_j = r_j + W_j + p_j$$
$$= S_j + p_j$$

From the above relationship, $S_j = r_j + W_j$. Clearly, if $W_j = 0$ then $S_j = r_j$.
Start time for generating a schedule is;

$$S_j = C_{j-1} \quad \text{if} \quad C_{j-1} > r_j$$
$$= r_j \qquad \text{otherwise,}$$

Example 2.1

The following table contains data pertaining to $1||\bar{F}$ problem.

Job (j)	1	2	3	4
p_j	4	2	6	5

Note that the release time $r_j = 0$ for all jobs. Hence, it is a static shop environment. Use the following sequences and find average waiting time $\overline{W}$ and the average flow time $\overline{F}$.

i) Numerical (natural) Job order sequence (1-2-3-4)
ii) Shortest Process Time (SPT) Sequence (2-1-4-3)
iii) Random Sequence (1-3-4-2)

Solution:

i) Numerical Order Sequence (1-2-3-4)

Using the numerical or natural order sequence, the schedule is shown using Gantt chart below.

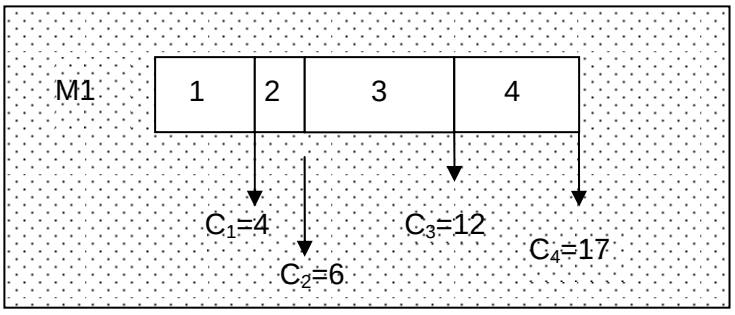


Figure 2.2 Gantt chart for numerical order sequence.

The waiting times for all the jobs is computed as shown in the following table

Job (j)	**p_j**	**S_j**	**C_j**	**W_j**	**F_j**
1	4	0	4	0	4
2	2	4	6	4	6
3	6	6	12	6	12
4	5	12	17	12	17

$$\overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = \frac{0 + 4 + 6 + 12}{4} = \frac{22}{4} = 5.5, \qquad \overline{F} = \frac{\sum F_j}{4} = \frac{39}{4} = 9.75$$

ii) SPT Sequence: (2-1-4-3)

For each job, the computations for the waiting time using the SPT sequence are shown below.

Job (j)	p_j	S_j	C_j	W_j	F_j
2	2	0	2	0	2
1	4	2	6	2	6
4	5	6	11	6	11
3	6	11	17	11	17

$$\overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = \frac{0 + 2 + 6 + 11}{4} = \frac{19}{4} = 4.75, \qquad \overline{F} = \frac{\sum F_j}{4} = \frac{36}{4} = 9,$$

iii) Random Sequence: (1-3-4-2)

When using the random sequence, the average waiting time calculations is given in the table below;

Job (j)	p_j	S_j	C_j	W_j	Fj
1	4	0	4	0	4
3	6	4	10	4	10
4	5	10	15	10	15
2	2	15	17	15	17

$$\overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = \frac{0 + 4 + 10 + 15}{4} = \frac{29}{4} = 7.25, \qquad \overline{F} = \frac{\sum F_j}{4} = \frac{46}{4} = 11.50$$

Note that the average values obtained by the SPT sequence are minimum for both $\overline{W}$ and $\overline{F}$.

Example 2.2

Consider a single machine sequencing problem with data as shown below:

Job(j)	1	2	3	4
p_j	4	2	6	5
d_j	8	12	11	10

Use the following sequences to find Makespen or maximum completion time C_{max}, Average waiting time $\overline{W}$, Average tardiness $\overline{T}$, and maximum lateness L_{max}.

i) SPT sequence, ii) EDD sequence

Solution:

i) Job sequence based on SPT is (2-1-4-3). The computations for SPT sequence are presented in the following table

Job(j)	p_j	S_j	C_j	d_j	W_j	L_j	T_j
2	2	0	2	12	0	-10	0
1	4	2	6	8	2	-2	0
4	5	6	11	10	6	1	1
3	6	11	17	11	11	6	6

$$C_{max} = \sum_{j=1}^{j=4} p_j = 17, \quad \overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = 4.75, \quad \overline{T} = \frac{\sum_{j=1}^{4} T_j}{4} = 1.75, \quad L_{max} = 6$$

ii) Job sequence based on earliest due date (EDD) is (1-4-3-2). The computations for EDD sequence are presented in table below

Job(j)	p_j	S_j	C_j	d_j	W_j	L_j	T_j
1	4	0	4	8	0	-4	0
4	5	4	9	10	4	-1	0
3	6	9	15	11	9	4	4
2	2	15	17	12	15	5	5

$$C_{max} = \sum_{j=1}^{j=4} p_j = 17, \quad \overline{W} = \frac{\sum_{j=1}^{4} W_j}{4} = 7, \quad \overline{T} = \frac{\sum_{j=1}^{4} T_j}{4} = 2.25, \quad L_{max} = 5$$

It should be noted that the value of C_{max} remains the same for both sequences. Also, the value for L_{max} for EDD sequence is smaller than the value of L_{max} for SPT sequence. In addition, the average waiting time and the average tardiness for the SPT sequence are very small when compared the values obtained by the EDD sequence. **What can you infer about these results?**

2.3 MINIMIZATION OF THE MAXIMUM LATENESS PROBLEM *(1||L_{max})*

For single machine problems, if due dates (d_j) are specified, then earliest due date (EDD) sequence yields an optimal solution to the maximum lateness L_{max} and the maximum tardiness T_{max}. This rule applies to the class of problems specified by $1||L_{max}$ or $1||T_{max}$ terminology. In such class of problems, it is implicitly assumed that release time of all jobs; $r_j \equiv 0$ (static shop)

Example 2.3

Find an optimal sequence for $1 \,||\, L_{max}$ problem. The data is given in the table below. Release time of all jobs is zero; i.e., $r_j = 0$ $(1 \le j \le 6)$. Compute the maximum lateness L_{max} and the average lateness $\overline{L}$.

Job (j)	1	2	3	4	5	6
p_j	10	3	4	8	10	6
d_j	15	6	9	23	20	30

Solution:

The EDD sequence of the 6-job problem from table is;

Job (j)	2	3	1	5	4	6
p_j	3	4	10	10	8	6
D_j	6	9	15	20	23	30

The calculations of Completion times (C_j) and Lateness (L_j) are shown below:

Job (j)	2	3	1	5	4	6
C_j	3	7	17	27	35	41
L_j	-3	-2	2	7	12	11

Maximum Lateness, $L_{max} = \max \{ \max\{L_j, 0\}; j = 1, \cdots 6 \} = 12$ and the average lateness; $\overline{L} = \dfrac{\sum_{j=1}^{6} L_j}{6} = \dfrac{27}{6} = 4.5$.

2.4 MINIMIZATION OF TOTAL WEIGHTED COMPLETION TIME PROBLEM *(1|| $\Sigma\omega_j C_j$)*

Often the jobs in a shop have priorities attached on their tags which are specified by the term ω_j. To schedule such jobs, Weighted Shortest Processing Time (WSPT) sequence is applied. As a first step to perform WSPT sequence, calculate process time to weight ratio for each job, and, then rank jobs in increasing order of processes time to weight ratio values.

Example 2.4 (WSPT Sequence)
Assume the weight assigned ω_j for each job j (importance or priority) and then solve the following 4-jobs problem using WSPT sequence. Also, compute the total weighted completion times.

Job(j)	p_j	ω_j	p_j/ω_j	d_j
1	4	8	0.5	8
2	2	7	0.285	12
3	6	3	2	11
4	5	15	0.333	10

Solution: Jobs Sequence based on WSPT $\rightarrow$ (2-4-1-3)

Job(j)	p_j	ω_j	S_j	C_j	d_j	W_j	L_j	T_j	$\omega_j C_j$
2	2	7	0	2	12	0	-10	0	14
4	5	15	2	7	10	2	-3	0	105
1	4	8	7	11	8	7	3	3	88
3	6	3	11	17	11	11	6	6	51

Hence, from the above table, the total weighted completion $\Sigma\omega_j C_j$ can be found which is 388.

2.5 MINIMIZATION OF TOTAL WEIGHTED COMPLETION TIME PROBLEM WITH PRECEDENCE RELATIONS *(1| PREC| $\Sigma\omega_j C_j$)*

There are instances when jobs have precedence relationships. As shown in precedence network diagram below, job 3 depends on job 2 and, job 2 depends on job 1.

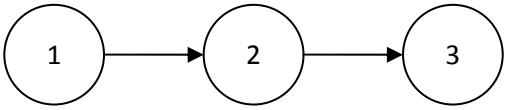


Figure 2.3 Example of precedence network.

In the following example, a single machine problem is presented where jobs have precedence relationship. The objective function is the minimization of total weighted completion time ($\sum_{j=1}^{n} \omega_j C_j$). The solution methodology is based on *Chain method* which is described below.

1. For each set of jobs in the precedence network diagram, form job sets from unscheduled jobs for each chain.
2. Find minimum value of ρ-factor for each chain, where ρ-factor = $\frac{\sum p_j}{\sum \omega_j}$
3. Select the jobs from the chain having overall minimum value of ρ -factor.
4. Include these jobs in the partial schedule, and delete them from the network diagram.
5. Repeat steps (1) to (4) until all jobs are scheduled.

Example 2.5 (1| *prec* | $\Sigma\omega_j C_j$)

Consider the following problem as an instance of the 1| *prec* | $\Sigma\omega_j C_j$. A 7-job single machine data with job precedence constraints graph is given below.

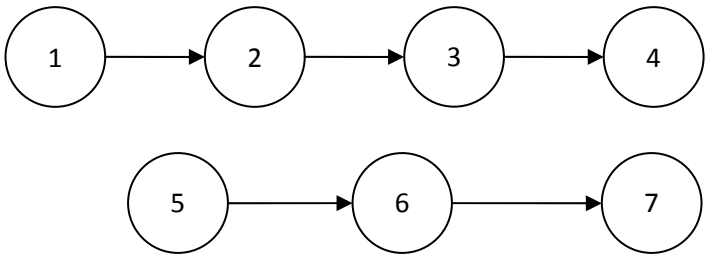


Figure 2.4 Precedence constraints graph.

The weights and process times of the jobs are given in the following table.

Jobs	1	2	3	4	5	6	7
p_j	3	6	6	5	4	8	10
ω_j	6	18	12	8	8	17	18

Solve the problem to minimize total weighted completion times using chain-method.

Solution:

Apply *Chain method* as follows:

For each job set, find ρ-factor $= \frac{\sum p_j}{\sum \omega_j}$ ratio for all job sets:

Suppose job set is: $\{1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4\}$

Then, ρ-factor $= \frac{\sum p_j}{\sum \omega_j} = \frac{\sum p_1 + p_2 + p_3 + p_4}{\sum \omega_1 + \omega_2 + \omega_3 + \omega_4} = \frac{3+6+6+5}{6+18+12+8} = \frac{20}{44} = 0.455$

The ρ-Factor for chain 1

Job Set	1	$1 \longrightarrow 2$	$1 \longrightarrow 2 \longrightarrow 3$	$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4$
ρ-Factor	$\frac{3}{6}$	$\frac{9}{24}$	$\frac{15}{36}$	$\frac{20}{44}$
	0.5	0.375	0.416	0.455

Minimum value for ρ-Factor is 0.375 for the job set $\{1 \longrightarrow 2\}$.

The ρ-Factor for Chain 2

Job Set	5	$5 \longrightarrow 6$	$5 \longrightarrow 6 \longrightarrow 7$
ρ-Factor	$\frac{4}{8}$	$\frac{12}{25}$	$\frac{22}{43}$
	0.5	0.480	0.512

Minimum value of ρ-Factor is 0.480 for the job set $\{5 \longrightarrow 6\}$.

Comparison of the ρ-Factor of Chain-1 and Chain-2.

	Min ρ-Factor	Job Set
Chain-1	0.375	$1 \longrightarrow 2$
Chain-2	0.48	$5 \longrightarrow 6$

Overall minimum value of ρ-Factor is 0.375 for Chain-1 and job set $\{1 \longrightarrow 2\}$. Thus, the following partial sequence is obtained: {1-2-X-X-X-X-X-X-X}. Then, jobs 1 and 2 should be marked on the network diagram.

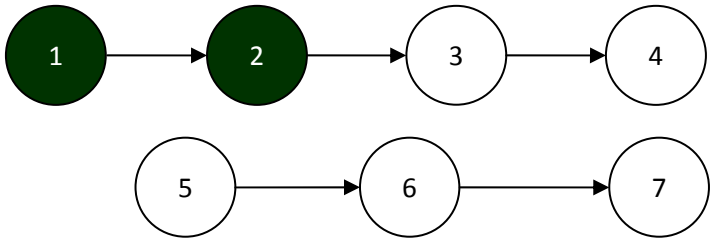


Figure 2.5 Precedence constraints graph when marking Jobs 1 and 2.

The unscheduled jobs in Chain-1 are now; 3 and 4. The corresponding two job sets are; {3} and {3 $\longrightarrow$ 4}.
Then, the revised ρ-Factor values for these 2 jobs are calculated as follows:

Job Set	3	3 $\longrightarrow$ 4
ρ-Factor	$\frac{6}{12}$	$\frac{11}{20}$
	0.5	0.55

Next, compare the ρ-factor for the two chains as follows:

	Min ρ-Factor	Job Set
Chain-1	0.5	3
Chain-2	0.48	5 $\longrightarrow$ 6

Overall minimum value of ρ-Factor is 0.48 which belong to Chain-2 for Job Set: {5 $\longrightarrow$ 6}. Thus, the Partial Sequence is {1-2-5-6-X-X-X}. Mark jobs 5 and 6 on the network diagram.

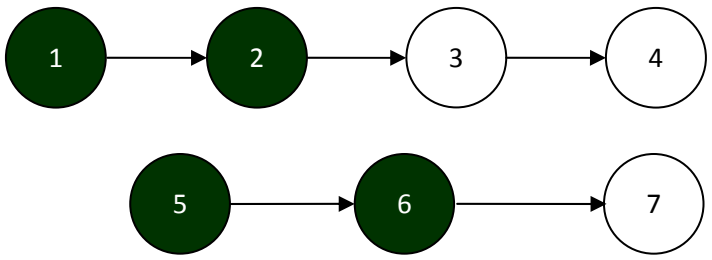


Figure 2.6 Precedence constraints graph when marking Jobs 1, 2, 5, and 6.

Chains			Min ρ_factor	Job Set
Chain-1	Job Set :{3} ρ_factor : 0.5	Job Set : {3 ⟶ 4} ρ-Factor : 0.55	0.5	{3}
Chain-2	Job Set :{7} ρ-Factor : 0.55		0.55	{7}

Minimum value of ρ-Factor is 0.5 which belong to Chain-1 for Job Set: {3}. Thus, the Update for the Partial Sequence is as follows: {1-2-5-6-3-X-X}. Next, mark job 3 on the network diagram as shown below.

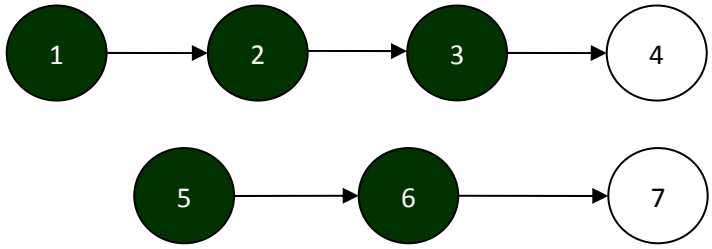


Figure 2.7 Precedence constraints graph when marking Jobs 1, 2, 5, 6, and 3.

For the remaining jobs in Chain-1 and Chain-2 are Job 4 and job 7 respectively. Thus, the Minimum value of ρ-Factor can be computed as shown below.

Chain		Min ρ_factor	Job Set
Chain-1	Job Set :{4} , ρ_factor : 0.625	0.625	{4}
Chain-2	Job Set :{7} , ρ-Factor : 0.55	0.55	{7}

The minimum value of ρ-Factor is 0.55 which belong to Chain-2 for Job Set: {7}. The partial sequence can be updated as follows: {1-2-5-6-3-7-X}. Next, mark job 7 on the network diagram as shown below.

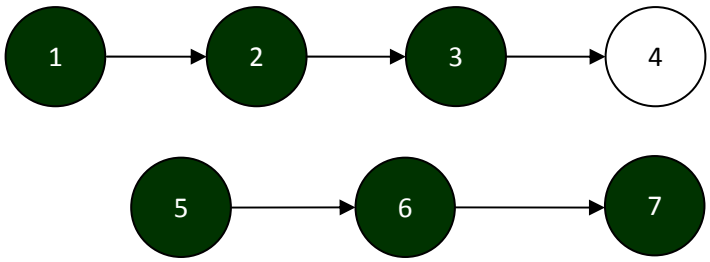


Figure 2.8 Precedence constraints graph when marking Jobs 1, 2, 5, 6, 3, and 7.

There is only one unscheduled job which is job 4. Thus, this job can be attached to the end of the partial sequence which means the final sequence is being developed and it is {1-2-5-6-3-7-4}. Using this final sequence the total weighted completion can be computed as follows:

Job(j)	p_j	ω_j	S_j	C_j	$\omega_j C_j$
1	3	6	0	3	18
2	6	18	3	9	162
5	4	8	9	13	104
6	8	17	13	21	357
3	6	12	21	27	324
7	10	18	27	37	666
4	5	8	37	42	336
				Σωj Cj	1967

2.6 NON-PREEMPTIVE SCHEDULING

This type of scheduling is considered when release time of all jobs in the shop is same. Sequence of jobs is decided by scheduling policy. Once, the sequence is determined, the jobs are loaded on the machine accordingly. The jobs are continuously processed over the machine in the sequencing order. No change in sequencing order is made once machine starts processing the jobs. Every job is processed on the machine on its turn. No job is removed from the machine during its stay on the machine even if a high priority job arrives on that machine. Hence, no interruption is allowed during processing of the jobs and, all jobs complete their processing according to pre-defined sequence.

Example 2.6

Data pertaining to 4-job single machine problem is given in the following table.

Job(j)	p_j	r_j	d_j
1	4	0	8
2	2	3	12
3	6	3	11
4	5	5	10

Generate a non-preemptive schedule using EDD sequence, and then compute the C_{max} and the L_{max}.

Solution:

Since, jobs have distinct r_j values, the problem data presents a dynamic environment. EDD Sequence for the 4-jobs based on due dates is {1-4-3-2}. The following Gantt chart gives complete information on the EDD sequence.

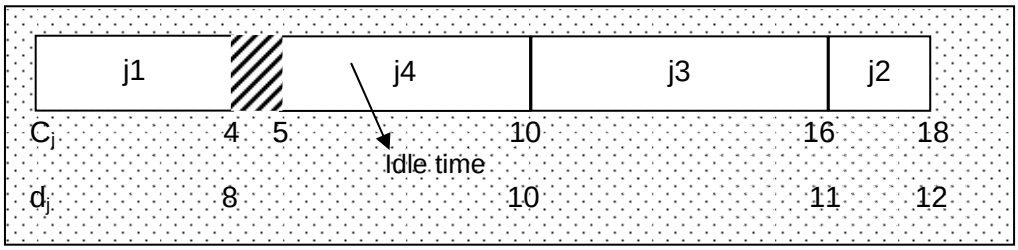


Figure 2.9 Gantt chart for the EDD sequence.

Job 4 arrives at time 5, where as the machine is free at time 4. Hence, there is an idle time from time 4 to time 5. The makespan for the sequence is 18 unit of time. The L_{max} = max {-4, 0, 5, 6} = 6. If the release time of all jobs would have been $r_j = 0$, then, the makespan is 17 unit of time ($\Sigma p_j = 17$). Also, in this case, the maximum lateness L_{max} would have been 5.

2.7 PREEMPTIVE SCHEDULING

This type of scheduling is considered when jobs arrive in the shop at different times; i.e., the values of r_j is greater zero and it could be different for each job. In this case, jobs are scheduled according to a pre-determined sequence. However, flexibility is built in the sequencing. This means a job can be removed from the machine if a high priority job is to be processed ahead of currently scheduled job. Then, the low priority jobs or preempted jobs are processed later on.

Example 2.7

Consider example 2.6 once again. If preemption of scheduled jobs is allowed in which the priority is given for jobs with earliest due date (EDD). What will be the new schedule? Is it a better schedule (Why or why not?)

Solution:

The EDD Sequence is {1-4-3-2}

Time, t=0.

Job arrived at time zero is job 1. Job 1 is scheduled at time zero. Then, the machine will complete job1 by time 4.

Time, t=3.

Jobs 2 and 3 arrived in the shop. The machine is busy and can not process any job.

Time, t=4.

Processing of job 1 is completed and the machine becomes free and ready to process any of the waiting jobs. Referring to the EDD sequence, the next job in EDD sequence is job 4, however, it has not arrived yet because its arrival time is at time 5. Thus, the next job in the EDD sequence is picked up which is job 3. Since job3 has arrived already then it is assigned to the machine.

Time, t=5.

Job j4 has arrived in the shop. Currently the machine is processing job 3. From the EDD sequence, it is clear that job 3 has lower priority than job 4. Thus, preempt job 3 from being processed on the machine. Then, assign job 4 to the machine. Job j_4 has a process time of 5 unit of time which means it will be completed at time 10.

Time, t=10

Processing on job 4 is completed. Then, the next job in the EDD sequence is job 3. This job has already been partially processed from time t=4 to t=5. Its remaining process time is 5 unit of time. Thus, assign job 3 to the machine and will be completed at time 15.

Time, t=15

Processing on job j_3 is completed. Then, the next job in the EDD sequence is job 2. Its process time is 2 unit of time. Next, assign job 2 to the machine and be completed at time 17.

Time, t=17.

Processing on job 2 is completed. There is no other job to be processed, hence the procedure is STOP.

The following Gantt chart presents the schedule of the jobs

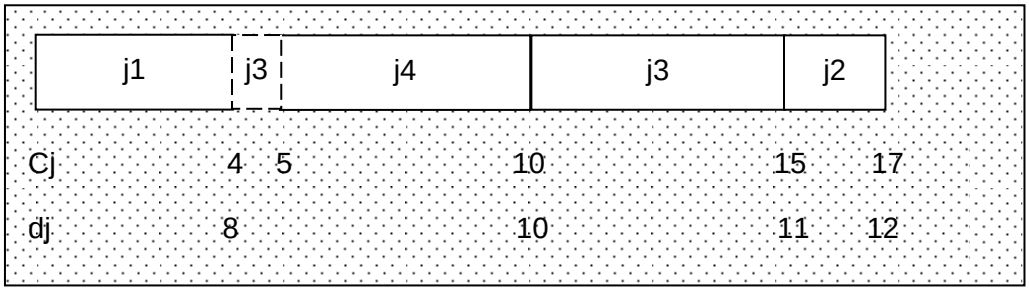


Figure 2.10 Gantt chart for the preemptive EDD schedule.

From the Gantt chart above, it should be clear that the values of makespan and maximum lateness are as follows: makespan =17 and L_{max}=5.

It should be clear that for this problem data, the preemptive scheduling provides a better solution than non-preemptive schedule.

2.8 NUMBER OF TARDY JOBS *(1 || n_t)*

A popular scheduling objective is minimization of number of tardy jobs. It is expressed as 1 || n_t problem in scheduling terminology. Hodgson's Algorithm provides an optimal solution for this type of problem. Hodgson's Algorithm has the following steps:

Step 1: Order the jobs in EDD sequence.

Step 2: Build the Gantt chart for the EDD sequence and compute T_j values for all jobs; $(1 \le j \le n)$

Step 3: If T_j values for all jobs are zero (this means there is no tardy job). STOP. The EDD sequence is an optimal sequence for the scheduling problem under consideration 1|| n_t Otherwise continue to Step 4.

Step 4: For the current sequence, find the first tardy job in the sequence, say k.

Step 5: Remove job j among the scheduled job so far $(1 \le j \le k)$ with longest process time from the sequence and put it in the set of tardy jobs. Then, Go To Step 2.

Example 2.8

Solve the following 1 || n_t problem in which the following data is given:

Job (j)	1	2	3	4	5	6
p_j	10	3	4	8	10	6
d_j	15	6	9	23	20	30

Solution:

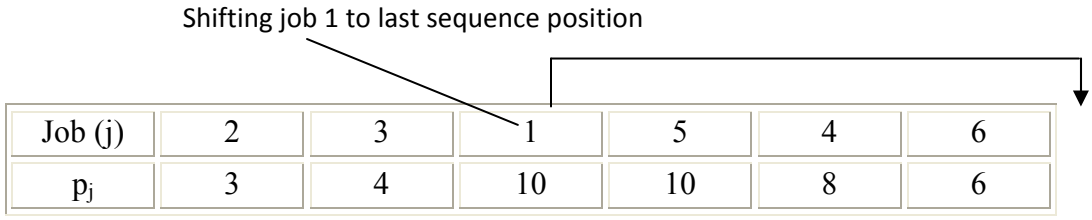
Rearrange jobs according to EDD sequence as follows;

Job (j)	2	3	1	5	4	6
p_j	3	4	10	10	8	6
d_j	6	9	15	20	23	30

The calculations of completion times (C_j) and tardiness (T_j) are shown in the table below. The number of tardy jobs; n_t is equal to 4.

Job (j)	2	3	1	5	4	6
p_j	3	4	10	10	8	6
C_j	3	7	17	27	35	41
L_j	-3	-2	2	7	12	11
T_j	0	0	2	7	12	11

From Step 4, the first tardy job in the sequence is job 1. According to Step 5, jobs 2, 3, and 1 are candidates to be removed to the set of the tardy jobs. Since job 1 has the largest processing time p_j. Then, remove job 1 from the current scheduled and put it in the set of tardy jobs which means job 1 will be attached to the end or in the last position in the sequence.

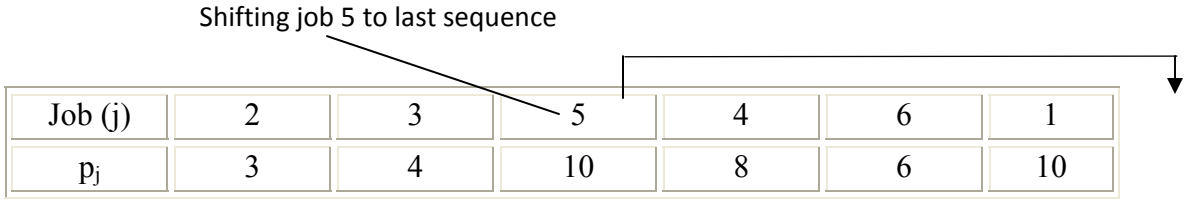


Job (j)	2	3	1	5	4	6
p_j	3	4	10	10	8	6

Recalculate T_j and C_j values as shown below in Table.

Job (j)	2	3	5	4	6	1
p_j	3	4	10	8	6	10
d_j	6	9	20	23	30	15
C_j	3	7	17	25	31	41
L_j	-3	-2	-3	2	1	26
T_j	0	0	0	2	1	26

The first tardy job in new schedule is job 4 (k = 4). From the scheduled job set {2 – 3 – 5 – 4}, job 5 has largest processing time p_j. Thus, shift job 5 to last position in the sequence as shown below.



Job (j)	2	3	5	4	6	1
p_j	3	4	10	8	6	10

After shifting job 5 to last sequence position, recalculate Tj and Cj as shown in Table below.

Job (j)	2	3	4	6	1	5
p_j	3	4	8	6	10	10
D_j	6	9	23	30	6	20
C_j	3	7	15	21	31	41
L_j	-3	-2	-8	-9	25	21
T_j	0	0	0	0	25	21

It should be clear that there is no tardy job in the scheduled job set. Hence, there are only two tardy jobs; namely, jobs 1 and 5. Therefore, the total number of tardy jobs; n_t is equal to 2. The following are the two optimal sequences for this problem: {2 – 3 – 4 – 6 – 1 – 5} and {2 – 3 – 4 – 6 – 5 – 1}

2.9 BRANCH & BOUND METHOD

The branch and bound method uses a sequence tree. Each node in the tree contains a partial sequence of jobs. For n job problem, there are n-1 numbers of levels for a tree. At level zero, root node will be placed with all n empty sequence positions. At level 1, there will be n number of nodes. Each node will contain a partial sequence of jobs. The first position in the sequence will be occupied by a job in numerical order. Similarly, each node at $(n-1)^{th}$ level will be branched to (n-2) number of nodes. The process will continue till each node has exactly one *leaf*. The construction of a tree for generating all sequences for a 3-job problem is presented in the Figure 11.

Generation of all sequences is combinatorial in nature and, will result in enormous number of sequences even for a small number of jobs. For example, for a 10-job problem there will be 10! Sequences. To reduce the computational effort, lower bounds are calculated at every level for each node. The formula used to compute the lower bound is pertinent to objective function of the scheduling problem. Branching is carried out only from those nodes with the minimum lower bound. By doing so, only small proportion of the nodes is explored resulting in fewer amounts of computations. The branch and bound (B&B) method is applied in almost every scheduling problem. In the following paragraph, this methodology is applied to solve single machine problems where jobs have distinct release times (r_j) and objective function is to minimize L_{max}. As scheduling terminology implies, these types of problems as termed as $1 \mid r_j \mid L_{max}$ problem.

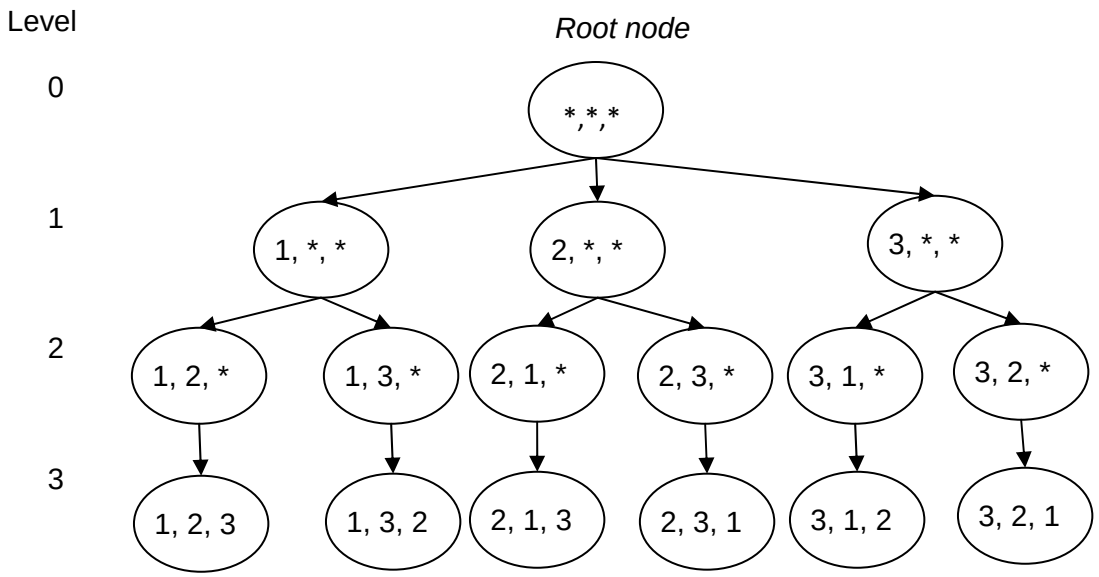


Figure 2.11 Sequence Generation tree.

2.10 MINIMIZATION OF MAXIMUM LATENESS WITH READY TIME PROBLEM *(1 | r_j | L_{max})*

In this section, the branch and bound (B&B) method is used to solve the following scheduling problem: 1 | r_j | L_{max}. The following guidelines should be followed when developing the scheduling generating tree for the branch and bound for the problem under consideration:

1. At any level of the tree, generate only those child nodes for the selected parent node which satisfies the following relationship:

$$r_j < \min_{k \in J} \{\max(t, r_k) + p_k\}$$

Where:

J = set of jobs not yet scheduled

t = time at which job j is supposed to start.

This mean only those jobs which are ready to be processed will be considered.

2. Compute the lower bounds value for all nodes at any level using **preemptive EDD scheduling**. This means, at every node the L_{max} is computed using the developed preemptive EDD schedule. Then, pick the node(s) (sequence(s)) having minimum L_{max} value for further branching to lower level.

The following example will demonstrate the implementation of the branch and bound solving the following scheduling problem: 1 | rj | L_{max}.

Example 2.9

The following Table presents an instance of 1 | rj | L_{max} problem. Find an optimal solution for this problem using the branch and bound method.

Job (j)	1	2	3	4
p_j	4	2	6	5
d_j	8	12	11	10
r_j	0	1	3	5

Solution:

At the start, there are four possible nodes at Level 1 as shown in the figure below.

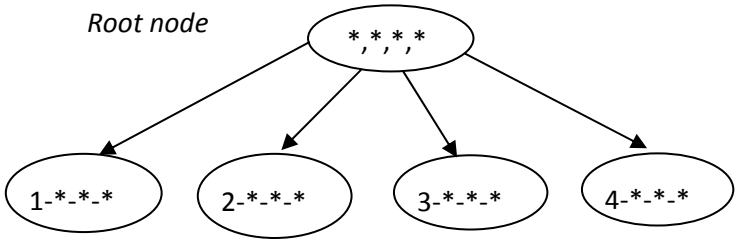


Figure 1 Level 1 Nodes for 4-job Problem.

Iteration I – Step (1)

For the four nodes at level 1, verify the condition;

$$r_j < \min_{k \in J} \{\max(t, r_k) + p_k\}$$

The following paragraphs will consider each partial sequence one by one as follows:

a) Partial Sequence: (1-*-*-*)

$r_1 = 0$ t=0

Unscheduled jobs are (2-3-4). The minimum completion time for all these jobs can be computed as follows.

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
2	1	1	2	3
3	3	3	6	9
4	5	5	5	10
			Min	3

Since $r_1 < 3$, then, include this node (1-*-*-*) in the tree.

b) Partial Sequence: (2-*-*-*)

$r_2 = 1$ t=0

Unscheduled jobs are (1-3-4). The minimum completion time for all these jobs can be computed as follows.

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
1	0	0	4	4
3	3	3	6	9
4	5	5	5	10
			Min	4

Since $r_2 < 4$, then, include this node (2-*-*-*) in the tree.

c) Partial Sequence: (3-*-*-*)

$r_3 = 3$ t=0

Unscheduled job are (1-2-4). The minimum completion time for all these jobs can be computed as follows:

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
1	0	0	4	4
2	1	1	2	3
4	5	5	5	10
			Min	3

Since r_3 is not less than 3. Then, do not include this node (3-*-*-*) in the tree.

d) Partial Sequence: (4-*-*-*)

$r_4 = 5$ t=0

Unscheduled jobs are (1-2-3). The minimum completion time for all these jobs can be computed as follows:

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
1	0	0	4	4
2	1	1	2	3
3	3	6	5	11
			Min	3

Since r_4 is not less than 3. Then, do not include this node (4-*-*-*) in the tree.

Iteration – I: Step (2)

Find L_{max} for the sequences (1-*-*-*) and (2-*-*-*) as follows:

EDD sequence based on the due dates is as follows: (1-4-3-2)

Schedule for the sequence (1-*-*-*) using preemptive scheduling is as follow:

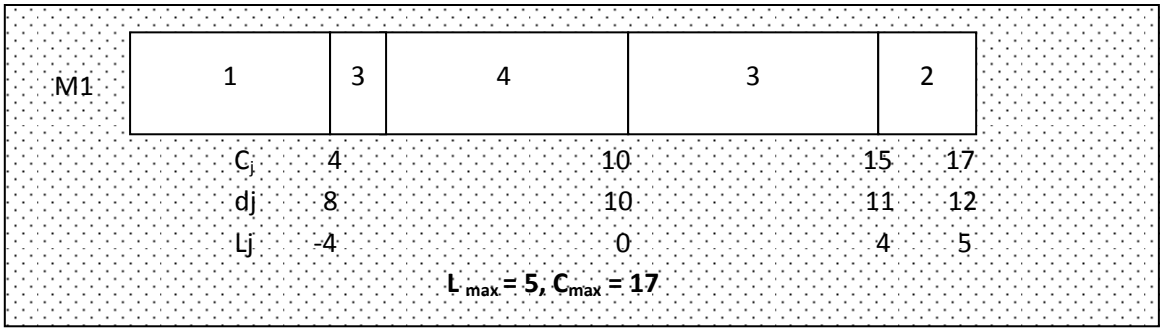


Figure 2.13 Gantt chart for the partial sequence (1-*-*-*).

Schedule for the sequence (2-*-*-*) using preemptive scheduling is as follows:

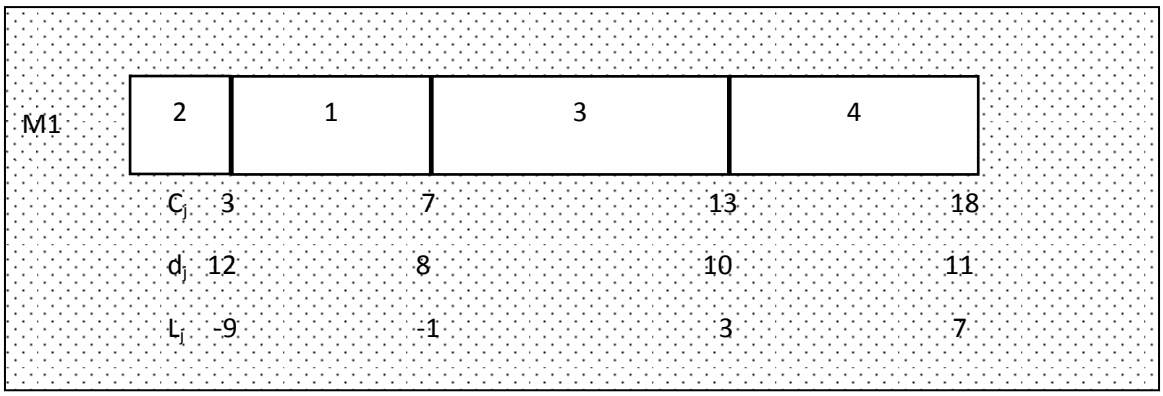


Figure 2.14 Gantt chart for the partial sequence (2-*-*-*).

From Figure 13, it should be clear that the machine has been idle during the time interval [0, 1] because the first job in sequence is job 2 and its ready time is 1. The lower bound (LB) for node (1-*-*-*) is lower than the LB for node (2-*-*-*). This means, branching should be continued from node (1-*-*-*). The following figure shows the branch from node (1-*-*-*).

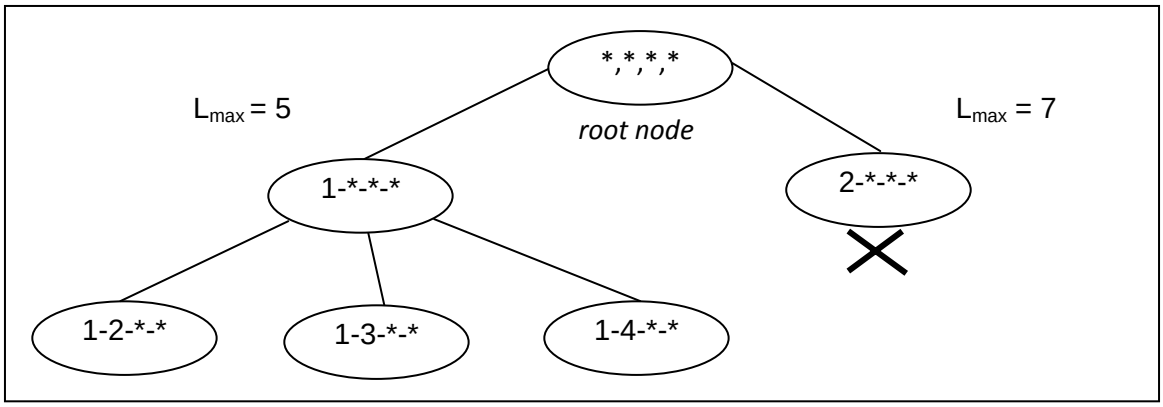


Figure 2.15 Level 2 Nodes for 4-job Problem.

It should be clear from the figure above, there are three nodes emanating from node (1-*-*-*). Next, verify the following condition in order to determine which node(s) to consider:

$$r_j < \min_{k \in J} \{\max(t, r_k) + p_k\}$$

Iteration 2 – Step (1)

a) Partial Sequence: (1-2-*-*)

r_2 = 1 t=4

Unscheduled jobs are (3-4). The minimum completion time for all these jobs can be computed as follows:

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
3	3	4	6	10
4	5	5	5	10
			Min	10

Since r_2 < 10, then, include this node (1-2-*-*) in the tree.

b) Partial Sequence: (1-3-*-*)

r_3 = 3 t=4

Unscheduled jobs are (2-4). The minimum completion time for all these jobs can be computed as follows:

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
2	1	4	2	6
4	5	5	5	10
			Min	6

Since r_3< 6, then, include this node (1-3-*-*) in the tree.

c) Partial Sequence: (1-4-*-*)

$r_4 = 5$ t=4

Unscheduled jobs are {2-3}. The minimum completion time for all these jobs can be computed as follows:

k	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
2	1	4	2	6
3	3	4	5	9
			Min	6

Since $r_4 < 6$, then, include the node (1-4-*-*) in the tree.

Iteration 2: Step (2)

Find L_{max} for the three sequences as follows:

In sequence (1-2-*-*), the first and the second positions are already assigned to jobs 1 and 2 respectively. For the remaining two positions, based on the EDD sequence (1-4-3-2), job 4 will be assigned to position three and job 3 will be assigned to position four. The computation for the lower bound which is L_{max} is as follows:

job (j)	p_j	r_j	S_j	C_j	d_j	L_j
1	4	0	0	4	8	-4
2	2	1	4	6	12	-6
4	5	5	6	11	10	1
3	6	3	11	17	11	6

Hence, from the table above the L_{max} is equal to 6.

In sequence (1-3-*-*), the first and the second positions are already assigned to jobs 1 and 3 respectively. For the remaining two positions, based on the EDD sequence, job 4 will be assigned to position three and job 2 will be assigned to position four. The computation for the lower bound is as follows:

job (j)	p_j	r_j	S_j	C_j	d_j	L_j
1	4	0	0	4	8	-4
3	6	3	4	10	11	-1
4	5	5	10	15	10	5
2	2	1	15	17	12	5

Hence, from the table above the L_{max} is equal to 5.

In sequence (1-4-*-*), the first and the second positions are already assigned to jobs 1 and 4 respectively. For the remaining two positions, based on the EDD sequence, job

3 will be assigned to position three and job 2 will be assigned to position four. The computation for the lower bound is as follows:

job (j)	p_j	r_j	S_j	C_j	d_j	L_j
1	4	0	0	4	8	-4
4	5	5	5	10	10	0
3	6	3	10	16	11	5
2	2	1	16	18	12	6

Hence, from the table above the L_{max} is equal to 6. When the lower bound values for the three nodes are compared, it should be clear that node (1-3-*-*) has the minimum value. Thus, from this node branching should continue as shown in the Figure below.

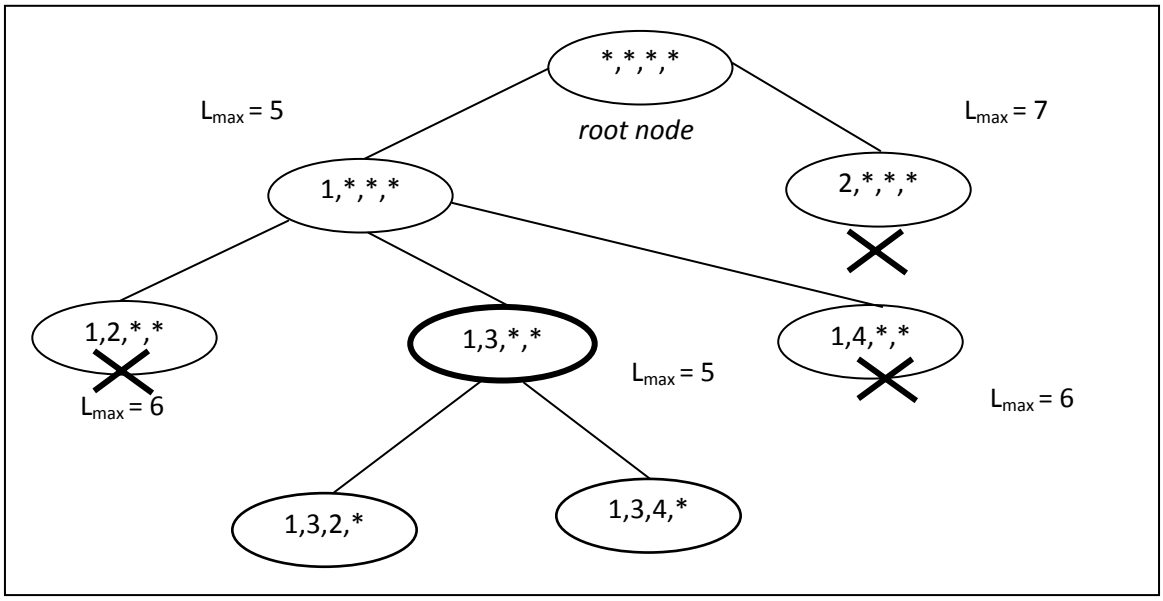


Figure 2.16 Level 3 Nodes for 4-job Problem.

Next, verify the following condition in order to determine which node(s) to consider:

$$r_j < \min_{k \in J} \{\max(t, r_k) + p_k\}$$

Iteration 3 – Step (1)

a) Partial Sequence: (1-3-2-*)

r_2 = 1 **t=10**

Unscheduled job is {4}. The minimum completion time for this job can be computed as follows:

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
4	5	10	5	15
			Min	15

Since $r_2 < 15$, then, include this node (1-3-2-*) in the tree.

b) Partial Sequence : (1,3,4,*)

r_2 = 5 t=10

Unscheduled job is {2}. The minimum completion time for this job can be computed as follows:

K	r_k	S_k=max(t,r_k)	p_k	$S_k + p_k$
2	1	10	2	12
			Min	12

Since $r_4 < 12$, then, include this node (1-3-4-*) in the tree.

Iteration 3: Step (2)

Find L_{max} for the two sequences as follows:

In sequence (1-3-2-*), the first position, the second position, the third position are already assigned to jobs 1, 3, and 2 respectively. For the remaining job which is job 4, it will be assigned to the last position. The computation for the lower bound will be as follows:

job (j)	p_j	r_j	S_j	C_j	d_j	L_j
1	4	0	0	4	8	-4
3	6	3	4	10	11	-1
2	2	1	10	12	12	0
4	5	5	12	17	10	7

Hence, from the table above the L_{max} is equal to 7.

In sequence (1-3-4-*), the first position, the second position, the third position are already assigned to jobs 1, 3, and 4 respectively. For the remaining job which is job 2, it will be assigned to the last position. The computation for the lower bound will be as follows:

job (j)	p_j	r_j	S_j	C_j	d_j	L_j
1	4	0	0	4	8	-4
3	6	3	4	10	11	-1
4	5	5	10	15	10	5
2	2	1	15	17	12	5

Hence, from the table above the L_{max} is equal to 5. When the lower bound values for the two nodes are compared, it should be clear that node (1-3-4-2) has the minimum value. This concluded the procedure of the branch and bound and also means the optimal sequence has been obtained. The optimal sequence is **{1-3-4-2}.**

2.11 MINIMIZATION OF TOTAL WEIGHTED TARDINESS PROBLEM *(1 || $\Sigma\omega_j T_j$)*

In classical scheduling theory, minimization of total weighted tardiness has been researched thoroughly. A variety of approaches have been developed. In the following pages, one of these approaches which is the branch and bound (B&B) methodology will be explored in solving this problem. In order to reduce the solution space for $\Sigma\omega_j T_j$ which means minimizing the search effort for near optimal solution in the solution space, consider the following lemma which helps in build relative relationship among jobs which produces precedence constraints among some of the jobs.

Lemma:

When minimizing $1 \parallel \Sigma\omega_j T_j$ problem, and for any two jobs say j and k, the following is true:

$P_j \leq P_k$

$d_j \leq d_k$ and

$W_j \geq W_k$,

Then there exists an optimal sequence that minimizes $1 \parallel \Sigma\omega_j T_j$ problem in which job j appears before job k.

As mentioned earlier, this lemma will help in build relative relationship among jobs which produces precedence constraints among some of the jobs which consequently helps in eliminating a significant number of possible sequences which reduces the solution space and search effort.

Method:

Step 1: For the given problem data, identify job's sequence position.

Step 2: Construct a branch and bound tree with only nodes which contains possible jobs to be in the last position in the sequence. Use the Lemma mentioned above to construct those nodes at level one of the tree.

Step 3: Find the lower bound for each node developed. This means compute the $\Sigma\omega_j T_j$ as the lower bound for each node.

Step 4: Branch from the node(s) which has (have) the minimum value(s) of Lower bound.

Step 5: When branching to a lower level node, include only those jobs which satisfy the Lemma mentioned above.

Step 6: Continue branching to lowest level of the tree till all the jobs are included in the schedule.

Example 2.10

Find an optimal sequence for $1 \mid\mid \Sigma\omega_j T_j$ problem given the data in the following table:

Job(j)	1	2	3	4
w_j	4	5	3	5
p_j	12	8	15	9
d_j	16	26	25	27

Solution:

The first step is to determine the set of jobs that satisfy the relationship condition in the lemma mentioned earlier in which the conditions for any jobs are as follows:

$$p_j \le p_k \quad \text{and} \quad d_j \le d_k \quad \text{and,} \quad \omega_j \ge \omega_k$$

By the applying this relationship, it can be found that in an optimal sequence job 1 will appear before job 3. Similarly, job 2 will appear before job 4. Consequently, for the B&B tree, only two nodes can be constructed which represent the partial sequences as shown in the following figure.

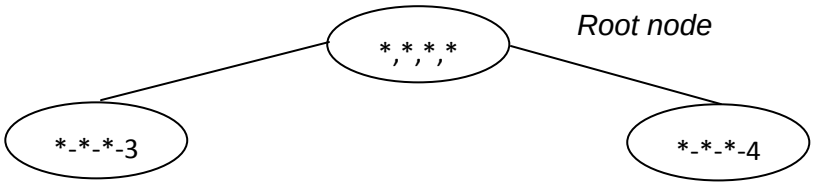


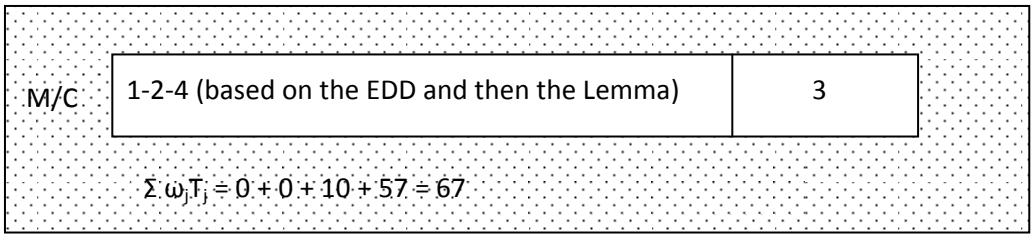
Figure 3 Level 1 Nodes for 4-job Problem.

For the node with partial sequence (*-*-*-4), the lower bound (LB) which is $\Sigma\omega_j T_j$ can be computed as follows:

M/C	1-3-2 (based on EDD and then the Lemma)	4

$\Sigma\ \omega_j T_j = 0 + 6 + 45 + 85 = 136$

Similarly, the lower bound (LB) for node with partial sequence (*-*-*-3) can be computed as follows:



Thus, since the node with partial sequence (*-*-*-3) has a smaller lower bound value, then, branching should be continued with this node. This means, the node with partial sequence (*-*-*-4) should be fathomed. Only two nodes can be investigated with the two partial sequences (*-*-4-3) and (*-*-1-3) as shown in the figure below.

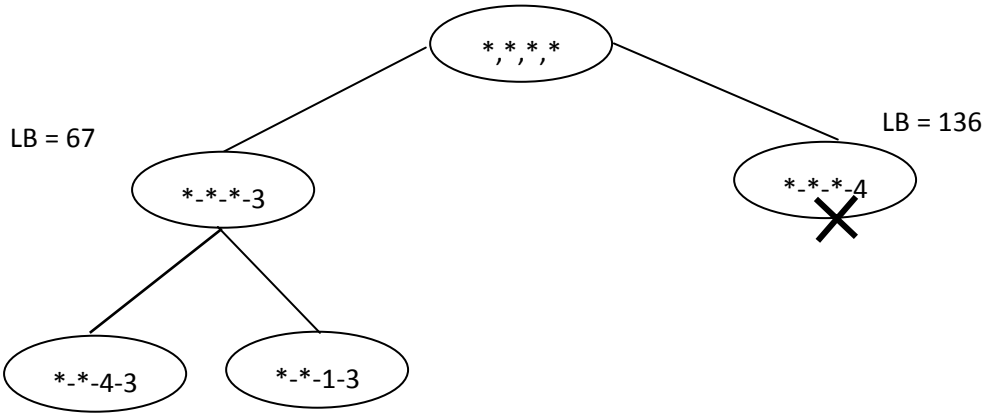
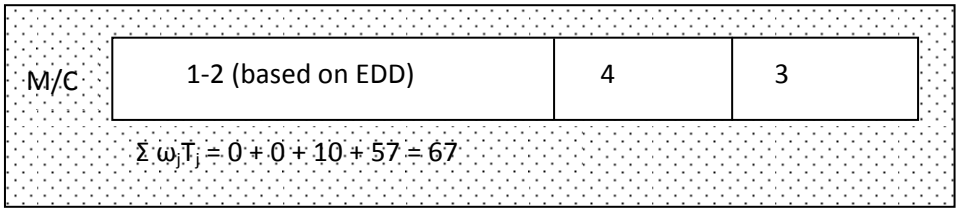
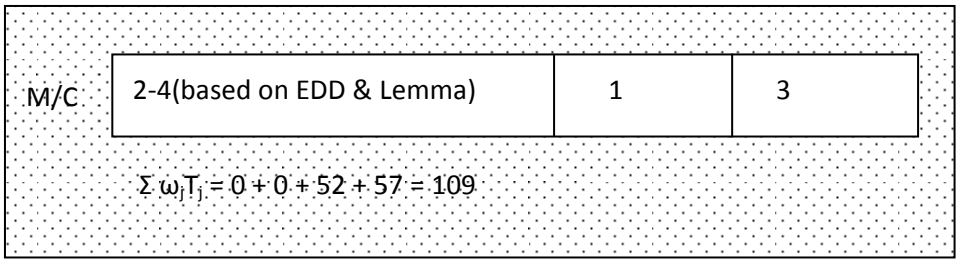


Figure 2.18 Level 2 Nodes for 4-job Problem.

The lower bound for the partial sequence (*-*-4-3) which is $\Sigma\omega_j T_j$ can be computed as follows:



The lower bound for the partial sequence (*-*-1-3) which is $\Sigma\omega_j T_j$ can be computed as follows:



Therefore, since the node with partial sequence (*-*-4-3) has lower bound value, then, branching should be continued from this node. This means the node with partial sequence (*-*-1-3) should be fathomed. Only two nodes can be investigated with the two partial sequences (*-1-4-3) and (*-2-1-3) as shown in the figure below.

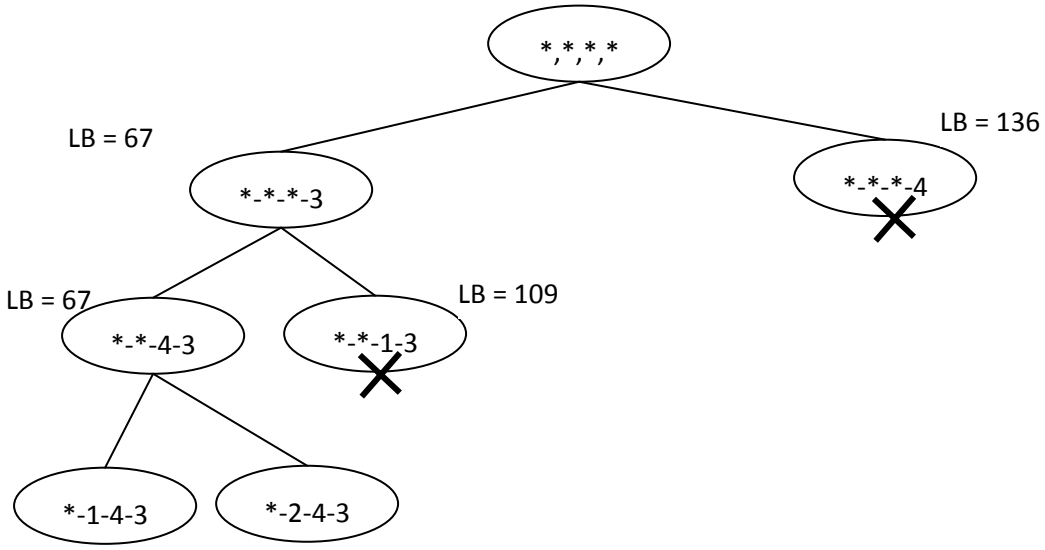
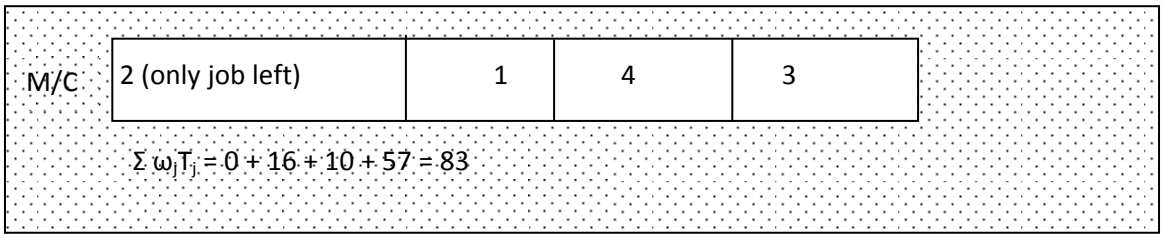
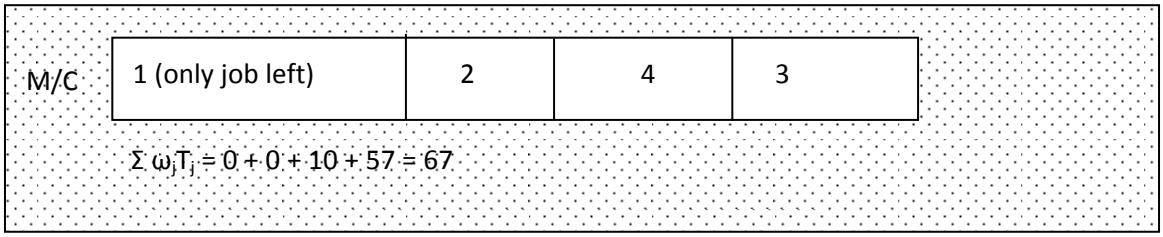


Figure 3.19 Level 3 Node for 4-job problem.

The lower bound for the partial sequence (*-1-4-3) which is $\Sigma\omega_j T_j$ can be computed as follows:



The lower bound for the partial sequence (*-2-4-3)) which is $\Sigma\omega_j T_j$ can be computed as follows



Thus, since the node with partial sequence (*-2-4-3) has lower bound value, then, an optimal sequence has been found which is (1-2-4-3) because no more branching can be done all jobs have been scheduled. The complete computation for the schedule is given in the following table:

Job(j)	Pj	wj	Cj	dj	Tj	ωjTj
1	12	4	12	16	0	0
2	8	5	20	26	0	0
4	9	5	29	27	2	10
3	15	3	44	25	19	57

Total weighted Tardiness, $\Sigma\omega_j T_j = 67$. The same approach can be used to solve the following scheduling problem: $1 \mid\mid \Sigma T_j$. This means, the problem is solved with given that all jobs have equal weight (ω_j=1).

2.12 MINIMIZATION OF MAXIMUM LATENESS WITH PRECEDENCE PROBLEM *(1 | PREC | L_{max})*

In this section, the scheduling problem has jobs that have precedence relationship among them. While scheduling jobs on the single machine, the objective is to minimize the maximum lateness (L_{max}). One of the well known solution methodology to solve this problem is based on the least remaining slack (RS) rule. This rule is applied to solve 1 | prec | L_{max} problem.

The RS_j can be computed as follows:

$$RS_j = d_j - p_j - t.$$

where,

RS_j = the remaining slack of job j.
p_j = processing time of job j.
d_j = due date for job j.
t = time of the schedule,

The algorithm to implement the RS rules is as follows:

Step 1: At time zero, set t = 0.
Step 2: From the given precedence graph, form the set of schedulable jobs.
Step 3: Calculate the RS values of these jobs.
Step 4: Select a job j having minimum value of RS_j and schedule it on the machine.
Step 5: Remove the recently scheduled job from the set schedulable jobs.
Step 6: If all jobs have been scheduled, STOP. Otherwise update the set of schedulable job from precedence graph. Update value of t. Go to step 3.

Example 2.11
Consider 1| *prec* | L_{max} problem with the data given in the following table:

Job (j)	1	2	3	4	5	6	7	8
p_j	2	3	2	1	4	3	2	2
D_j	5	4	13	6	12	10	15	19

Also, the precedence network for the jobs is given in the following figure:

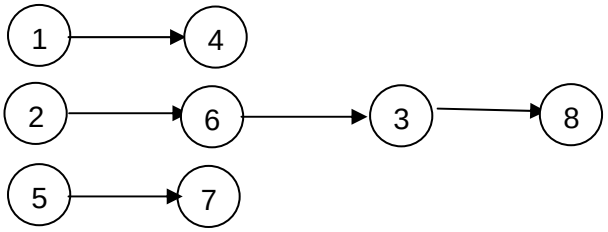


Figure 2.20 Precedence Graph of 8-Job problem

Solution:

At time zero, t = 0. Set of schedulable jobs based on precedence graph are contains the following jobs: {1-2-5}. The computation for the RS values for these jobs is given in the table:

Job(j)	d_j	P_j	t	RS_j
1	5	2	0	3
2	4	3	0	1
5	12	4	0	8
			MIN(RS_j)	1

Job 2 has the minimum RS value. Then, Job 2 is scheduled first on the machine at time 0 and it will be completed at time 3. Thus, updated the value of t to be = 3. The set S = {2}.

Next, job 6 is added to the Set of schedulable jobs. This means the set contains the following jobs: {1-5-6}. The computation for the RS values for these jobs in the set is given in the following table:

Job(j)	d_j	p_j	t	RS_j
1	5	2	3	0
5	12	4	3	5
6	10	3	3	4
			MIN(RS_j)	0

Since job 1 has the minimum RS value, then, it is scheduled second on time 3 the machine. The job will be completed ate time 5. Therefore, the updated value for t is = 5. The set S = {2-1}.

Next, job 4 is added to the schedulable jobs list. The set of schedulable jobs is {5-6-4}. Out of the three jobs in the set, job 4 has the minimum RS value as shown in the following table:

Job(j)	dj	Pj	t	RSj
4	6	1	5	0
5	12	4	5	3
6	10	3	5	2
			MIN(RSj)	0

Thus, job 4 is scheduled at time 5 and it will be completed at time 6. The set S = {2-1-4}. This will update the t value to be = 6. At this time, only two jobs are in the schedulable set. These jobs are {5-6}. Out of these two jobs, job 6 has the minimum RS value as shown in the following table:

Job(j)	d_j	p_j	t	RS_j
5	12	4	6	2
6	10	3	6	1
			MIN(RS_j)	1

Then, job 6 is scheduled at time 6 and it will be completed at time 9. The set S = {2-1-4-6}. This will updated the t value to be = 9. At this time, job 3 is added to the schedulable set. The jobs in the schedulable set are {5-3}. Out of these two jobs, job 5 has the minimum value of RS as shown in the following table:

Job(j)	d_j	p_j	t	RS_j
3	13	2	9	2
5	12	4	9	-1
			MIN(RS_j)	-1

Therefore, job 5 is scheduled at time 9 and completed at time 13. The set S = {2-1-4-6-5}. This will updated the value of t to be = 13. At this time, job 7 is added to the schedulable set of jobs. The set of schedulable jobs contains the following jobs: {3-7}. Out of these two jobs, job 3 has the minimum value of RS as shown in the following table.

Job(j)	d_j	p_j	t	RS_j
3	13	2	13	-2
7	15	2	13	0
			MIN(RS_j)	-2

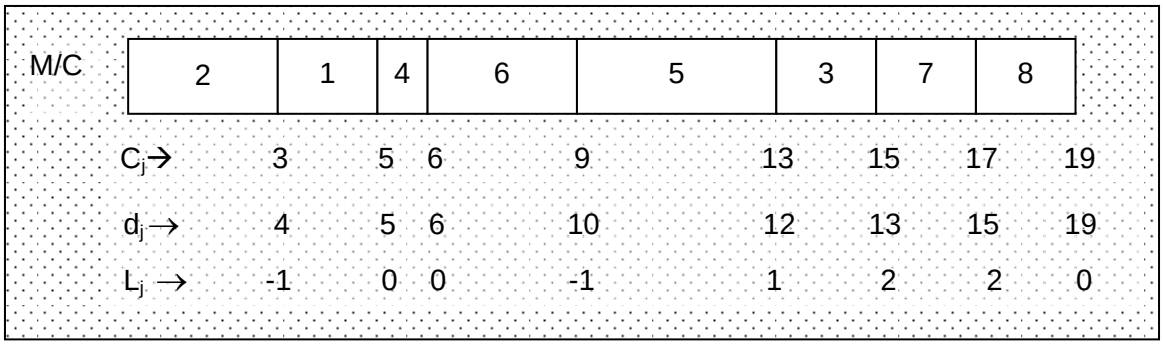
Next, job 3 is scheduled at time 13 and it will be completed at time 15. The set S = {2-1-4-6-5-3}. The value of t is = 15. At this time, job 8 is added to the schedulable set. The set of schedulable jobs is {7-8}. Out of these two jobs, job 7 has the minimum value of RS as shown in the following table:

Job(j)	d_j	p_j	t	RS_j
7	15	2	15	-2
8	19	2	15	2
			MIN(RS_j)	-2

Thus, Job 7 is scheduled at time 15 and it will be completed at time 17. The set S = {2-1-4-6-5-3-7}. The value of t is = 17. The only unscheduled job is job 8 with RS value of 0 as shown in the following table:

Job(j)	d_j	P_j	t	RS_j
8	19	2	17	0
			MIN(RS_j)	0

Then, job 8 is scheduled at time 17 and it will be completed at time 19. The schedulable job set is empty. Thus, STOP. The final sequence of jobs on the machine is as follows: {2-1-4-6-5-3-7-8}. The Gantt chart for this sequence is shown below:



From the Gantt chart, the maximum lateness (L_{max}) is 2.

EXERCISES

2.1 Consider the Shortest process time [SPT] sequence as

$$t_{[1]} \leq t_{[2]} \leq \ldots\ldots\ldots \leq t_{[n]}$$

In particular, prove

a) SPT minimizes mean flow time.
b) SPT minimizes mean waiting time
c) SPT minimizes mean completion time.
d) SPT minimizes $C_{max.}$

2.2 Construct an example single-machine problem to show that SPT does not minimize mean tardiness.

2.3 Prove that LPT sequencing maximizes mean waiting time.

2.4 A single machine facility faces problem of sequencing the production work for six customer orders described in table below.

Order	1	2	3	4	5	6
Processing time	18	26	14	8	17	22

a. What sequence will minimize the mean flow time of these orders? What is the mean flow time in this schedule?
b. Suppose that customer orders 1 and 5 are considered twice important as the rest what sequence would you propose.

2.5 Consider the following single machine scheduling problem

Jobs (j)	1	2	3	4	5
P_j	8	3	3	6	3
d_j	4	8	11	10	11

a. What sequence will minimize the average waiting time? Then compute Average completion time, Maximum flow time, and Average waiting time.
b. What sequence will minimize both maximum lateness and the maximum tardiness? Then, compute maximum lateness, maximum tardiness, maximum earliness, total tardiness, and total earliness.

2.6 Consider the following single machine scheduling problem

Job	1	2	3	4	5	6
P_j	8	12	7	16	9	4
ω_j	4	10	4	3	8	10

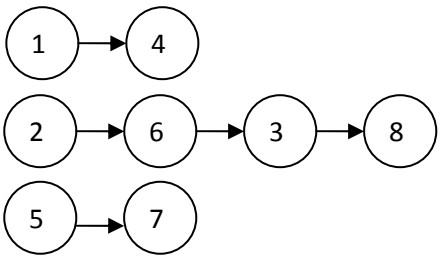
Generate a sequence to minimize weighted completion time. What is the flow time of each job in the shop?

2.7 Show that for any job i:

$\mathbf{L_i = F_i - a_i = C_i - r_i - a_i = C_i - d_i}$, and hence conclude that

$\mathbf{\overline{L} = \overline{F} - \overline{a} = \overline{C} - \overline{r} - \overline{a} = \overline{C} - \overline{d}}$

2.8 Consider the following problem as an instance of the $1 \mid prec \mid \sum w_j C_j$. An 8-job single machine data with job precedence constraints graph is given below.



The weights and process times of the jobs are given in the following table.

Job	1	2	3	4	5	6	7	8
P_j	3	6	6	5	4	8	10	4
w_j	6	18	12	8	8	17	18	15

Solve the problem to minimize total weighted completion times using chain-method.

2.9 Consider the following single machine scheduling problem

Job	1	2	3	4	5	6	7
p_j	8	3	12	1	7	5	3
D_j	12	7	23	4	21	17	8

a) Use SPT sequence and, find average waiting time ($\overline{W}$).

b) Use EDD sequence and, find maximum lateness (L_{max}), average Lateness ($\overline{L}$).

c) Generate a sequence to minimize number of tardy jobs (n_t).

2.10 Solve the following $1 \| n_t$ problem in which the following data is given:

Job	1	2	3	4	5	6	7
P_j	9	4	3	7	10	6	8
d_j	15	7	5	12	20	23	30

2.11 Consider $1 \| n_t$ problem with the following data:

Job	1	2	3	4	5	6	7	8	9	10
P_j	15	11	10	5	25	4	8	3	20	11
d_j	71	76	73	88	47	59	24	55	23	47

Find the sequence that minimizes the number of jobs tardy and compute n_t

2.12 Bin-laden contracting company has orders for five houses to be built. Bin-laden is well known company and has good reputation for excellence, thus, the customers will wait as long as necessary for their house to be built. The revenue in Saudi Riyals to Bin-laden for each house respectively is as follows: 145000, 290000, 910000, 1150000, and 200000. Also, the times needed in days to build each house respectively are as follows: 150, 200, 400, 450, and 1000. Assuming that Bin-laden, can only work on one house at time, what would be an appropriate measures to schedule building the houses? Using this measure, what schedule should Bin-laden?

2.13 At Toyota (Fast service) repair shop there are six cars in for repair. The car's owners will wait in the waiting and will leave when their cars are finished. At night shift, there is only one mechanic available to do the repairs whose name is Mohammed. Mohammed estimates the times needed for repair for each cars respectively as follows: 115, 145, 40, 25, 70, and 30 minutes. What schedule would you recommend for Mohammed? How would you help Mohammed to justify having another mechanic with him? (**Show all of your work and show all assumptions you make**)

2.14 Consider an instance of the $1 \mid r_j \mid L_{max}$ problem with data as follows.

Job	1	2	3	4
p_j	3	4	6	10
r_j	6	0	7	6
d_j	3	10	17	18

Find optimal sequence and show complete schedule.

2.15 Consider the following data as an instance of the $1 \mid\mid \Sigma w_j T_j$ problem;

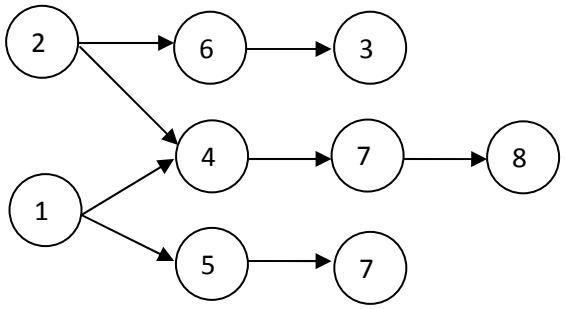
Job	1	2	3	4	5
p_j	6	3	2	4	5
w_j	3	2	1	5	3
d_j	11	4	12	7	9

Find optimal sequence and compute $\Sigma w_j T_j$.

2.16 Consider the following data as an instance of the $1| prec | L_{max}$ problem.

Job	1	2	3	4	5	6	7	8
p_j	2	3	2	2	4	3	2	2
d_j	5	4	13	6	12	10	15	19

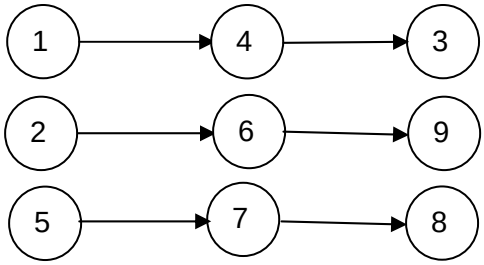
The precedence network for the problem is shown below:



Find the best efficient sequence this problem and compute L_{max} ?

2.17 Consider $1| prec | \Sigma \omega_j C_j$ problem in which the data and precedence network are given below:

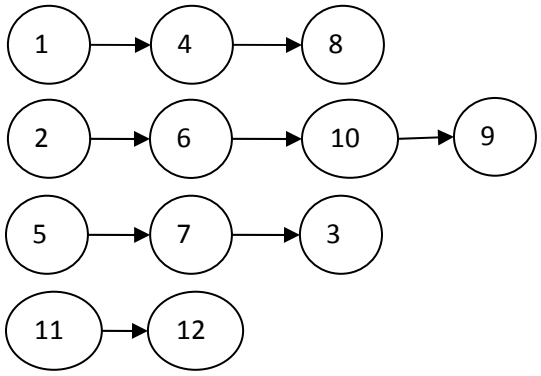
Job	1	2	3	4	5	6	7	8	9
Process time	2	3	2	1	4	3	2	2	4
Weight	5	4	6	6	12	11	12	10	13



Solve the problem to minimize total weighted completion times using chain-method. Then, constructing the Gantt chart for the solution obtained. Next, compute Average flow time, Average completion time, Maximum flow time, and Average waiting time.

2.18 Consider 1 | prec | L_{max} problem with the problem data and precedence network given below:

Job	1	2	3	4	5	6	7	8	9	10	11	12
Process time	2	3	2	1	4	3	2	2	5	4	6	4
Due dates	5	4	13	6	12	10	15	19	20	11	14	18



Solve the problem to minimize maximum lateness using remaining slack rule. Then, constructing the Gantt chart for the solution obtained. Next, compute maximum lateness, maximum tardiness, maximum earliness, total lateness, total tardiness, and total earliness.

2.19 Consider 1 | | n_t problem with the following data:

Job	1	2	3	4	5
P_i	7	8	4	6	6
d_i	9	17	18	19	21

Find optimal sequences and compute n_t, T_{max}, and $\overline{\mathbf{T}}$.

2.20 A company has a cell that can produce three parts: A, B and C. The time required to produce each part is 25, 80, and 10 minutes for each part respectively. The value to produce the parts is 5 riyals, 20 riyals, and 1 riyal respectively. How would you schedule the parts through the cell to minimize the value of work in process?

2.21 Consider the following problem: 1/ r_j / ΣF_j and find the best schedule using the appropriate dispatching rule (**not optimal**) which will give the minimum total flow time with release time using the following data:

Job	1	2	3	4	5	6	7	8	9	10
P_i	16	11	6	18	2	20	19	20	8	16
r_i	22	6	0	6	21	7	29	121	64	48

2.22 Use the weighted shortest processing time to find the optimal solution for the data under consideration. Then, compute total weighted completion time. Also, compute flow time for each job in the shop

Job	1	2	3	4	5	6
p_j	8	4	9	12	11	4
w_j	3	2	1	6	5	7

2.23 Consider 1 | | L_{max} problem with the following the processing times and due dates

Job	1	2	3	4	5	6	7
P_j	6	18	12	10	10	17	16
d_j	8	42	44	24	90	85	68

Find the optimal sequence and compute L_{max} and $\overline{L}$.

2.24 Consider 1|r_j | L_{max} problem with the following data:

Job	1	2	3	4	5
P_j	6	18	12	10	10
r_j	0	18	12	8	8
d_j	8	42	44	24	90

Find the optimal sequence using the branch and bound with the preemptive due date as the lower bound and compute L_{max}

2.25 Consider an instance of the 1 || ΣT_j problem with data as follows.

Job	1	2	3	4	5

p_j	3	9	6	5	12
d_j	13	10	7	3	5

Find optimal sequence and show complete schedule

2.26 Consider the following data as an instance of the $1 \mid\mid \Sigma w_j T_j$ problem;

Job	1	2	3	4
p_j	6	3	2	4
w_j	3	2	1	5
D_j	11	9	12	7

Check for the following relationship from the given data to make initial sequence

$$d_j \le d_k \ldots, p_j \le p_{k,\text{and},} \omega_j \ge \omega_k .$$

Use Branch and Bound method to find optimal solution.

2.27 Consider an instance of the $1 \mid rj \mid L_{max}$ problem with data as follows.

Job	1	2	3	4	5
p_j	3	4	6	10	2
r_j	0	1	7	6	9
d_j	3	10	17	18	15

Find optimal sequence and show complete schedule.

2.28 Consider the following data

Job	1	2	3	4	5	6	7
p_j	8	12	7	6	9	4	12
ω_j	4	1	4	3	8	1	5

Apply WSPT, and minimize $\Sigma\omega j$ Cj. Find Flow time of each job in the shop.

2.29 Consider $1 \mid\mid L_{max}$ problem with the following data:

Job	1	2	3	4	5	6	7	8
d_j	5	4	13	6	12	10	15	19
P_j	2	3	2	1	4	3	2	2

Given also the following precedence constraints

$2 \Rightarrow 6 \Rightarrow 3$

$1 \Rightarrow 4 \Rightarrow 7 \Rightarrow 8$

Find the optimal sequence and compute $\overline{L}, \overline{T}, \overline{F}$, T_{max}, and L_{max}

2.30 Consider 1 || n_t problem with the following data

Job	1	2	3	4	5	6	7	8	9	10
d_j	19	16	25	3	8	14	31	23	2	15
P_j	5	3	1	2	4	4	2	1	1	4

Find the optimal sequence and compute $\overline{L}, \overline{T}, \overline{F}$, T_{max}, and L_{max}

2.31 A manufacturer of charm bracelets has five jobs to schedule for a leading customer. Each job requires a stamping operation followed by a finishing operation. The finishing operation can begin immediately after its stamping is complete for any item. The table below shows operation times per item in minutes for each job. At the stamping operation, each job requires set-up before processing begins, as described in the table. Find a schedule that completes all five jobs as soon as possible.

		Operation Time per Item		
Job no.	Number in lot	Stamp	Finish	Set-up
1	20	2	8	100
2	25	2	5	250
3	100	1	2	60
4	50	4	2.5	60
5	40	3	6	80

3

Parallel Machine Scheduling

CHAPTER CONTENTS

3.1 INTRODUCTION

When similar type of machines are available in multiple numbers and jobs can be scheduled over these machines simultaneously, parallel machines scheduling environment is at hand as shown in Figure 3.1 below.

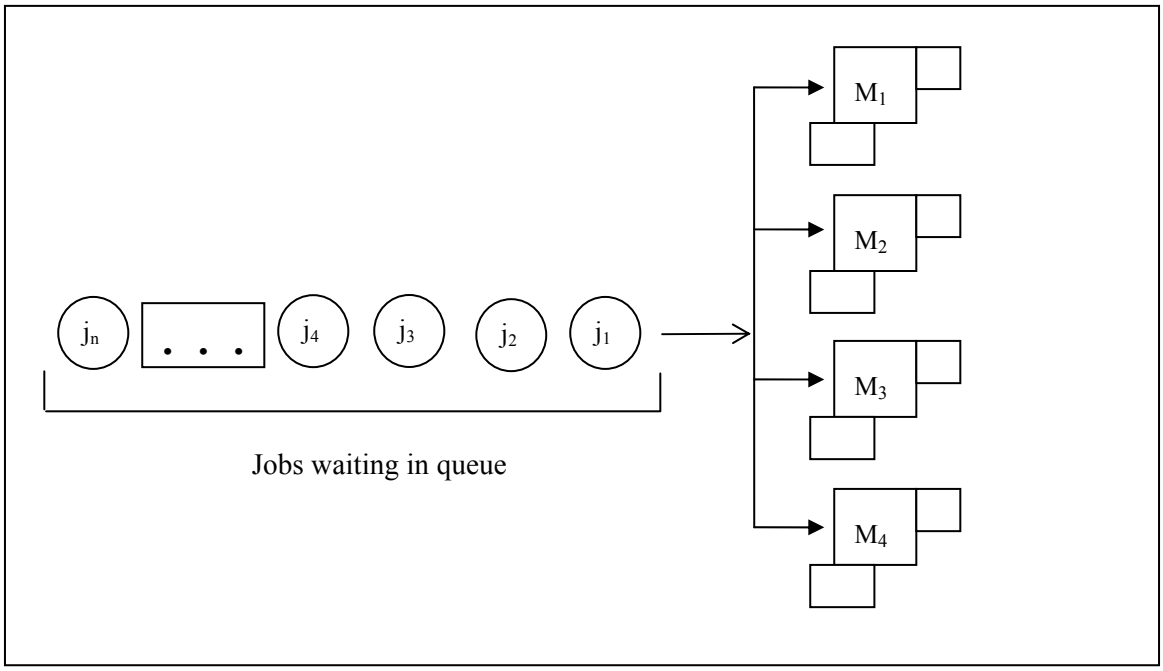


Figure 3.1 Four Parallel Machines with n-jobs.

The existence of parallel machines environment is common in real world flow shop and job shops systems. Knowledge of parallel machines modeling is useful to design the large-scale flexible flow shop and job shop systems

3.2 MINIMIZATION OF MAKESPAN PROBLEM *(P_m || C_{max})*

This problem deals with scheduling m parallel machines when the objective function is to minimize the makespan. A variety of heuristics are employed to determine a near-optimal schedule. Some of these heuristics include longest processing time first (LPT) rule and load- balancing heuristic.

3.1.1 Longest Processing Time first (LPT) Rule

A common heuristic used in parallel machines scheduling is the LPT rule. According to this heuristic, jobs are arranged in decreasing order of process times. The jobs having large values of process times are given high priority for scheduling on the parallel machines. The following relationship can be used to find how far the solution obtained by the LPT rule is far from an optimal solution.

$$\frac{C_{max}(LPT))}{C_{max}(OPT))} \leq \frac{4}{3} - \frac{1}{3m}$$

Example 3.1
Using LPT rule, find the best schedule for jobs on the machines for the following $P_4 \mid\mid C_{max}$ problem.

Job (j)	1	2	3	4	5	6	7	8	9
p_j	7	7	6	6	5	5	4	4	4

Solution:

The LPT sequence is as follows: 1-2-3-4-5-6-7-8-9. From the LPT sequence, select job 1 to be scheduled on machine 1, then, select job 2 to be scheduled on machine 2, next, select job 3 to be scheduled on machine 3, and finally job 4 to be scheduled on machine 4. The partial schedule generated for each machine is presented in the following table

Table 3.1 Partial schedule for the four machines.

Machine (M)	Job assigned	Start Time S_j	Process Time p_{ij}	Completion Time (C_j)
M_1	1	0	7	7
M_2	2	0	7	7
M_3	3	0	6	6
M_4	4	0	6	6

From the LPT sequence, the unscheduled jobs are {5-6-7-8-9}. Since machines 3 and machine 4 are free and, available for next jobs at times 6, schedule job 5 and job 6 at these machines at time 6. The schedule of these jobs is presented in Table 2 as follows:

Table 3.2 Partial schedule for job set $\{j_5, j_6\}$.

Job (j)	Machine (M)	Start Time S_j	Process Time p_{ij}	Completion Time C_j
j_5	M_3	6	5	11
j_6	M_4	6	5	11

The next unscheduled jobs in the ordered set are job 7 and job 8. Machine 1 and machine 2 are free and available for next jobs at time 7. Schedule job 7 and job 8 at machine 1 and machine 2 at time 7 respectively. The schedule of these jobs is presented in Table 3 as follows:

Table 3.3 Partial schedule for job set $\{j_7, j_8\}$.

Job (j)	Machine (M)	Start Time S_j	Process Time p_{ij}	Completion Time C_j
j_7	M_1	7	4	11
j_8	M_2	7	4	11

The next unscheduled job in the ordered set is job 9. Machine 1, machine 2, machine 3, and machine 4 are all free for next job at time 11. Schedule job 9 on any machine; say at machine 1 at time 11. The complete schedule of all jobs on 4 parallel machines is given in Table 4 as follows:

Table 3.4 Complete schedule for job set $\{j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8, j_9\}$.

Job (j)	Machine (M)	Start Time S_j	Process Time p_{ij}	Completion Time C_j
j_1	M_1	0	7	7
j_2	M_2	0	7	7
j_3	M_3	0	6	6
j_4	M_4	0	6	6
j_5	M_3	6	5	11
j_6	M_4	6	5	11
j_7	M_1	7	4	11
j_8	M_2	7	4	11
j_9	M_1	11	4	15

Gantt chart for $P_4 \parallel C_{max}$ schedule (shown in Table 3.4) is presented in Figure 3.2 as shown below;

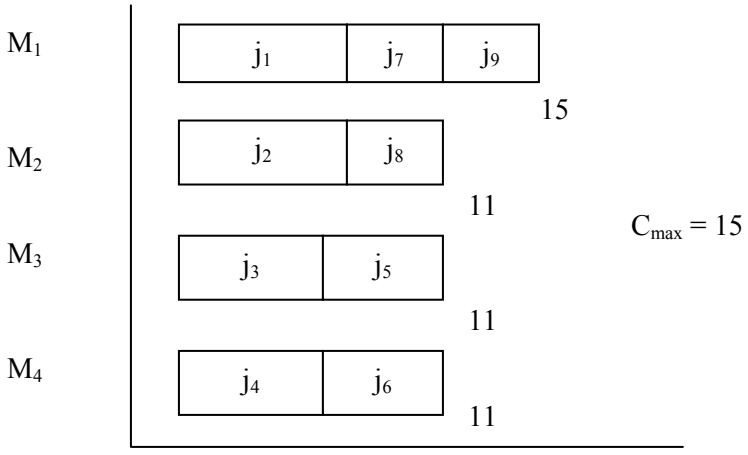


Figure 3.2 Gantt chart For P4|| C_{max} Problem. Note $C_{max} = 15$.

Example 3.2

Is the C_{max} value obtained in Example 3.1 optimal? If not, what is the optimal solution?

Solution:

For finding optimality of schedules, the following ratio between $C_{max}(OPT)$ and $C_{max}(LPT)$ is observed;

$$\frac{C_{max}(LPT))}{C_{max}(OPT))} \leq \frac{4}{3} - \frac{1}{3m}$$

Since, there are 4 machines, so m = 4. The ratio $\frac{C_{max}(LPT))}{C_{max}(OPT))} \leq 1.25$

The ratio of 1.25 indicates that LPT rule will give 25% more value of C_{max} in worst case as compared to C_{max} using optimal methodology. Hence, the optimal value of C_{max} in Example 3.1 should be 12, as calculated as follows:

Since, $C_{max}(LPT) = 1.25 \times C_{max}(OPT)$,

This implies that; $C_{max}(OPT) = C_{max}(LPT)/1.25 = 15/1.25 = 12$

The optimal value of C_{max} in Example 3.1 can be calculated by using *Load Balancing* heuristic as follows;

Let T_w = Total work content of all jobs in the problem. Then, tentative load per machine may be estimated by taking the ratio of T_w and m as follows:

$$\text{Load per machine} = \frac{T_w}{m}$$

For data presented in Example 3.1, $T_w = \sum_{j=1}^{9} p_j = 48$, and m = 4. Hence, tentative load per machine is 12. Table 3.5 presents combination of jobs for which average load per machine is 12. Optimal schedule is presented in Gantt chart (Figure 3.3).

Table 3.5 Optimal Schedule using Load Balancing Heuristic.

Machine	Jobs	Total Time
M_1	j_1, j_5	7 + 5 = 12
M_2	j_2,j_6	7 + 5 = 12
M_3	j_3, j_4	6 + 6 = 12
M_4	j_7, j_8 , j_9	4 + 4 + 4 = 12

Gantt chart for $P_4 \parallel C_{max}$ schedule (shown in Table 3.5) is presented in Figure 3.3 as shown below;

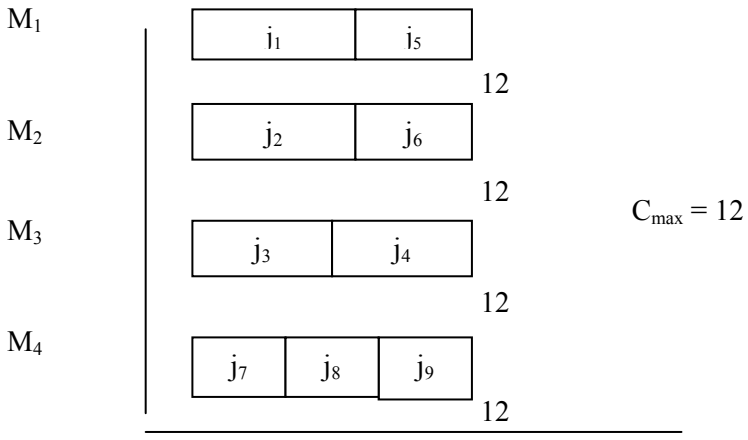


Figure 1.3 Gantt chart for optimal solution.

3.3 MINIMIZATION OF MAKESPAN WITH PRECEDENCE *(P_m | prec | C_{max})*

This scheduling problem occurs when jobs have precedence relationship among them. A precedence relationship diagram shows the precedence relationship. The scheduling calculations are carried out in two steps. In first step, forward pass calculations are made to find out earliest completion times of jobs as under:

Let,

$$C'_j = \text{EarliestCompletionTime of job } j$$

$$C'_j = \max_{i \in \Psi} (C'_i) + p_j$$

Where, C'_i is the earliest completion time of job *i* that belongs to set ψ. Set ψ contains predecessor jobs for job j.

To find critical path on the precedence network, latest completion times of all jobs are calculated as follows:

Let,

$$C''_j = \text{LatestCompletionTime of job } j$$

$$C''_j = \min_{k \in \Omega} (C''_k - p_k)$$

Where,

C''_k = Latest completion time of job *k* that belongs to set Ω

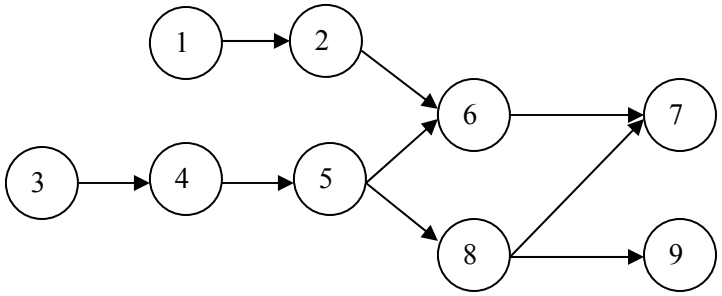
Set Ω contains successor jobs for job j. All the jobs without any successor are assigned a value equal to C_{max} for backward pass calculations.

Example 3.3

A parallel shop has nine jobs with process times as follows:

Job (j)	1	2	3	4	5	6	7	8	9
p_j	4	9	3	3	6	8	8	12	6

The job's precedence constraints are shown in the following precedence diagram.



Assume infinite number of machines. Use directed graph technique and find;
1) Makespan 2) Critical Path

Solution:

The earliest completion times C_j' of all jobs are shown below in Table 6.

Table 3.6 Earliest completion times of all jobs.

Job (j)	1	2	3	4	5	6	7	8	9
C_j'	4	13	3	6	12	21	32	24	30

The earliest completion times are shown below in Figure 4 on the nodes of the precedence network.

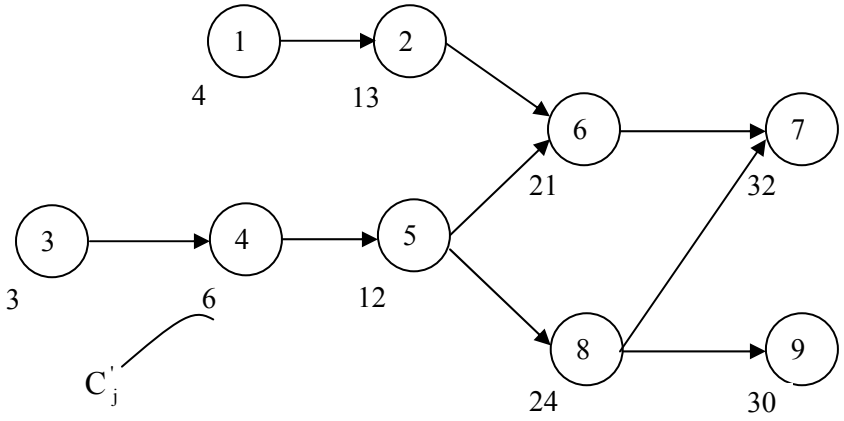


Figure 3.4 Precedence diagram showing earliest completion times.

Job 7 has maximum completion time, which is equal to 32. Hence, the value of C_{max} is equal to 32. To find critical path on the precedence network, latest completion times of all jobs is calculated as follows:

All the jobs without any successor are assigned a value equal to C_{max} for backward pass calculations. In the example problem, jobs 7 and 9 have no successor. Hence, jobs 7 and 9 are assigned latest completion times equal to 32. The latest completion time of job 8 is calculated as follows:

$$C_8'' = \min\{C_7'' - p_7, C_9'' - p_9) = \min\{32-8, 32-6\} = \min\{24, 26\} = 24$$

The latest completion times C_j'' of all jobs are shown below in Table 3.7.

Table 3.7 Latest completion times of all jobs.

Job (j)	1	2	3	4	5	6	7	8	9
C_j''	5	16	3	6	12	24	32	24	32

The earliest and latest completion times of all jobs are shown below in Figure 3.5 on the corresponding nodes of the precedence network. The nodes having equal values of earliest and latest completion times define the critical jobs as well as critical path. For example, jobs j_3, j_4, j_5, j_8 and j_7 have equal values of earliest and latest completion times. The arcs joining these nodes form critical path as shown in Figure 5.

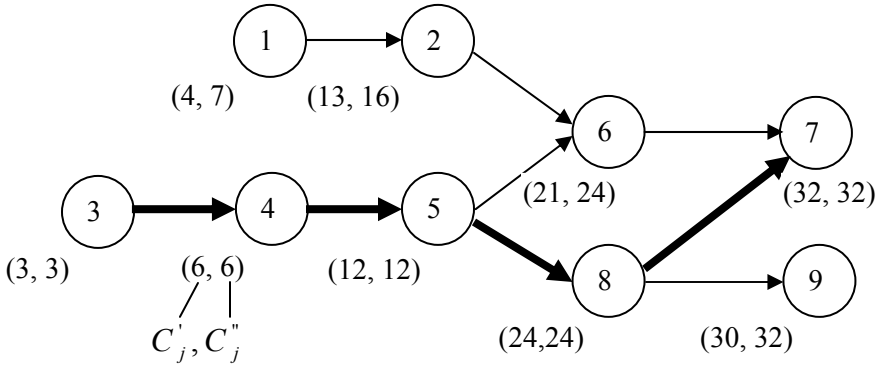


Figure 3.5 Critical Path (j_3-j_4-j_5-j_8-j_7) for the precedence diagram.

3.4 MINIMIZATION OF MAKESPAN WITH P_j =1 AND PRECEDENCE PROBLEM ($P_m \mid P_j = 1, tree \mid C_{max}$)

This parallel machine-scheduling problem pertains to jobs having precedence relationship represented by a *tree.* All the jobs on the tree have process time equal to unity. If the jobs have precedence relationship described either by *intree* or by *outtree* (shown below), then CP rule can be applied to minimize C_{max}.

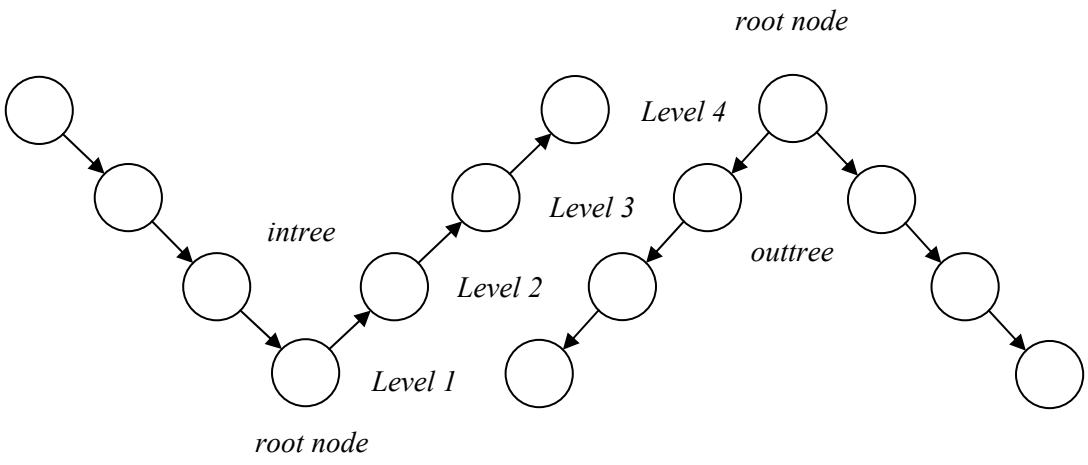


Figure 3.6 Relationship of *intree* and *outtree*

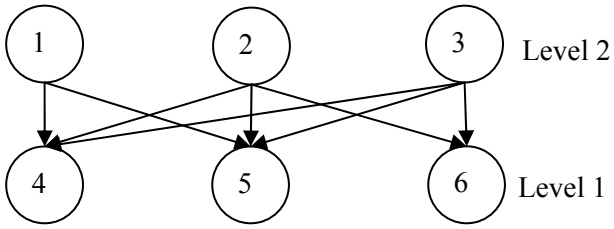
3.4.1 CP Rule

The root node for *intree* precedence graph is at lowest level (Level 1), whereas root node for *outtree* precedence graph is at highest level (Level 5). CP rule imparts high priority to a job having longest string of jobs in the precedence graph. This means that a job having highest level in the precedence graph will be schedules first. Hence CP rule is in fact ***Highest Level First*** rule. This means that root node of *outtree* precedence graph will have highest priority and, root node of *intree* precedence graph will be given least priority. Note that upper bound (UB) on the worst case performance of CP rule is given by the following formula.

$$\frac{C_{max}(CP)}{C_{max}(OPT)} \leq \frac{4}{3}$$

Example 3.4

For the Pm | $p_j = 1$, tree | C_{max}, precedence graph shown below, apply CP rule and find C_{max}. The production shop has two machines in parallel.



Solution:

Using CP rule, first select jobs with Highest Level, hence the jobs in candidate set at time t=0 are from Level 2 jobs; i.e., jobs $\{j_1, j_2, j_3\}$. Schedule job 1 on machine 1 and job 2 on machine 2. At time 1, only job 3 is schedulable. Hence, schedule job 3 on machine 1 at time 1. At time 2, all the jobs at level 1 are schedulable. Jobs at Level 1 include $\{j_4, j_5, j_6\}$. However, only two jobs can be scheduled. So, schedule job 4 and 5 on machine 1 and 2 respectively at time 2. Finally, at time 3 schedule job 6 on machine 2. The complete schedule is shown in Gantt chart (Figure 3.7).

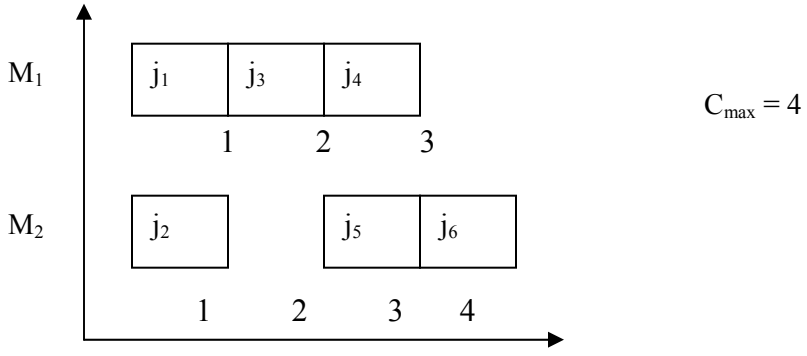


Figure 3.7 Gantt chart for schedule generated by CP Rule.

3.4.2 LNS Rule

The LNS rule is stated as follows:

"Find the number of successors (NS) of all jobs in the precedence graph. While scheduling jobs on parallel machines, give priority to jobs having largest number of successors."

Example 3.5

Solve the problem in Example 3.4 using LNS rule.

Solution

The number of successors for each job in example 3.5 according to precedence graph is shown in the following table:

Job (j)	1	2	3	4	5	6
p_j	1	1	1	1	1	1
NS	2	3	3	0	0	0

At time t=0,

Assign jobs j_2 and j_3 to machines M_1 and M_2.

At time t=1,

Schedule job j_1 on machine M_1, machine M_2 is also free and ready at time t=1. All the three jobs $\{j_3, j_4, j_5\}$ have NS=0. But only job j_6 is schedulable at t=1 because it satisfies the precedence constraint. Hence, schedule job j_6 at time t=1.

At time t=2,

The remaining unscheduled jobs $\{j_4, j_5\}$ are schedulable. So, schedule jobs j_4 and j_5 on machines M_1 and M_2 respectively at time t=2. Note, the makespan (C_{max}) of this schedule is 3 time units.

The complete schedule is presented in Gantt chart (Figure 3.8)

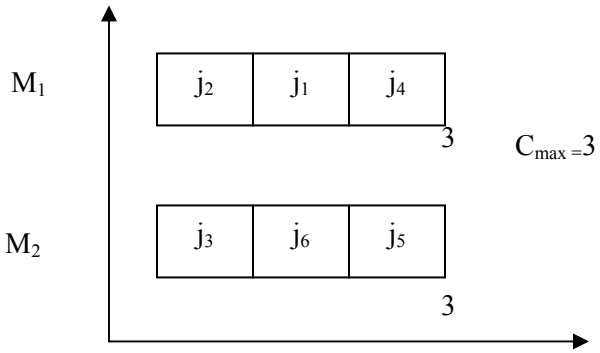
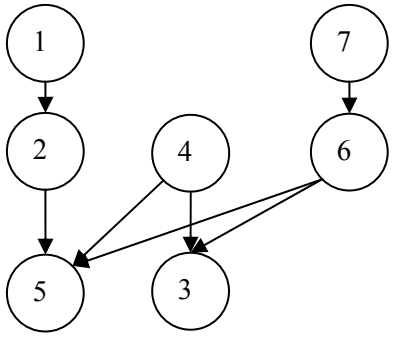


Figure 3.8 Gantt chart for Optimal Schedule generated by LNS rule.

Example 3.6

For the parallel machine problem $P_2 \mid p_j = 1, \text{tree} \mid C_{max}$ the precedence graph is shown below.



Apply the following rules and find $C_{max.}$

a) CP rule b) LNS rule

Is the schedule obtained by CP or LNS rule optimal?

Solution:

a) CP Rule.

To apply CP rule, we determine levels for the jobs from the precedence graph.

Jobs	Level No
j5, j3	1
j2, j4, j6	2
j1, j7	3

At time, t=0

Jobs j_1 and j_7 have highest level. At time t=0, schedule job j_1 on M_1 and job 7 on M_2

At time, t=1

Three jobs are at level 2; namely, j_2, j_4 and j_6. Arbitrarily select jobs j_2 and j_4 to be scheduled at machines M_1 and M_2 at time t=1.

At time, t=2

Only job j_6 is still unscheduled from the jobs at Level 2. So, schedule it at time t=2 on machine M_1. Machine M_2 is free and ready. Jobs at level 1 are $\{j_5, j_3\}$. Select job j_5 and schedule it on M_2.

At time, t=3

Only job j_3 is still unscheduled from the jobs at Level 1. So, schedule it at time t=3 on machine M_1. makespan (C_{max}) of the schedule is 4.

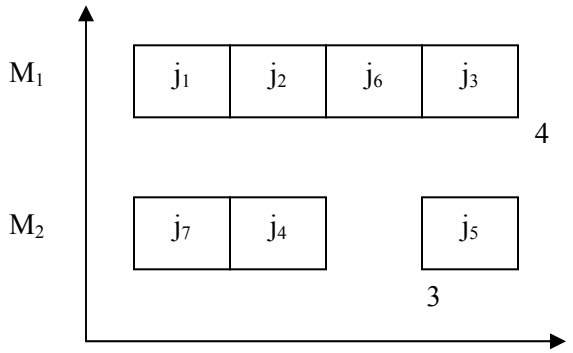


Figure 3.9 Gantt chart of schedule using CP Rule (C_{max} = 4).

b) LNS Rule.

To apply LNS rule, we determine number of successors for each job from the precedence graph.

Job (j)	**1**	**2**	**3**	**4**	**5**	**6**	**7**
p_j	1	1	1	1	1	1	1
NS(j)	1	1	0	2	0	2	1

At time, t=0

Set of scheduled jobs = ϕ

Set of unscheduled jobs = $\{j_1, j_2, j_3, j_4, j_5, j_6, j_7\}$

Jobs j_4 and j_6 have highest values of NS. But job j_6 does not meet precedence condition at time t=0. The predecessor of job j_6 is job j_7, which has NS value of 1. So schedule job j_4 on machine M_1 and schedule job j_7 on M_2.

At time, t=1

Set of scheduled jobs = $\{j_4, j_7\}$

Set of unscheduled jobs = $\{j_1, j_2, j_3, j_5, j_6\}$

Job j_6 has a value of NS=2 and it is yet unscheduled. So schedule j_6 on M_1. Machine M_2 is free and ready now. From the unscheduled set, jobs j_1 and j_2 have high value of NS=1. But only job j_1 can be scheduled at this time. So, select job j_1 and schedule it on M_2.

At time, t=2

Set of scheduled jobs = $\{j_4, j_7, j_6, j_1\}$

Set of unscheduled jobs = $\{j_2, j_3, j_5\}$

From the set of unscheduled jobs, only job j_2 has NS value equal to 1. So, schedule it at time t=2 on machine M_1. M_2 is free and ready. Jobs

j_3 and j_5 both have equal value of NS, which is zero for these jobs. Select arbitrarily job j_5 and schedule it on M_2.

At time, t=3

Set of scheduled jobs = $\{j_4, j_7, j_6, j_1, j_2, j_5\}$

Set of unscheduled jobs = $\{j_3\}$

Only job j_3 is still unscheduled. So, schedule it at time t=3 on machine M_1. makespan (C_{max}) of the schedule is 4.

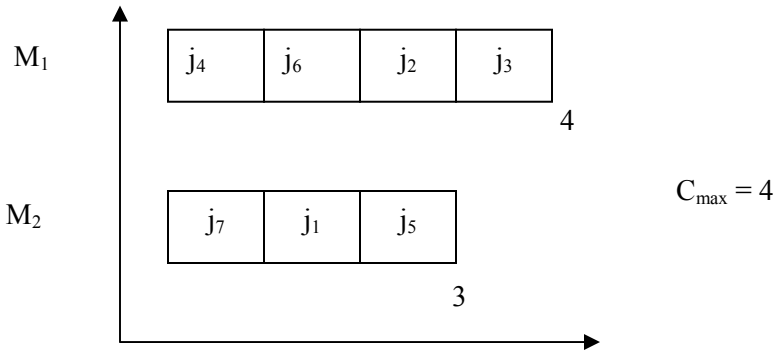
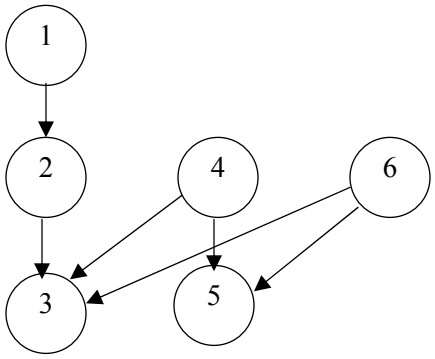


Figure 3.10 Gantt chart of schedule using LNS Rule ($C_{max} = 4$)

Note that both rules produce schedules bearing C_{max} value of 4. Total work content is equal to 7 time units. Both schedules do not contain any inserted idle time, so both schedules are giving optimal value of C_{max} which is equal to 4.

Example 3.7

Apply the LNS rule and solve the $P_2 \mid p_j = 1, \text{tree} \mid C_{max}$ problem. The precedence graph is shown below.



Solution:

For applying LNS rule, first find number of successors for all jobs.

Job (j)	1	2	3	4	5	6
NS(j)	1	1	0	2	0	2

The schedule of jobs using time scale is presented in Table 3.8 as well as a Gantt chart in Figure 3.11.

Table 3.8 Jobs schedule using LNS rule.

Time t	Set of Scheduled Jobs	Set of Unscheduled Jobs	Schedule on M_1	Schedule on M_2
0	{ }	All jobs	j_4 (NS = 2)	j_6 (NS = 2)
1	$\{j_4, j_2\}$	$\{j_1, j_3, j_5, j_6\}$ Schedulable jobs are j_5 and j_1	j_5 (NS = 0) Although j_2 has NS value of 1, but its predecessor is not yet scheduled	j_1 (NS = 1)
2	$\{j_4,j_2,j_1,j_5\}$	$\{j_2,j_3\}$ Schedulable job is j_2 only	None Job j_3's predecessor condition is not OK	j_2 (NS = 1}
3	All except job j_3	$\{j_3\}$ Job j_3 is schedulable	j_3 (NS = 0}	None

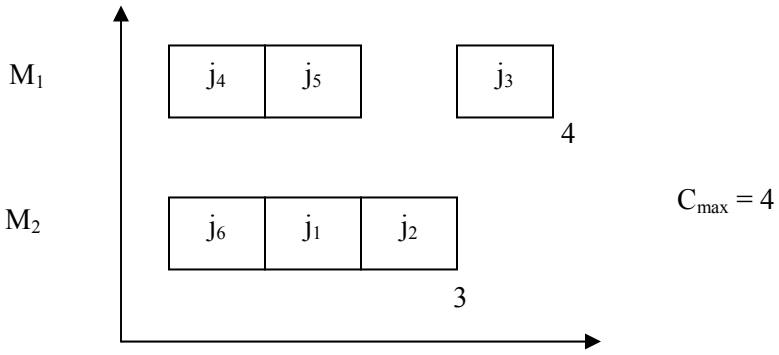


Figure 2 Gantt chart Using LNS rule.

Example 3.8
Apply CP rule and generate schedule for precedence graph of Example 3.7.
Solution:
The levels of the jobs in precedence graph are as under;

Jobs	Levels
j_1	3
j_2, j_4, j_6	2
j_3 , j_5	1

The job's scheduling using time scale is presented in Table 3.9.

Table 3.9 Job's scheduling by LNS rule.

Time t	Set of Scheduled Jobs	Set of Unscheduled Jobs	Schedule on M_1	Schedule on M_2
t=0	{ }	All jobs	Job j_1 has highest level of 3. So schedule job j_1	j_4 and j_6 at Level 2 can be scheduled. Pick job j_4
t=1	$\{j_4,j_1\}$	$\{j_2,j_3,j_6,j_5\}$ Job's Level Level 2 -> $j_2,j_6\}$ Level 1 -> $j_3,j_5\}$	Schedule j_2	Schedule j_6
t=2	All except job j_3,j_5	$\{j_3,j_5\}$	Schedule j_3	Schedule j_5
t=3	All jobs	{ }	***	***

The Gantt chart shows optimal schedule in Figure 3.12.

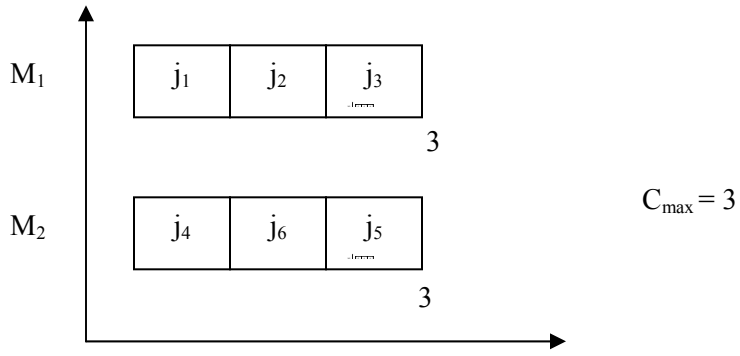


Figure 3.12 Gantt chart showing Optimal Schedule by LNS Rule

3.5 MINIMIZATION OF MAKESPAN WITH PREEMPTION *(P_m | prmp | C_{max})* PROBLEM

For P_m | prmp | C_{max}, under optimal schedule the makespan is as follows:

$$C_{max} \geq \max\left\{\frac{1}{m}\sum_{j=1}^{n} p_j, \max(p_j)\right\} = C^*_{max}$$

Remember, no job can be on two machines at the same time.

Proof:

For single machine problem with job preemption allowed, C_{max} is equal to $\sum_{j=1}^{n} p_j$. For parallel machines shop with m machines in parallel, the lower bound on C_{max} is clearly the average of the total process times; i.e., $\bar{p}$, where $\bar{p} = \frac{1}{m}\sum_{j=1}^{n} p_j$.

However, there is an exception to this rule as follows:
In case, certain job k has significant large value of process time P_k,
Such that, $P_k = \max(P_j)$.
If the value of $P_k > \bar{p}$, then lower bound on C_{max} will be defined by P_k. Hence, lower bound on C_{max} (C^*_{max}) is maximum of (P_k , $\bar{p}$). Suppose, we have the following scheduling problem

Jobs	1	2	3	4
Processing Time	2	1	3	12

Consider it to be a P_2 | *prmp* | C_{max} problem.

Average process time, $\bar{p} = \frac{2+1+3+12}{2} = 9$, and, $P_k = \max(2,1,3,12) = 12$. Hence, C_{max} is 12. The Gantt chart for the schedule is shown in Figure 3.13

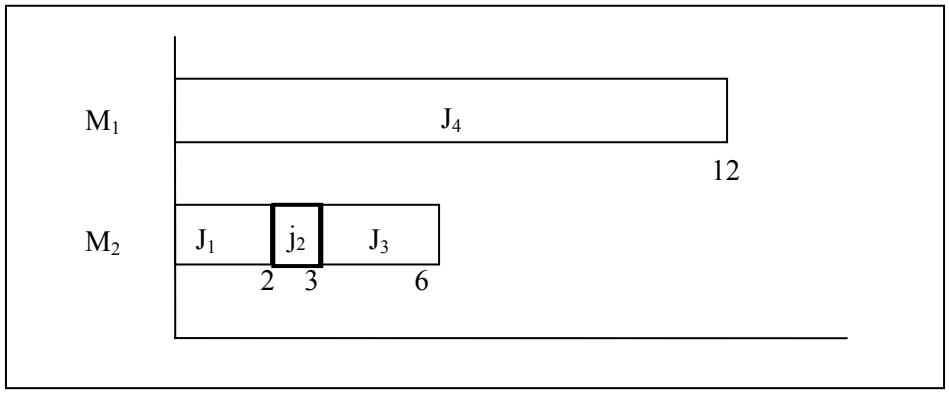


Figure 3.13 Gantt chart for optimal schedule by (P_m | prmp | C_{max}) problem**.**

3.6 LONGEST REMAINING PROCESSING TIME FIRST LRPT RULE $(P_m \mid prmp \mid C_{max})$

This problem deals with minimization of makespan in parallel machines environment when preemptions are allowed in discrete point in time. The following rule applies to this environment.

Rule:

LRPT (longest remaining process time first) yields an optimal schedule for $P_m \mid prmp \mid C_{max}$ *problem in discrete points in time.*

Example 3.9

The following data presents an instance of the $P_2 \mid prmp \mid C_{max}$.

job (j)	j_1	j_2	j_3
p_j	8	7	6

Use LRPT rule and find optimal schedule. Preemptions are allowed in discrete time.

Solution:

Using tabular method we will generate the schedule. One time unit in each row increases time. The jobs are given priority for assigning to machines M_1 and M_2 according to LRPT rule. Jobs having largest values of remaining process time are assigned to machines. If jobs with small value of remaining process time are currently assigned to machines, these jobs will be removed if jobs with higher values of LRPT are waiting in queue for their turn (Remember, preemptions are allowed). The complete procedure is presented in Table 3.10.

Table 3.10 Generation of schedule For $P_2 \mid prmp \mid C_{max}$ Problem.

start time	Remaining Process Time j1	j2	j3	Schedule Time	Job on M1	Job on M2
0	8	7	6	[0 , 1]	$j_1 \leftarrow$	$j_2 \leftarrow$
1	7	6	6	[1 , 2]	j_1	$j_2 \rightarrow$
2	6	5	6	[2 , 3]	j_1	$j_3 \leftarrow$
3	5	5	5	[3 , 4]	j_1	$j_3 \rightarrow$
4	4	5	4	[4 , 5]	$j_1 \rightarrow$	$j_2 \leftarrow$
5	3	4	4	[5 , 6]	$j_3 \leftarrow$	j_2
6	3	3	3	[6 , 7]	$j_3 \rightarrow$	j_2
7	3	2	2	[7, 8]	$j_1 \leftarrow$	$j_2 \rightarrow$
8	2	1	2	[8 , 9]	j_1	$J_3 \leftarrow$
9	1	1	1	[9 , 10]	$j_1 \rightarrow$	$j_3 \rightarrow$

start time	Remaining Process Time j1	j2	j3	Schedule Time	Job on M1	Job on M2
10	0	1	0	[10 , 11]	$j_2 \leftarrow$	

($\rightarrow$) indicates job's removal from the machine and ($\leftarrow$) indicates job's loading on the machine

Time, t =0 and time interval, [0, 1]

As shown in Table, jobs j_1 and j_2 have large values of remaining process time (RPT) than job j_3. So, assign j_1 to M_1 and, j_2 to M_2 from time interval [0, 1].

Time, t =1 and time interval, [1, 2]

The values of remaining process time for jobs j_1 and j_2 are updated. Remaining process time (RPT) for job j_1 is still largest for job j_1. So, job j_1 remains assigned to machine M_1. However, jobs j_2 and j_3 have equal amount of remaining process time. Since, job j_2 is currently assigned to machine M_2, it is kept assigned to M_2 during interval [1, 2].

Time, t=2 and time interval, [2, 3]

At the start of time 2, remaining process times for jobs j_1, j_2 and j_3 are 6, 5 and 6, respectively. Keep job j_1 assigned to M_1. Since job 3 has larger value of RPT than job 2, *remove job j_2 from M_2 and, assign j_3 to M_2 during time interval [2, 3]*. Preemption of job j_2 by j_3 is shown by arrows $\rightarrow$ and $\leftarrow$.

Increase time by one unit. Reduce the remaining process time accordingly for the jobs currently assigned to machines M_1 and M_2.

Complete the procedure till all jobs have RPT equal to zero as shown in last row of the Table 3.10.

Final schedule is shown by Gantt chart in Figure 3.14.

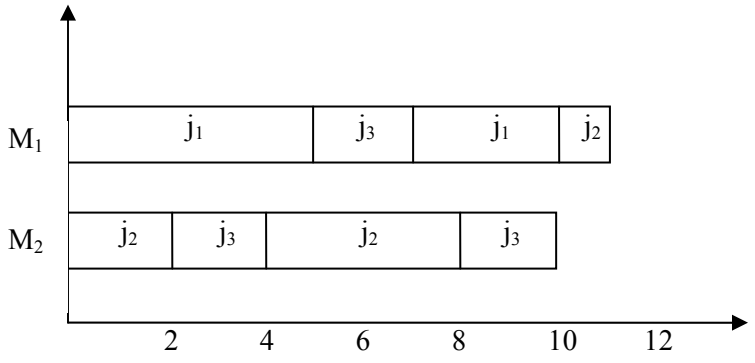


Figure 3.14 Gantt chart for LRPT schedule.

3.7 MINIMIZATION OF MAKESPAN FOR MACHINES WITH DIFFERENT SPEED AND WITH PREEMPTION ($Q_m \mid prmp \mid C_{max}$) PROBLEM

This problem relates to a parallel machine environment with m machines having varying processing speeds**.** The objective is to optimize makespan with preemptions allowed. The following rule applies to optimizing C_{max} in this environment.

Rule:

Assign jobs with largest remaining process time (LRPT) to fastest machines (FM). Every time a fastest machine completes a job, the job on the second fastest machine is transferred to fastest machine, and , so on. This rule is called LRPT-FM rule.

Example 3.10

Consider three machines in parallel with varying speeds as follows:

Machine	M_1	M_2	M_3
Speed (v_i)	3	2	1

Three jobs are to be processed on these machines having process times as follows:

job	j_1	j_2	j_3
p_j	36	34	12

Generate schedule applying LRPT-FM rule in discrete time.

Solution:

The schedule generated is shown in Table 3.11 below.

Table 3.11 Schedule for Q_m| prmp |C_{max} Problem.

start time	**Remaining P. T.** j_1	j_2	j_3	**Schedule Time**	**Job on M_1 (3)**	**Job on M_2 (2)**	**Job on M_3 (1)**
0	36	34	12	[0 , 1]	j_1	j_2	j_3
1	33	32	11	[1 , 2]	j_1	j_2	j_3
2	30	30	10	[2 , 3]	j_1	j_2	j_3
3	27	28	9	[3 , 4]	j_2	j_1	j_3
4	25	25	8	[4 , 5]	j_2	j_1	j_3
5	23	22	7	[5 , 6]	j_1	j_2	j_3
6	20	20	6	[6 , 7]	j_1	j_2	j_3

start time	Remaining P. T. j_1	j_2	j_3	Schedule Time	Job on M_1 (3)	Job on M_2 (2)	Job on M_3 (1)
7	17	18	5	[7, 8]	j_2	j_1	j_3
8	15	15	4	[8 , 9]	j_2	j_1	j_3
9	13	12	3	[9 , 10]	j_1	j_2	j_3
10	10	10	2	[10 , 11]	j_1	j_2	j_3
11	7	8	1	[11 , 12]	j_2	j_1	j_3
12	5	5	-	[12 , 13]	j_2	j_1	-
13	3	2	-	[13 , 14]	j_1	j_2	-
14	-	-	-				

Note that preemptions between jobs j_1 and j_2 occur at times; t=3, 5, 7, 9, 12 and 14 on machines M_1 and M_2**.**

3.8 MINIMIZATION OF TOTAL COMPLETION TIME FOR MACHINES WITH DIFFERENT SPEED AND WITH PREEMPTION (Q_m | prmp | ΣC_j) PROBLEM

This problem again pertains to parallel machine environment with m machines; each machine having varying speed. Objective is to minimize total completion times when job preemption is allowed.

The following rule is applied to optimize the objective function.

Rule:

Jobs with shortest remaining process time (SRPT) are assigned to fastest machine (FM). Every time, fastest machine completes a job, the job on the second fastest machine moves to the fastest machine. Preemptions of jobs are allowed at integer points in time.

Example 3.11

Consider 4-machine and 7-job problem with an instance of Q_4 | *prmp* | ΣC_j.

Machine	M_1	M_2	M_3	M_4
speed (vi)	4	2	2	1

Jobs are processed on these machines having process time as follows:

job (j)	j_1	j_2	j_3	j_4	j_5	j_6	j_7
pj	8	16	34	40	45	46	61

Find optimal schedule using SRPT-FM rule. Preemptions are allowed at discrete point in time.

Solution:

Arrange jobs according to SPT values. Similarly arrange machines in decreasing order of speed; fastest machine first in order, and, slowest machine last in order.

At time, t=0.

Assign job j_1 to machine M_1, j_2 to M_2, j_3 to M_3 and j_4 to M_4.

At time, t=2.

Job j_1 is completed. Move job j_2 to M_1, job j_3 to M_2, j_4 to M_3. Also, assign job j_5 to M_4.

At time, t=5.

Job j_2 is complete on machine M_1. Shift job j_3 to machine M_1, j_4 to M_2, j_5 to M_3. Assign job j_6 to M_4.

Proceed in this manner to finish all jobs. Complete schedule is presented in the Table 3.12 below.

Table 3.12 Schedule generated using SRPT-FM Rule**.**

start time	**Remaining P. T.** j_1	j_2	j_3	j_4	**Schedule Time**	**Job on M_1 (4)**	**Job on M_2 (2)**	**Job on M_3 (2)**	**Job on M_4 (1)**
0	8	16	34	40	[0 , 1]	j_1	j_2	j_3	j_4
1	4	14	32	39	[1 , 2]	j_1	j_2	j_3	j_4
2 ←	**j_2** 12	**j_3** 30	**j_4** 38	**j_5** 45	[2 , 3]	j_2	j_3	j_4	j_5
3	8	28	36	44	[3 , 4]	j_2	j_3	j_4	j_5
4	4	26	34	43	[4 , 5]	j_2	j_3	j_4	j_5
5 ←	**j_3** 24	**j_4** 32	**j_5** 42	**j_6** 46	[5 , 6]	j_3	j_4	j_5	j_6
6	20	30	40	45	[6 , 7]	j_3	j_4	j_5	j_6
7	16	28	38	44	[7, 8]	j_3	j_4	j_5	j_6
8	12	26	36	43	[8 , 9]	j_3	j_4	j_5	j_6
9	8	24	34	42	[9 , 10]	j_3	j_4	j_5	j_6

start time	Remaining P. T. j_1	j_2	j_3	j_4	Schedule Time	Job on M_1 (4)	Job on M_2 (2)	Job on M_3 (2)	Job on M_4 (1)
10	4	22	32	41	[10 , 11]	j_3	j_4	j_5	j_6
11 ←	**j_4** 20	**j_5** 30	**j_6** 40	**j_7** 61	[11 , 12]	j_4	j_5	j_6	j_7
12	16	28	38	60	[12 , 13]	j_4	j_5	j_6	j_7
13	12	26	36	59	[13 , 14]	j_4	j_5	j_6	j_7
14	8	24	34	58	[14 , 15]	j_4	j_5	j_6	j_7
15	4	22	32	57	[15 , 16]	j_4	j_5	j_6	j_7
16 ←		**j_5** **20**	**j_6** **30**	**j_7** **56**	[16 , 17]	j_5	j_6	j_7	
17		16	28	54	[17 , 18]	j_5	j_6	j_7	
18		12	26	52	[18 , 19]	j_5	j_6	j_7	
19		8	24	50	[19 , 20]	j_5	j_6	j_7	
20		4	22	48	[20 , 21]	j_5	j_6	j_7	
21 ←			**j_6** 20	**j_7** 46	[21 , 22]	j_6	j_7		
22			16	44	[22 , 23]	j_6	j_7		
23			12	42	[23, 24]	j_6	j_7		
24			8	40	[24 , 25]	j_6	j_7		
25			4	38	[25 , 26]	j_6	j_7		
26 ←				**j_7** 36	[26 , 27]	j_7			
27				32	[27 , 28]	j_7			
28				28	[28 , 29]	j_7			
29				24	[29 , 30]	j_7			
30				20	[30 , 31]	j_7			
31				16	[31 , 32]	j_7			

start time	Remaining P. T. j_1 j_2 j_3 j_4	Schedule Time	Job on M_1 (4)	Job on M_2 (2)	Job on M_3 (2)	Job on M_4 (1)
32	12	[32 , 33]	j_7			
33	8	[33 , 34]	j_7			
34	4	[34 , 35]	j_7			
35	-	[35 , 36]				

Gantt Chart for the schedule generated in Table 3.12 is;

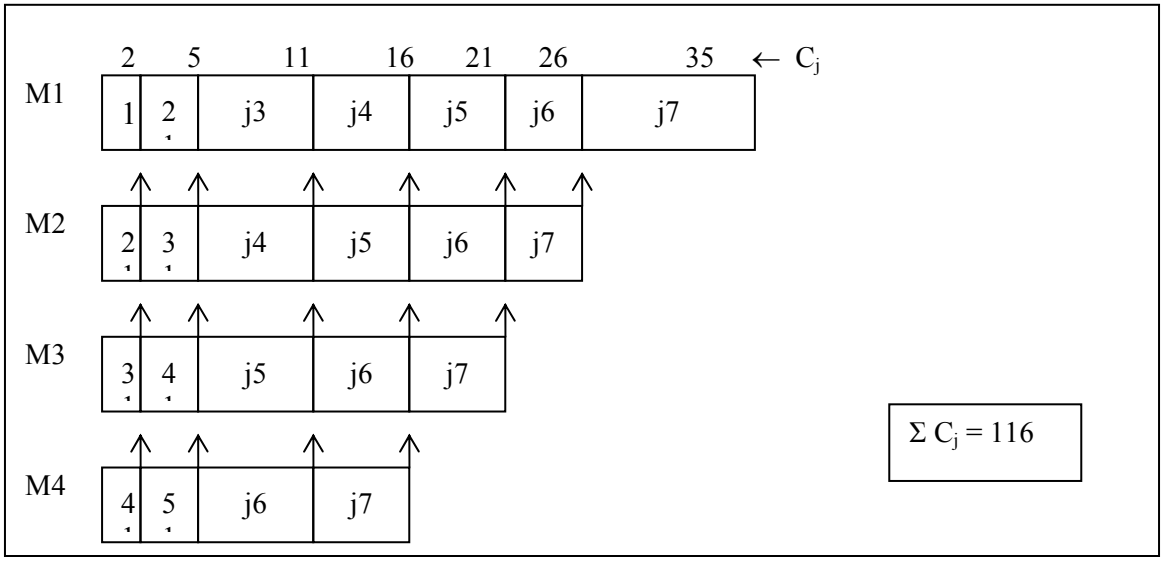


Figure 3.15 Gantt chart For schedule presented in Table 3.12.

3.9 MINIMIZATION OF TOTAL MAXIMUM LATENESS WITH PREEMPTION ($P_m \mid prmp \mid L_{max}$) PROBLEM.

The problem pertains to parallel machines environment with job preemption allowed. Objective function is minimization of maximum lateness**.** To solve this type of problem data is converted to $P_m \mid r_j , prmp \mid C_{max}$ problem. Then by using LRPT rule, a schedule is generated. By reversing the obtain schedule we will get schedule of jobs on the parallel machines. The procedure is explained by an example.

Example 3.13

The following data is an instance of $P_2 \mid prmp \mid L_{max}$. Generate schedule. Preemptions are allowed discrete in time.

Jobs (j)	j_1	j_2	j_3	j_4
d_j	4	5	8	9
p_j	3	3	3	8

Solution:

Find r_j values as under;

Let $d^* = \max (d_1, d_2, d_3, d_4) = (4 , 5 ,8 , 9) = 9$

Set $r_4 = d^* - d_4 = 9 – 9 = 0,$
$r_3 = d^* - d_3 = 9 – 8 = 1,$
$r_2 = d^* - d_2 = 9 – 5 = 4,$
$r_1 = d^* - d_1 = 9 – 4 = 5$

Using these values of r_j, formulate new problem as under;

job	j_1	j_2	j_3	j_4
r_j	5	4	1	0
p_j	3	3	3	8

The schedule generate is shown in table 3.13 below,

Table 3.13 Solution for problem using LRPT rule.

Start time	**Remaining Process Time** j_1 r_1=5	j_2 r_2=4	j_3 r_3=1	j_4 r_4=0	**Schedule Time**	**Job on M_1**	**Job on M_2**
0	3	3	3	(8)	[0 , 1]	j_4	
1	3	3	(3)	(7)	[1 , 2]	j_4	j_3
2	3	3	(2)	(6)	[2 , 3]	j_4	j_3
3	3	3	(1)	(5)	[3 , 4]	j_4	j_3
4	3	(3)		(4)	[4 , 5]	j_4	j_2
5	(3)	2		(3)	[5 , 6]	j_4	j_1
6	(2)	2		(2)	[6 , 7]	j_4	j_1
7	1	(2)		(1)	[7, 8]	j_4	j_2
8	(1)	(1)			[8 , 9]	j_1	j_2
9					[9 , 10]		

Release time r_j has significant influence on scheduling decision. Job j_1 can be assigned to machine not before time, t=5. This is the release time of job j_1. Similarly, job 3 can be assigned at time, t=1. Job j_2 can be assigned not before time, t= 4. Gantt chart of the schedule is shown in Figure 3.16 below.

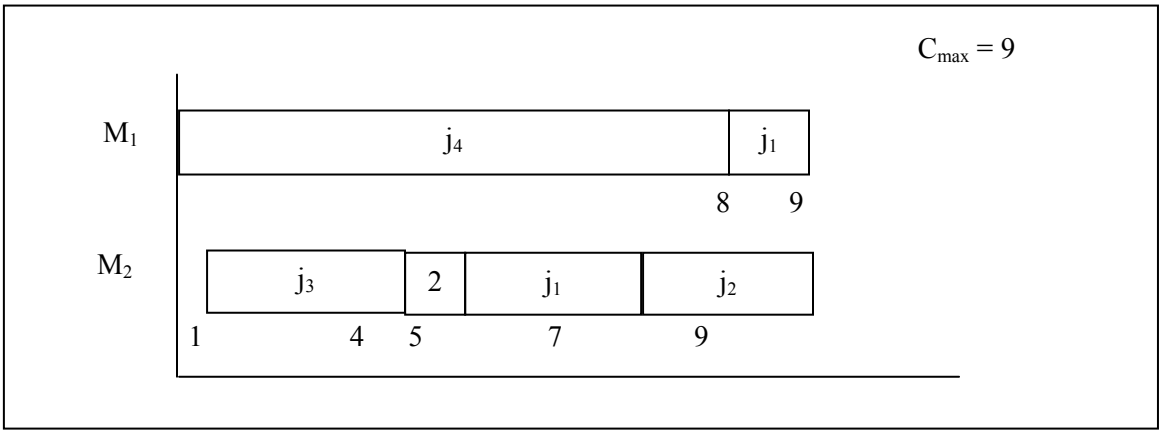


Figure 3.16 Gantt chart using LRPT Schedule.

Reverse the Gantt chart to find completion times (C_j) of jobs as shown in Figure 3.17 below;

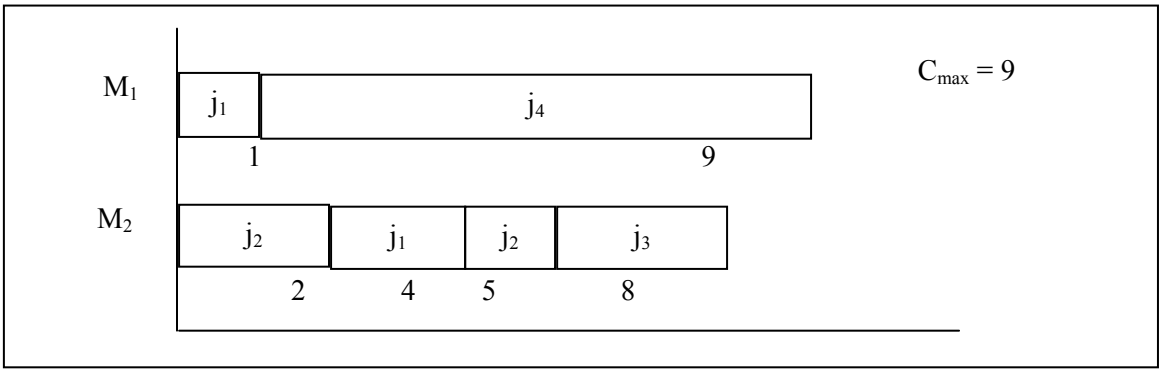


Figure 3.17 Reverse Gantt chart.

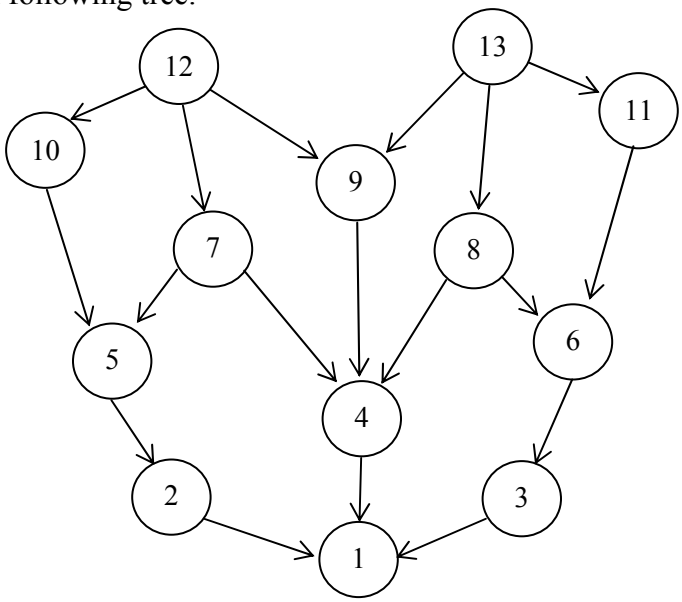
Then, the L_{max} can be found as follows:

J	**D_j**	**C_j**	**L_j**
1	4	4	0
2	5	5	0
3	8	8	0
4	9	9	0

Hence, $L_{max} = 0$

EXERCISES

3.1 Consider an instance of $P_4 \mid p_j=1, \text{tree} \mid C_{max}$ problem with precedence constraints as shown in following tree.

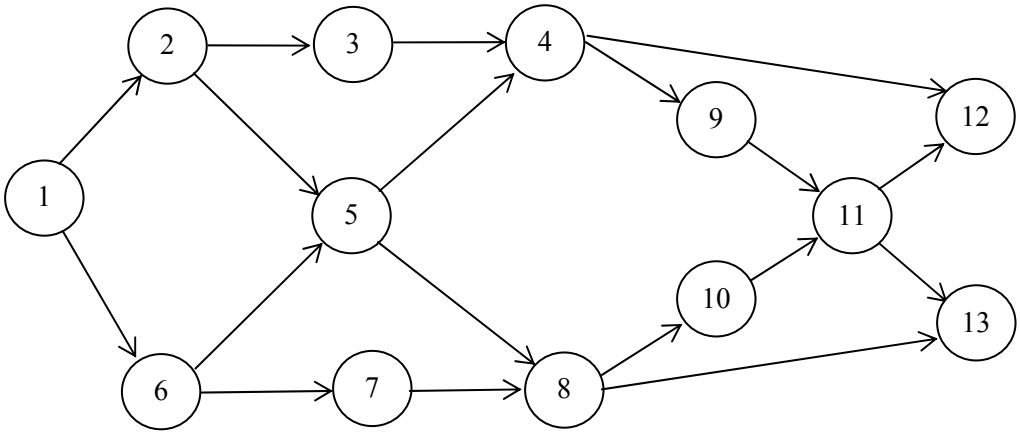


Generate schedule and find C_{max}, using
i) CP rule, ii) LNS rule

3.2 Consider an instance of the $P_\infty \mid \text{prec} \mid C_{max}$ problem with data for 13 jobs as follows:

Job	1	2	3	4	5	6	7	8	9	10	11	12	13
p_j	7	8	4	9	12	5	3	9	5	12	7	5	8

The precedence graph is shown below.



Assume infinite number of machines, and find C_{max} as well as identify critical path.

3.3 Consider an instance of $P_3 \mid\mid C_{max}$ problem with data as under:

Job	1	2	3	4	5	6	7	8
p_j	15	11	9	6	12	18	9	14

i) Apply LPT rule and assign jobs to machines. Is the solution optimal? Why or why not.
ii) Apply load balancing heuristics and find C_{max}.

3.4 Suppose the data in Question #3.3 pertains to an instance of $P_3 \mid prmp \mid C_{max}$. Generate schedule using LRPT rule and find C_{max}. Does preemption improve solution?

3.5 The following data represents an instance of $P_2 \mid prmp \mid C_{max}$ Problem

job	1	2	3	4	5
p_j	4	3	13	12	2

Generate schedule to minimize C_{max}. Also draw Gantt chart. Preemptions are allowed at discrete intervals of time

3.6 Consider $P_4 \mid\mid C_{max}$ problem with the following data:

Job	1	2	3	4	5	6	7	8	9	10	11
p_j	2	2	3	4	5	6	7	8	2	2	3

Find a sequence that minimize C_{max} and compute C_{max}. Then, prove that the optimal sequence found is optimal by a principle.

3.7 Consider the following data pertaining to problem as an instance of $Q_3 \mid prmp \mid C_{max}$ There are three machines in parallel with varying speeds as follows:

Machine	M_1	M_2	M_3
Speed (v)	2	1	4

Four jobs are to be processed with process time as follows:

Jobs	1	2	3	4
Process time	12	8	16	24

Generate schedule applying LRPT-FM rule in discrete intervals of time.

3.8 Consider 3-machine and 6-job problem with an instance of $Q_3 \mid prmp \mid \Sigma C_j$.

Machine	M_1	M_2	M_3
Speed (v_i)	6	4	2

Six jobs are to be processed with process time as follows:

Job	1	2	3	4	5	6
P_j	12	16	33	48	56	38

Find optimal schedule using SRPT-FM rule. Preemptions are allowed at discrete point in time

3.9 The following data is an instance of $P_3 \mid prmp \mid L_{max}$

Job	1	2	3	4	**5**
P_j	4	2	1	6	7
D_j	5	7	3	14	12

Generate a schedule to minimize $L_{max.}$ Preemptions are allowed at discrete intervals of time.

3.10 The following data is an instance of $P_2 \mid prmp \mid L_{max}$. Generate a schedule to minimize L_{max}. Preemption is allowed in discrete time.

Job	1	2	3	4	5
D_j	16	9	25	21	13
p_j	3	5	9	7	10

3.11 Consider $P_2 \mid \mid n_t$ Problem with the following data:

Job	1	2	3	4	5	6	7	8	9
D_j	8	12	7	3	17	14	16	23	22
P_j	7	5	9	2	8	8	5	2	1

Generate a sequence to minimize the total number of jobs tardy and then compute the total number of jobs tardy and the makespan.

3.12 Consider $Q_4 \mid prmp \mid C_{max}$ Problem with the following data:

Job	1	2	3	4	5	6	7	8	9
P_j	17	5	9	12	8	8	5	22	10

In which the four machines processing speeds are as follows:

Machine	M_1	M_2	M_3	M_4
Speed (v_i)	3	4	2	2

Generate a sequence to minimize the makespan. Then, constructing the Gantt chart for the solution obtained and computes the makespan.

3.13 Consider $Q_2 \mid prmp \mid \Sigma W_j C_j$ Problem with the following data:

Job	1	2	3	4	5
Process time	17	5	9	12	8
Weight	1	2	4	6	4

In which the two machines processing speeds are as follows:

Machine	M_1	M_2
Processing Speed (v_i)	1	2

Generate a sequence to minimize the total weighted completion time. Then, constructing the Gantt chart for the solution obtained and compute the total weighted completion time.

3.14 The following data represents an instance of $P_2 \mid prmp \mid L_{max}$ Problem

job	1	2	3	4	5
d_j	8	12	7	3	17
p_j	7	5	9	2	8

Generate schedule to minimize L_{max}. Also draw Gantt Chart. Preemptions are allowed at discrete intervals of time.

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 ($F_2 \mid\mid C_{max}$) 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.

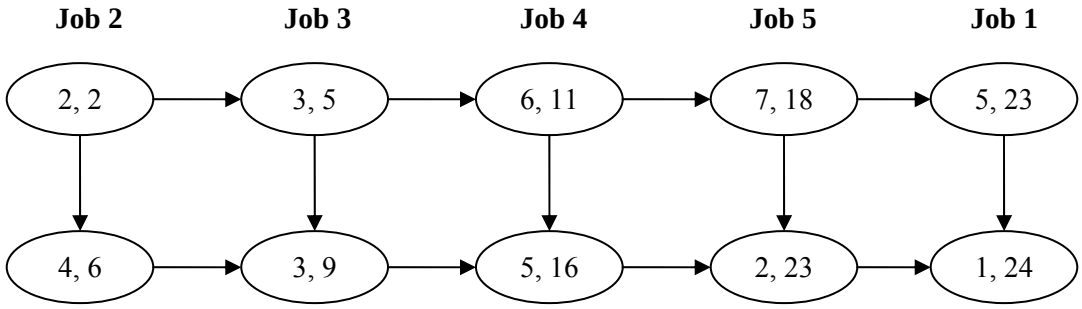


Figure 4.1 Directed Graph For Optimal Sequence { j_2, j_3, j_4, j_5, j_1}

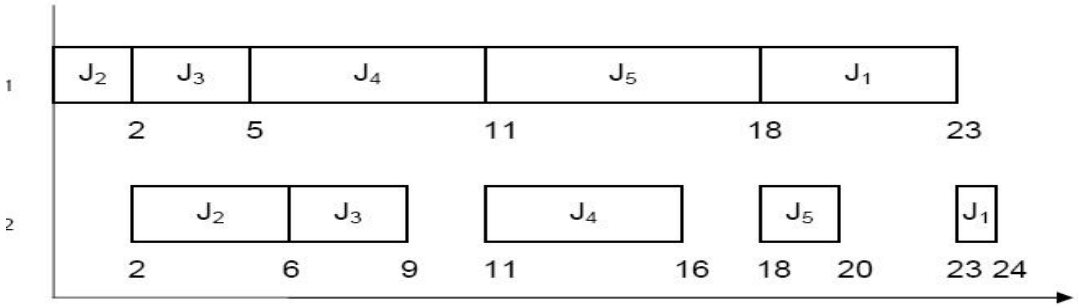


Figure 4.2 Gantt chart For optimal sequence { j_2, j_3, j_4, j_5, j_1}

4.3 MINIMIZATION OF MAKESPAN FOR ($F_3 \mid\mid C_{max}$) 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 ($F_m \mid\mid C_{max}$) 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 $F_3 \mid\mid C_{max}$ 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;

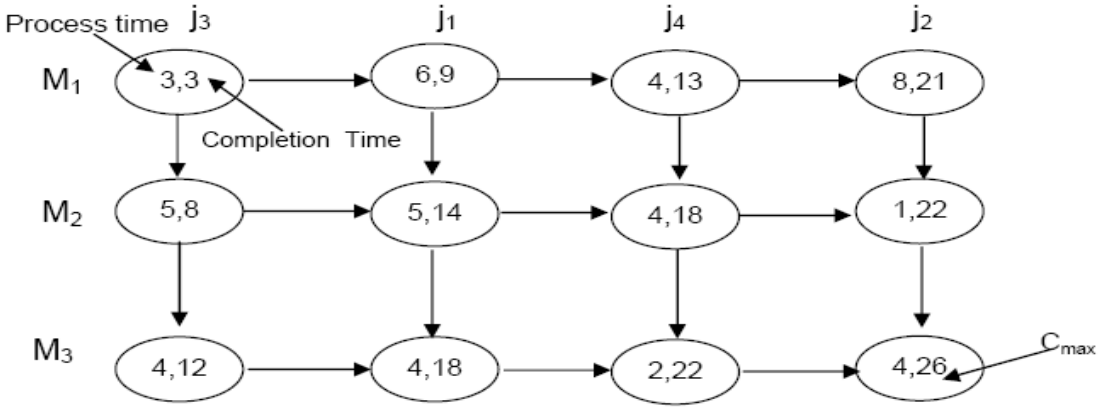


Figure 3.3 Directed graph for seq. { j_3 , j_1, j_4 , j_2 }

Directed graph for sequence 2 is;

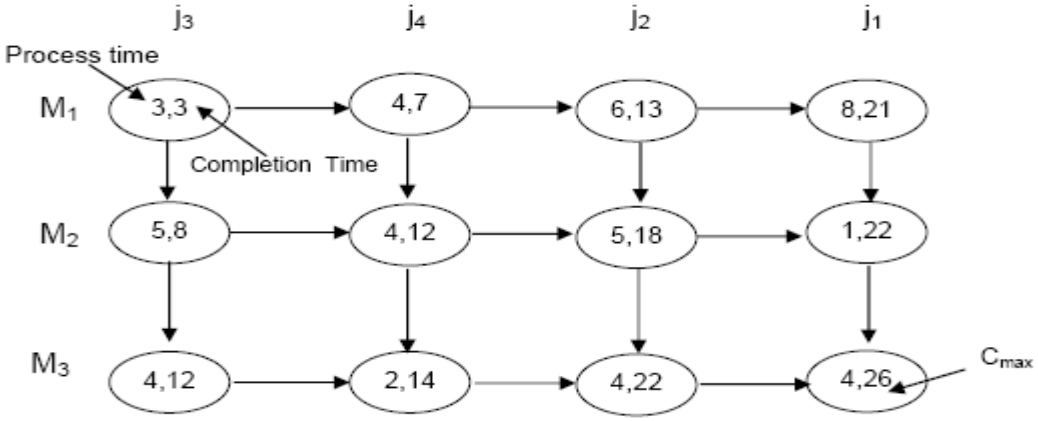


Figure 3.4 Directed graph for sequence { j_3 , j_4, j_1 , j_2 }

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

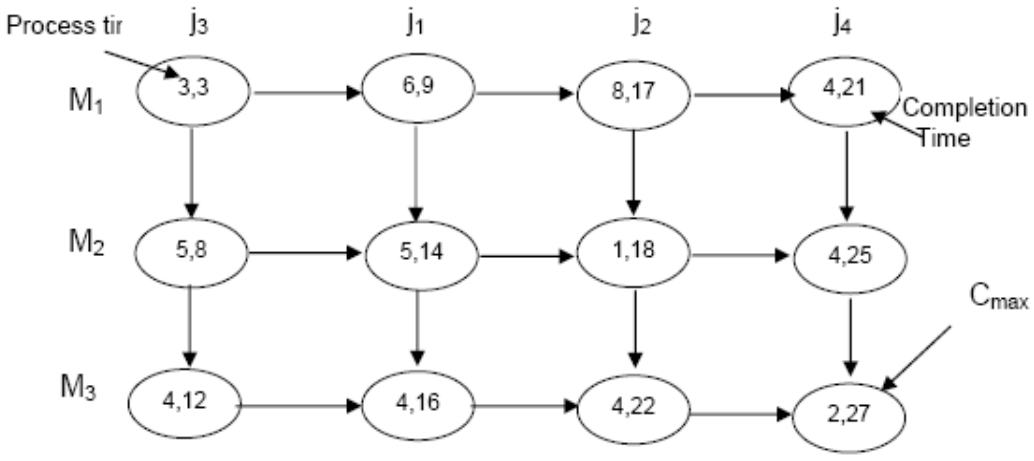


Figure 4.5 Directed Graph For Sequence/Schedule $\{j_3, j_1, j_2, j_4\}$

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

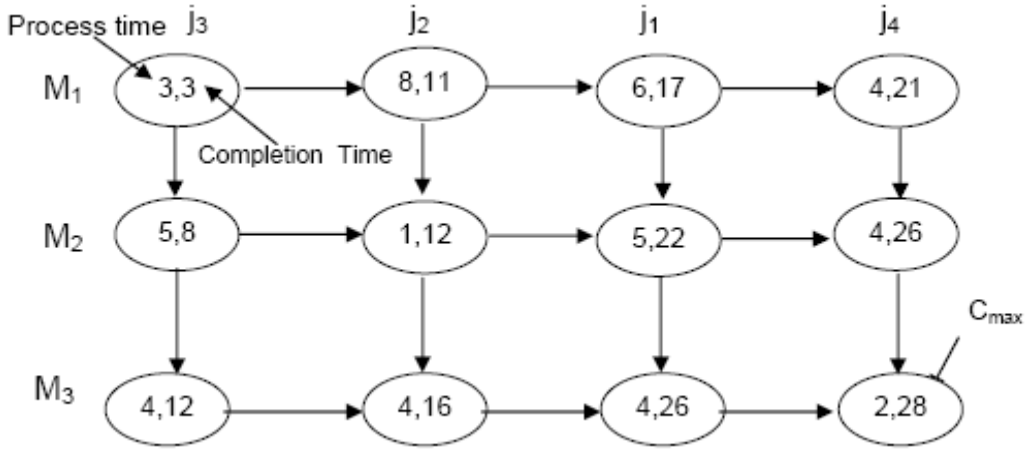


Figure 4.6 Directed graph for sequence/schedule {j_3, j_2, j_1, j_4}

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

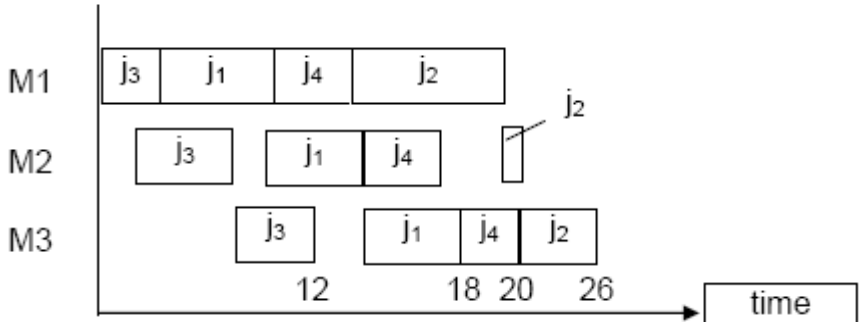


Figure 4.7 Gantt chart for sequence $\{j_3, j_1, j_4, j_2\}$.

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

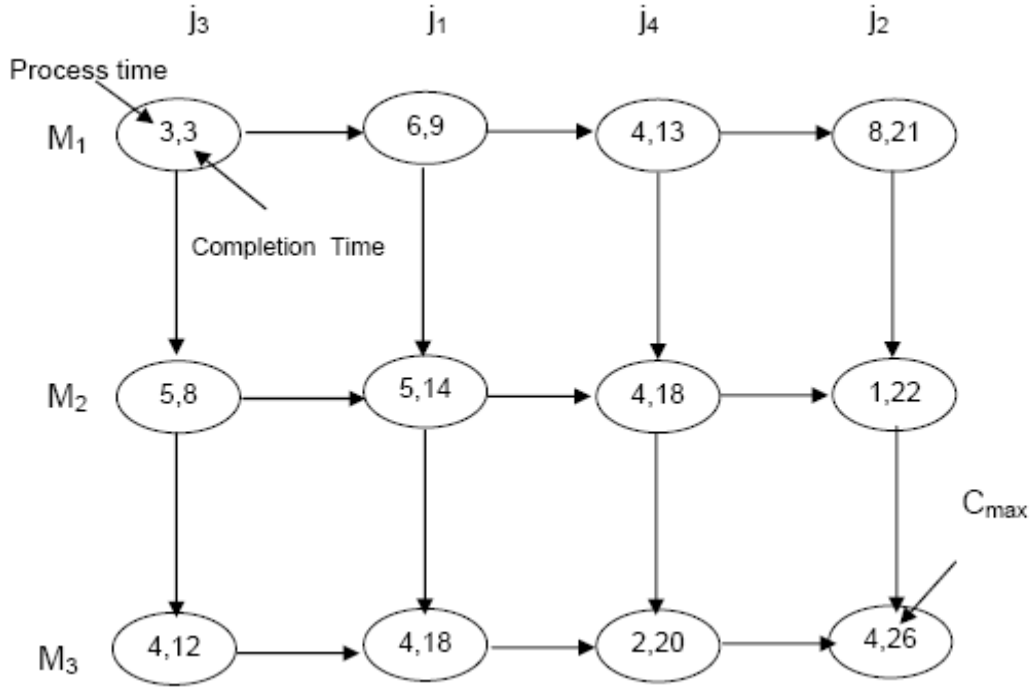


Figure 4.8 Directed graph for schedule $\{j_3, j_1, j_4, j_2\}$

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

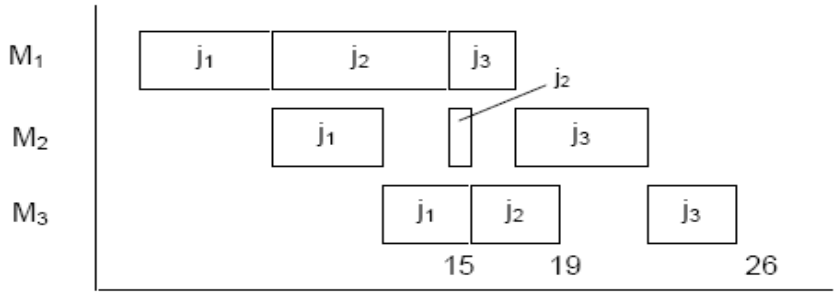


Figure 4.8 Gantt chart for the partial sequence S_{123*}: (j_1, j_2, j_3, *)

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.

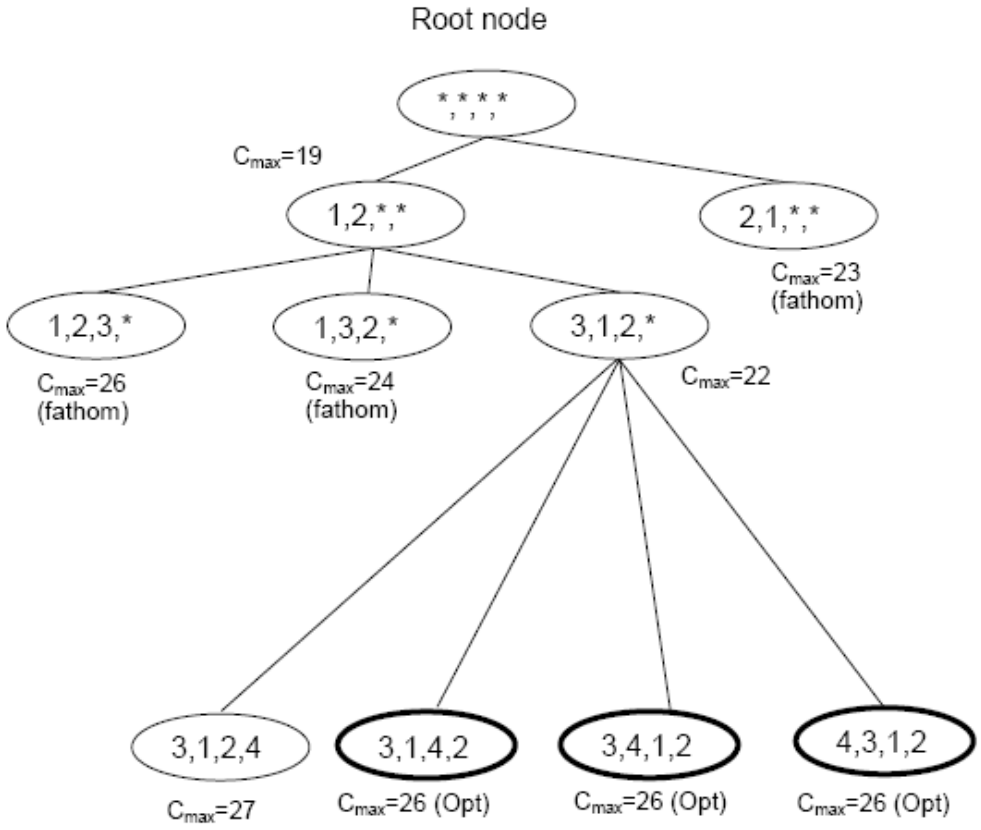


Figure 4.9 Enumeration tree for four-job problem using NEH algorithm**.**

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:

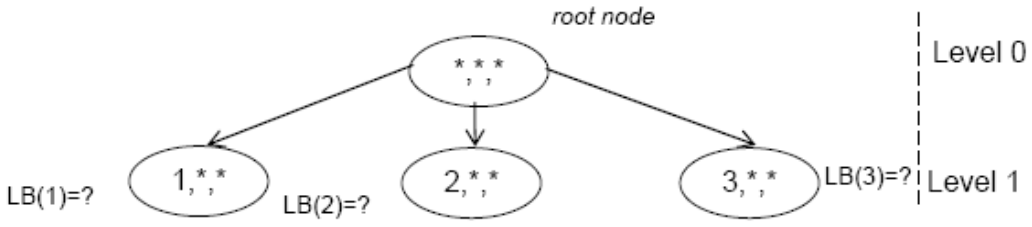


Figure 4.10 Branch and Bound Tree with Level 1 Nodes

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

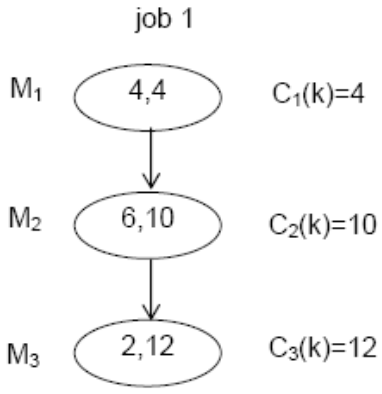


Figure 4.11 Directed graph for the partial sequence (1, *, *)

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

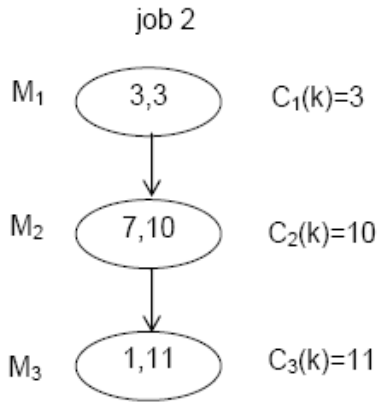


Figure 4.12 Directed graph for the partial sequence (2, *, *).

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
	Σp_{1j}=	Σp_{2j}=	Σp_{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.

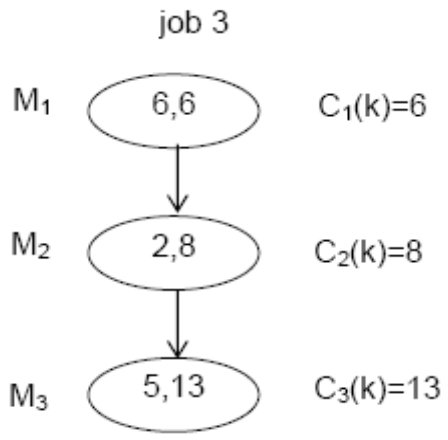


Figure 4.13 Directed graph for the partial sequence (3, *, *).

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}=	$\Sigma p2j$=	$\Sigma p3j$=	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.

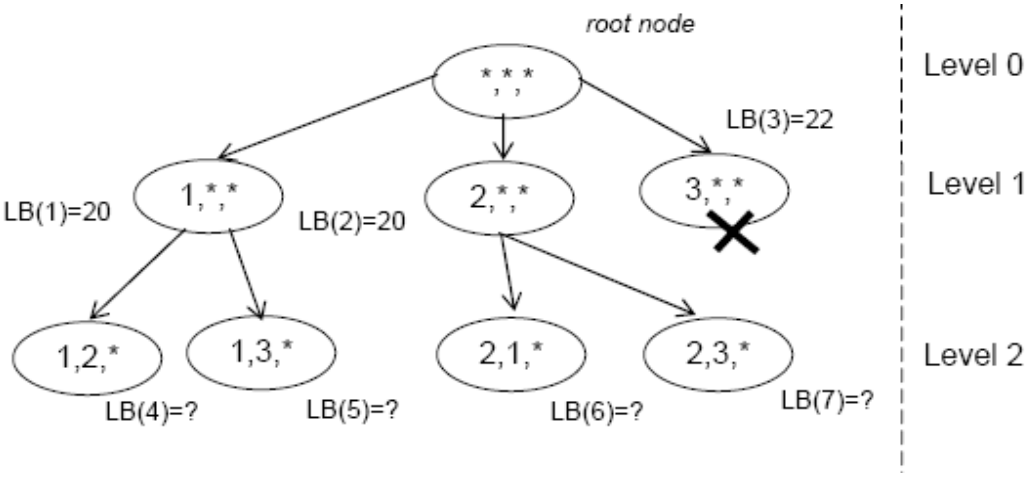


Figure 4.14 Branch and bound tree with level two nodes.

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.

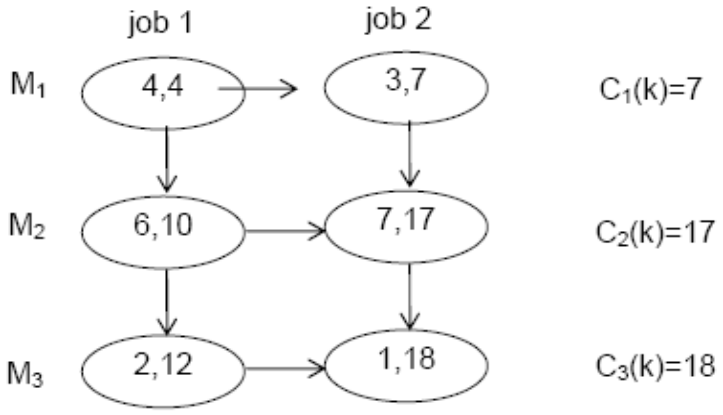


Figure 4.15 Directed graph for partial sequence.

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
	$\Sigma p_{1j}=$	$\Sigma p_{2j}=$	$\Sigma p_{3j}=$	min	min
	6	2	5	7	5

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

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

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

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

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.

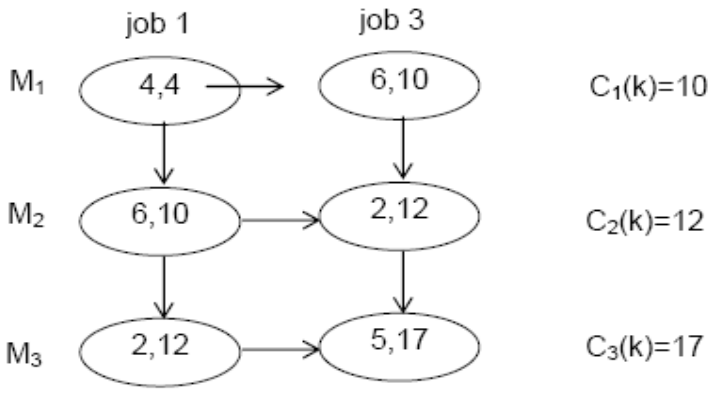


Figure 4.16 Directed graph for partial sequence (1, 3, *).

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.

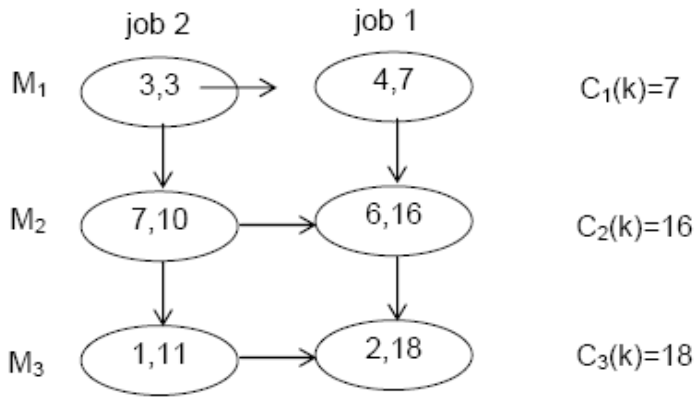


Figure 4.17 Directed graph for the partial sequence (2, 1, *).

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.

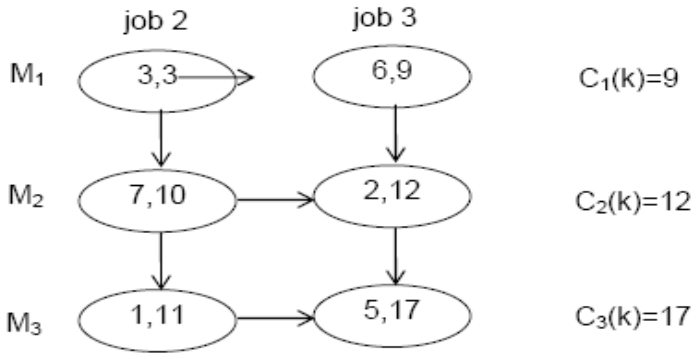


Figure 4.18 Directed graph for partial sequence (2, 3, *)

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.

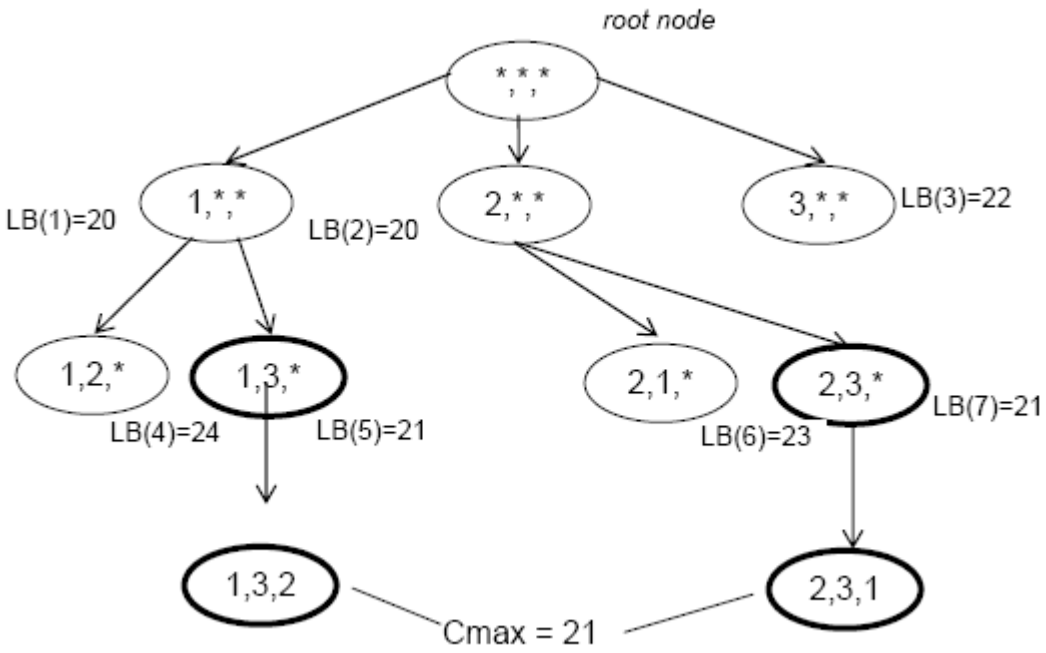


Figure 1 The complete branch and bound tree with an optimal solution.

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.

5

Job Shop Scheduling

CHAPTER CONTENTS

5.1 INTRODUCTION

The job shop comprises m machines and n jobs. Jobs have well defined processing order. However, possible number of schedules reaches to $(n!)^m$ for total enumerations. A variety of schedule generation techniques are in practice; namely, semi-active schedules, active schedules and non-delay schedules. Recently, an efficient schedule generation technique has been devised. It is called shifting bottleneck heuristic. It provides good solution in minimum amount of computer time.

5.2 EARLIEST START TIME (EST) SCHEDULE GENERATION

This methodology assigns priority to jobs having minimum EST values from among candidate jobs waiting in queue at a machine. The job having minimum value of EST is selected for assigning to the machine.

Example 5.1

Consider the $J_3 \parallel C_{max}$ instance with the following data for 3-job and 3-machine problem:

	Proc Time			Machine No		
Job	O_1	O_2	O_3	O_1	O_2	O_3
1	5	2	4	1	2	3
2	3	7	3	2	3	1
3	8	6	5	1	3	2

Generate an active schedule based on earliest start times (EST) of the schedulable operations as explained in the class. Use SPT rule to break ties arbitrarily.

Solution:

From the given data, form schedulable jobs list (S) at time zero as follows:

$$S = \{(j_1, O_1), (j_2, O_1), (j_3, O_1)\}$$

Compute values of EST for all jobs/operations in the list. Remember, first operation of every job will be included for EST calculations. In order to start building the schedule, the following notations need to be defined:

MAT: indicates the time at which required machine will be processing the particular job.

JAT: indicates the time at which job will be available for processing of the particular operation.

At time zero, all the values of JAT will be zero for static job shop environment.

Table 5.1 Computation of EST for all jobs at time zero

Job	Opn.	P_j	M_j	MAT	JAT	EST
1	1	5	1	0	0	0
2	1	3	2	0	0	0
3	1	8	1	0	0	0

Jobs j_1 and j_3 have scheduling conflict as these jobs require machine M_1 at the same time. Using SPT as tie-breaker, select job j_1 to be scheduled on machine M_1 as job j_1 process time is less than job j_3 process time on machine M_1. Job j_2 has operation O_1 on machine M_2. Since there is no schedule conflict at machine M_2 at time zero, schedule job j_2 on machine M_2. The earliest completion times (ECT) are shown in table 5.2 below.

Table 5.2 Earliest completion time for schedule (j_1, j_2)

Job	Opn	Pj	Mj	MAT	JAT	EST	ECT
1	1	5	1	0	0	0	5
2	1	3	2	0	0	0	3
3	1	8	1	0	0	0	------

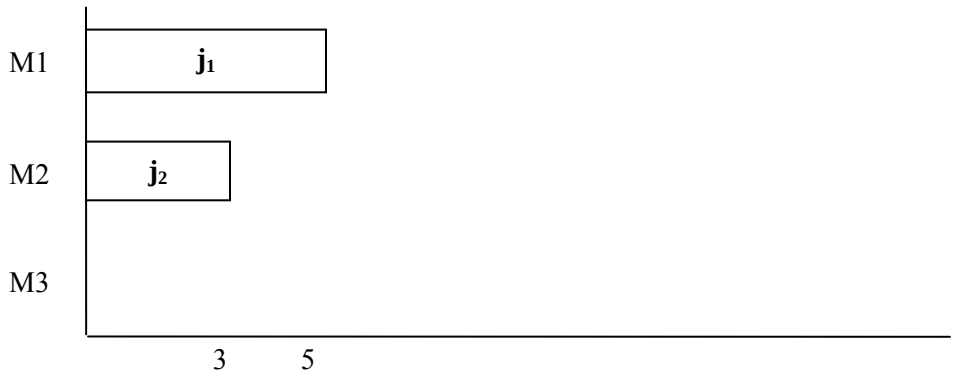


Figure 5.1 Gantt chart for the partial schedule.

From the Gantt chart for the partial schedule above, machine available times (MAT) for next processes in sequence are;

Machine	M_1	M_2	M_3
MAT	5	3	0

Job available times (JAT) for next operations in sequence are;

Job	j_1	j_2	j_3
JAT	5	3	0

Updating schedulable jobs list S, delete processes (j_1,O_1) and (j_2,O_1) from the list. Add the next processes for jobs j_1 and j_2 in the list. Then, the new list of schedulable jobs:

$$S = \{(j_1, O_2), (j_2, O_2), (j_3, O_1)\}$$

The computations of EST values for the new processes are shown below in the table 5.3.

Table 5.3 Computation of EST for the new processes

Job	Opn	Pj	Mj	MAT	JAT	EST
1	2	2	2	3	5	5
2	2	7	3	0	3	3
3	1	8	1	5	0	5

Since, there is no schedule conflict, schedule jobs j_1, j_2 and j_3 on machines M_2, M_3 and M_1 respectively with earliest completion times (ECT) as shown in the following Gantt chart.

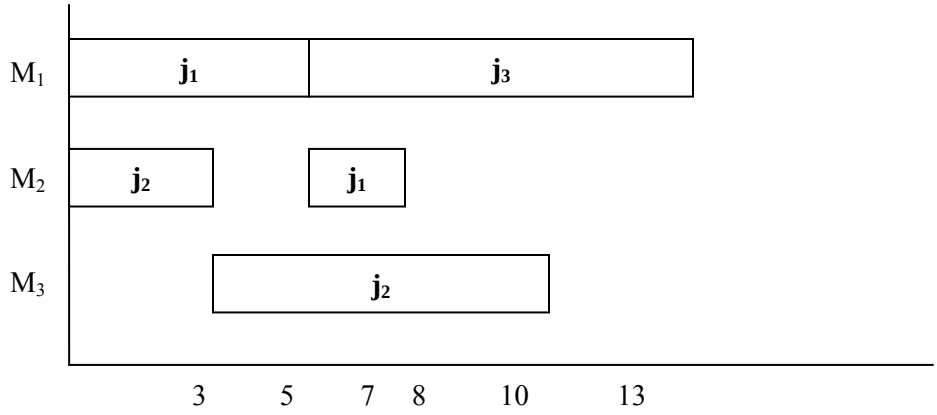


Figure 5.2 Gantt chart for partial schedule

From the Gantt chart for the partial schedule above, machine available times (MAT) for next processes in sequence are;

Machine	M_1	M_2	M_3
MAT	13	7	10

Job available times (JAT) for next operations in sequence are;

Job	j_1	j_2	j_3
JAT	7	10	13

Updating schedulable jobs list S, by deleting processes (j_1, O_2), (j_2, O_2) and (j_3, O_1) from the list. Add the next processes for jobs j_1, j_2 and j_3 in the list. Then, the new list of schedulable jobs:

$$S = \{(j_1, O_3), (j_2, O_3), (j_3, O_2)\}$$

The computations of EST values for the new processes are shown below in the table 5.4.

Table 5.4 Computation of EST for the new processes

Job	Opn	Pj	Mj	MAT	JAT	EST
1	3	4	3	10	7	10
2	3	3	1	13	10	13
3	2	6	3	10	13	13

Jobs j_1 and j_3 require machine M_3 for next scheduling. Since job j_1 value of EST is smaller than EST value of job j_3, schedule job j_1 on machine M_3 at time 10. Job j_2 has no machine conflict, schedule job j_2 on machine M_3 at time 13. The resulting Gantt chart in fig 5.3 shows the partial schedule.

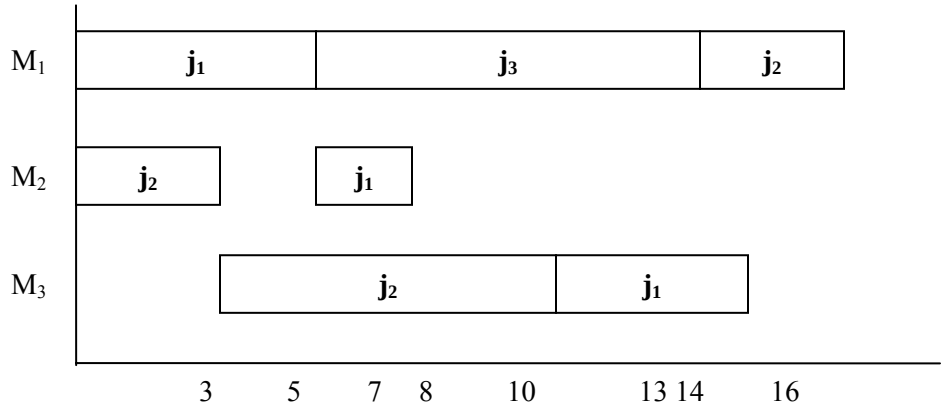


Figure 5.3 Gantt chart for partial schedule.

From the Gantt chart for the partial schedule above, machine available times (MAT) for next processes in sequence are;

Machine	M_1	M_2	M_3
MAT	16	7	13

Job available times (JAT) for next operations in sequence are;

Job	j_1	j_2	j_3
JAT	14	16	13

Updating schedulable jobs list S, by deleting processes (j_1, O_3) and (j_2, O_3) from the list. Note that jobs j_1 and j_2 have completed all of their operations. Thus, the updated list of schedulable jobs is as follows:

$$S = \{(j_3, O_2)\}$$

Table 5.5 Computation of EST for new processe

Job	Opn	Pj	Mj	MAT	JAT	EST
3	2	6	3	14	13	14

Schedule job j_3 is on machine M_3 at time 14. Processing of this operation will be completed at time 20. Then, the last operation of job j_3 is on machine M_2. EST value of job j_3 for last operation O_3 is shown in the following table.

Table 5.6 EST for operation O_3 of job j_3

Job	Opn	Pj	Mj	MAT	JAT	EST
3	3	5	2	7	20	20

Therefore, schedule job j_3 is on machine M_2 at time 20. Then, the operation will be completed at time 25. The complete Gantt chart for schedule is shown in the following fig 5.4.

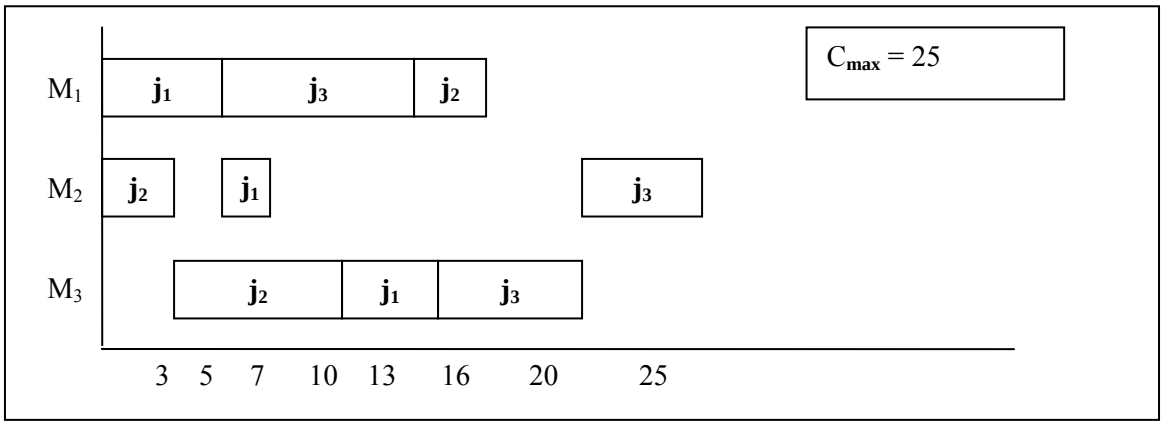


Figure 5.4 Gantt chart for complete schedule

The sequences on three machines using the EST heuristic are given in the following table 5.7.

Table 5.7 Sequences for all machines

Machine	Jobs Sequence
M_1	j_1, j_3, j_2
M_2	j_2, j_1, j_3
M_3	j_2, j_1, j_3

The value of C_{max} is 25

5.2 SHIFTING BOTTLENECK (SB) HEURISTIC

Several heuristics have been developed by several researchers to solve the job shop problem. Most of these heuristics have been reported by Conway, Maxwell, and Miller (1967), Baker (1974), Rinnooy (1976), Bellman, Esogbue, and Nabeshima (1982), French (1982), Morton and Pentico (1993), and Pinedo (1995). However, this section will be devoted to the recently developed heuristic which is known as the Shifting Bottleneck (SB) algorithm. The SB algorithm was developed in 1988 by Adams, Balas, and Zawack. Then, in 1993 it was modified by Dauzere-Peres and Lasserre. The SB algorithm was extended by Balas, Lenstra, and Vazacopoulos (1995). The SB algorithm is this chapter for four reasons:

1) It is the only well-known heuristic that simulates the management of bottleneck machines in the job shop environment.
2) It is known to be superior among all heuristics that were used to solve the job shop problems.
3) The SB algorithm and genetic algorithms were combined in several implementations.
4) The results obtained by the SB algorithm have been used as a benchmark to test the performance of several genetic algorithms.

The SB algorithm was developed to solve the general sequencing problem, where the makespan was minimized. The idea of the SB algorithm was described by Adams, Balas, and Zawack (1988), who stated:

> ...We sequence the machines one at a time, consecutively. In order to do this, for each machine not yet sequenced we solve to optimality a one-machine scheduling problem that is a relaxation of the original problem, and use the outcome both to rank the machines and to sequence the machine with highest rank. Every time a new machine has been sequenced, we reoptimize the sequence of each previously sequenced machine that is susceptible to

improvement by again solving a one-machine problem. (Adams, Balas, and Zawack 1988; 393)

The above description of the SB algorithm can be re-stated as follows. The SB algorithm sequences machines sequentially one at a time. The machines that have not yet been sequenced are ignored, and the machines that have been sequenced have their sequences held fixed. At each step, the SB algorithm determines a bottleneck machine from the set of machines that have not yet been sequenced by performing two steps:

1. Solving a one-machine scheduling problem for each un-sequenced machine.
2. The machine that yielded the maximum makespan is selected to be the bottleneck machine from the set of machines that have not yet been sequenced.

Then, the associated sequence that was obtained by the one-machine scheduling problem is used to sequence the bottleneck machine chosen. Every time a bottleneck machine is sequenced, a re-optimization procedure for the set of machines that have been sequenced is performed. The re-optimization is performed by freeing up and re-sequencing each machine in turn with the sequences on the other machines held fixed.

To test the quality of solutions obtained by the SB algorithm, several small problems for which an optimal solution was known were solved by Adams, Balas, and Zawack. Also, this team solved large problems which had up to 500 operations and ten machines. From the results obtained, they found out that the SB algorithm was able to find the optimal solution in all the problems with ten machines and over 30 jobs. The SB algorithm found in five minutes the optimal solution to a difficult problem that was designed by Fisher and Thompson (1963). In comparison, the optimal solution had only recently been found with extensive effort. The SB algorithm determination of the bottleneck machine was stable and accurate when there were many more jobs than machines, and this situation led the SB algorithm to converge to the optimal solution.

Adams, Balas, and Zawack solved forty problems to compare their algorithm to ten dispatching rules. These dispatching rules were FCFS, late start time (LST), early finish time (EFT), late finish time (LFT), most immediate successor (MIS), first available (FA), SPT, LPT, RANDOM, and JST. For the forty problems, they did not report the results of each dispatching rule. However, for each problem, they reported the best solution obtained by one of the dispatching rules and compared it to the solution obtained by the SB algorithm. From the results reported, the SB algorithm dominated in 38 problems.

The shifting bottleneck heuristic is an efficient method to find C_{max} and L_{max} objectives in a job shop. It is an iterative method. At each iteration of the method, a

bottleneck machine is identified using 1 | rj | L_{max} methodology. A processing sequence of jobs on the machine is found so as to minimize L_{max}. The method works as follows:

Step 1

Initialization,

Let M_O = set of bottleneck machines.

Set M = { ϕ }. Construct a CPM network of the jobs to find C_{max}.

Set C_{max} = Maximum path length from "S" to "E" nodes in the graph.

Step 2

For each machine m not in set M_O, solve 1 | rj | L_{max} problem to find L_{max} and optimal sequence of the jobs on the Machine m. Find r_j and d_j of each job j to be processed on the machine m as follows:

r_j = Earliest start time of the job node (m,j) in Graph for the particular machine m.

d_j = Minimum latest start time among the nodes those succeeded by the job node (m, j)in graph for the particular machine m.

Step 3

Identify bottleneck machine;

Compare L_{max} of all machines. The machine with highest value of L_{max} is the bottleneck machine.

Update set M_O by adding the newly selected bottleneck machine.

Update C_{max} value by adding L_{max} of bottleneck to current value of C_{max}.

Update graph G by adding new arcs from the sequence found for bottleneck machine.

If all machines are in set M_O stop, otherwise go to step (2) for next iteration.

5.2 Example

Consider J_4 || C_{max} instance with 4-machine and 3-job problem. Then, solve it using the SB heuristic.

jobs(j)	machine sequence	process times
1	1, 2, 3	p_{11}=10, p_{21}=8, p_{31}=4
2	2, 1, 4, 3	p_{22}=8, p_{12}=3, p_{42}=5, p_{32}=6
3	1, 2, 4	p_{13}=4,p_{23}=7, p_{43}=3

Solution:

Iteration 1: $M_O = \phi$, CPM network graph is as follows.

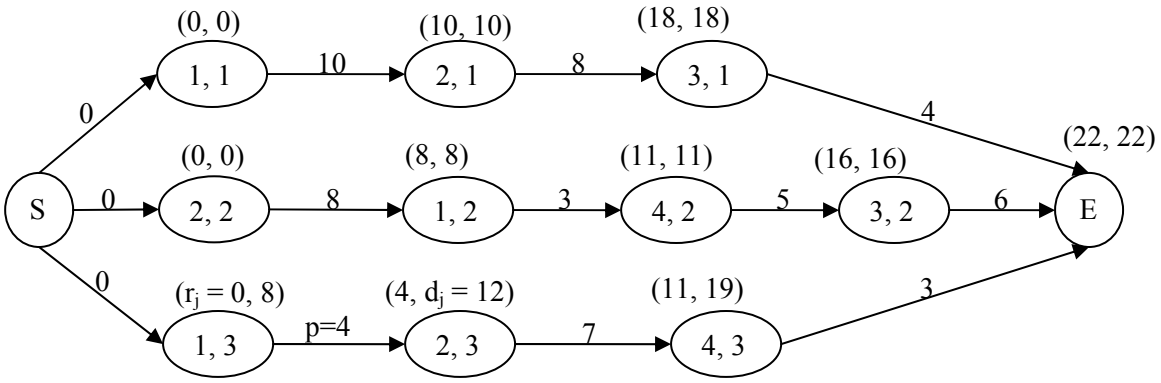


Figure 5.5 Disjunctive arc graphs of the jobs

C_{max} = Longest path in the Graph
C_{max} = 22 for job 1 and 2.

Machines required for the first operations for the three jobs are; M_1 and M_2. Thus, we will solve $1 \mid r_j \mid L_{max}$ for all the four machines (M_1, M_2, M_3, M_4) as follows:

Machine 1

$1 \mid r_j \mid L_{max}$ Problem

Table 5.8 $1 \mid r_j \mid L_{max}$ problem for machine M_1

Job (j)	**1**	**2**	**3**
Operation	(1,1)	(1,2)	(1,3)
p_{1j}	10	3	4
r_{1j}	0	8	0
d_{1j}	10	11	12

In order to minimize L_{max} for the problem above the following sequence is obtained: $\{j_1, j_2, j_3\}$.

The schedule for the jobs on machine M_1 using EDD (or B&B) is given in the following table 5.9

Table 5.9 Schedule for minimization of L_{max} for machine M_1

Job (j)	P_{1j}	r_{1j}	S_{1j}	C_{1j}	d_{1j}	L_{1j}
1	10	0	0	10	10	0
2	3	8	10	13	11	2
3	4	0	13	17	12	5

Thus, the maximum lateness on machine M_1 ($L_{max}(1)$) is equal to 5.

Machine 2

$1 \mid r_j \mid L_{max}$ Problem

Table 5.10 $1 \mid r_j \mid L_{max}$ problem for machine M_2

Job (j)	1	2	3
Operation	(2,1)	(2,2)	(2,3)
p_{2j}	8	8	7
r_{2j}	10	0	4
d_{2j}	18	8	19

In order to minimize L_{max}, for the problem above, the following sequence is resulted: $\{j_2, j_3, j_1\}$.

The schedule for the jobs on machine M_2 is given in the following table 5.11

Table 5.11 Schedule for minimization of L_{max} for machine M_2

Job (j)	P_{2j}	r_{2j}	S_{2j}	C_{2j}	d_{2j}	L_{2j}
2	8	0	0	8	8	0
3	7	4	8	15	19	-4
1	8	10	15	23	18	5

Using EDD (2-1-3) the $L_{max} = 6$, then why did sequence 2-3-1 is selected. The logic behind this is use EDD with the ready times. Therefore, the maximum lateness on machine M_2 ($L_{max}(2)$) is equal to 5.

Machine 3

1 | r_j | L_{max} Problem

Table 5.12 1 | r_j | L_{max} problem for machine M_3

Job (j)	**1**	**2**	**3**
Operation	(3,1)	(3,2)	(3,3)
p_{3j}	4	6	-
r_{3j}	18	16	-
d_{3j}	22	22	-

In order to minimize L_{max}, for the problem above, the following sequence is resulted: $\{j_2, j_1\}$.

The schedule for the jobs on machine M_3 is given in the following table 5.13

Table 5.13 Schedule for minimization of L_{max} for machine M_3

Job (j)	**p_{3j}**	**r_{3j}**	**S_{3j}**	**C_{3j}**	**d_{3j}**	**L_{3j}**
2	6	16	16	22	22	0
1	4	18	22	26	22	4

Hence, the maximum lateness on machine M_3 ($L_{max}(3)$) is equal to 4.

Machine 4

1 | r_j | L_{max} Problem

Table 5.14 1 | r_j | L_{max} problem for machine M_4

job (j)	**1**	**2**	**3**
Operation	(4,1)	(4,2)	(4,3)
p_{4j}	-	5	3
r_{4j}	-	11	11
d_{4j}	-	16	22

The minimum L_{max} for the above data is obtained using the following sequence: $\{j_2, j_3\}$.

The schedule for the jobs on machine M_4 is given in the following table 5.13

Table 5.15 Schedule for minimization of L_{max} for machine M_4

Job (j)	p_{4j}	r_{4j}	S_{4j}	C_{4j}	d_{4j}	L_{4j}
2	5	11	11	16	16	0
3	3	11	16	19	22	-3

Thus, the maximum lateness on machine M_4 (L_{max} (4)) is equal to 0.

Therefore, no more machines to consider which means the bottleneck machine can be identified as follows:

Bottleneck Machine is the machine with max {$L_{max}(1)$, $L_{max}(2)$, $L_{max}(3)$, $L_{max}(4)$ }= max {5 ,5, 4 , 0 }. Since, there is a tie between M_1 and M_2, let us pick M_1 as bottleneck machine with the jobs sequence on M_1 is as follows: {j_1, j_2, j_3}. This is the end of iteration 1.

Iteration 2: Update M_O = {M_1} and then updated Graph,

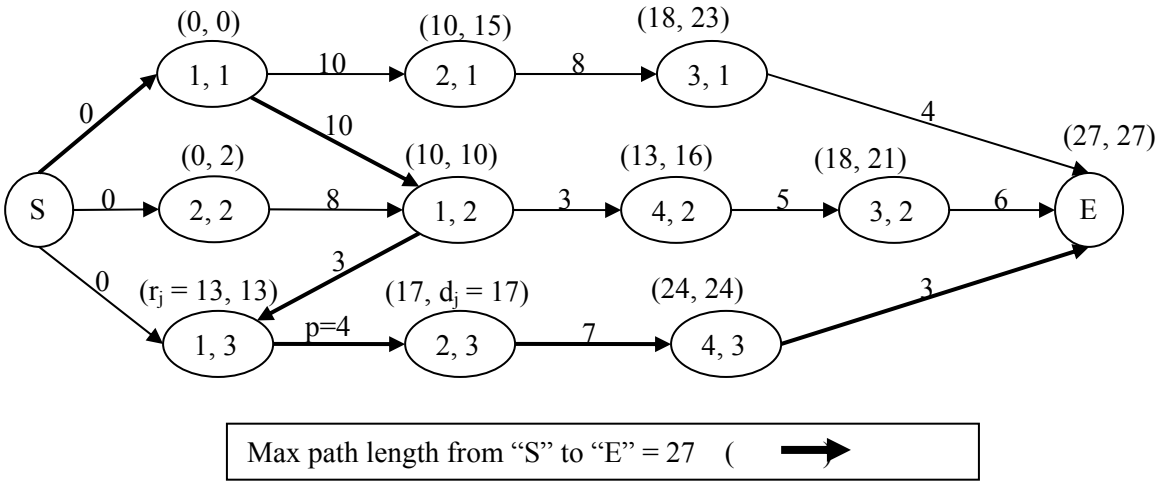


Figure 5.6 Updated disjunctive arc graphs of the jobs

Updating C_{max} value; $C'_{max} = C_{max} + l_{max} = 22 + 5 = 27$

Calculate L_{max} for machines M_2, M_3 and M_4 using 1 | r_j | L_{max}.

Machine 2

1 | r_j | L_{max} Problem

Table 5.14 1 | r_j | L_{max} problem for machine M_2

Job (j)	**1**	**2**	**3**
Operation	(2,1)	(2,2)	(2,3)
p_{2j}	8	8	7
r_{2j}	10	0	17
d_{2j}	23	10	24

Minimizing the L_{max} will result in the following sequence: $\{j_2, j_1, j_3\}$.
The schedule for the three jobs on machine M_2 is given in following table 5.15

Table 5.15 Schedule for minimization of L_{max} for machine M_2

Job (j)	**p_{2j}**	**r_{2j}**	**S_{2j}**	**C_{2j}**	**d_{2j}**	**L_{2j}**
2	8	0	0	8	10	-2
1	8	10	10	18	23	-5
3	7	17	18	25	24	1

Hence, the maximum lateness on machine M_2; $L_{max}(2) = 1$

Machine 3

1 | r_j | L_{max} Problem

Table 5.16 1 | r_j | L_{max} problem for machine M_3

Job (j)	**1**	**2**	**3**
Operation	(3,1)	(3,2)	(3,3)
p_{3j}	4	6	-
r_{3j}	18	18	-
d_{3j}	27	27	-

The following Sequence: $\{j_1, j_2\}$ will minimize the L_{max}.
The Schedule for the jobs on machine M_3 is given in the following table 5.17

Table 5.17 Schedule for minimization of L_{max} for machine M_3

Job (j)	p_{3j}	r_{3j}	S_{3j}	C_{3j}	d_{3j}	L_{3j}
1	4	18	18	22	27	-5
2	6	18	22	28	27	1

Thus, the maximum lateness on machine M_3; $L_{max}(3) = 1$

Machine 4

$1 \mid r_j \mid L_{max}$ Problem

Table 5.18 $1 \mid r_j \mid L_{max}$ problem for machine M_4

Job (j)	1	2	3
Operation	(4,1)	(4,2)	(4,3)
p_{4j}	-	5	3
r_{4j}	-	13	24
d_{4j}	-	21	27

Solving the above problem to minimize L_{max} will result in the following sequence: {j2, j3}.

The schedule for the jobs on machine M_4 is given in following table 5.19

Table 5.19 Schedule for minimization of L_{max} for machine M_4

Job (j)	p_{4j}	r_{4j}	S_{4j}	C_{4j}	d_{4j}	L_{4j}
2	5	13	13	18	21	-3
3	3	24	24	27	27	0

Therefore, the maximum lateness on machine M_4; $L_{max}(4) = 0$

Thus, no more machines to consider which means the bottleneck machine can be identified as follows:

Bottleneck Machine is the machine with max $\{L_{max}(2), L_{max}(3), L_{max}(4)\}$= max {1 ,1, 0 }. Since, there is a tie between M_2 and M_3, let us pick M_2 as the second bottleneck machine with the jobs sequence on M_2 is as follows: $\{j_2, j_1, j_3\}$. This is the end of iteration 2.

Iteration 3: Update $M_O = \{M_1, M_2\}$ and updated Graph.

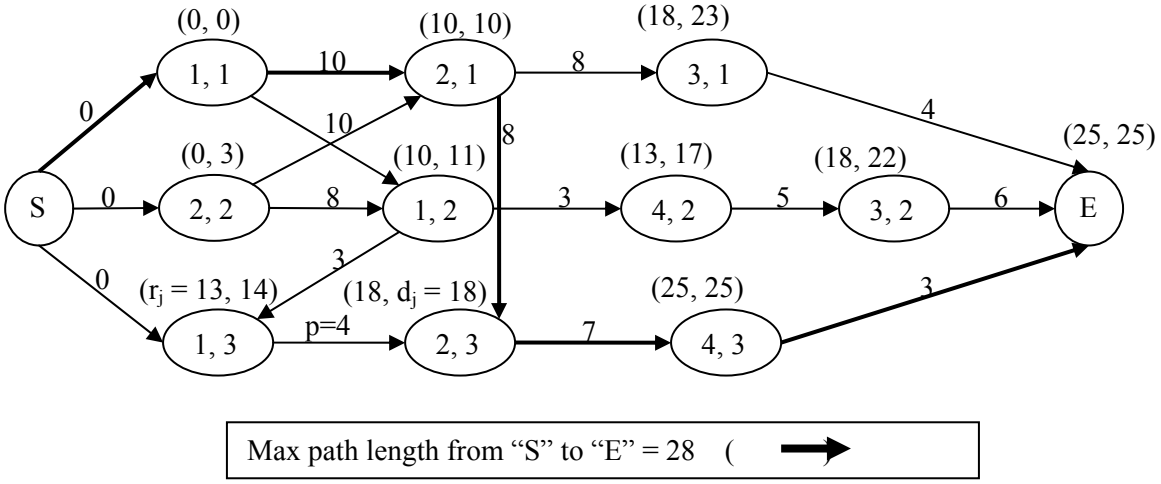


Figure 5.7 Updated disjunctive arc graphs of the jobs

Updating C_{max} value; $C''_{max} = C'_{max} + L_{max}(2) = 27 + 1 = 28$

Calculate L_{max} for machines M_3 and M_4 using $1 \mid r_j \mid L_{max}$.

Machine 3

$1 \mid r_j \mid L_{max}$ Problem

Table 5.20 $1 \mid r_j \mid L_{max}$ problem for machine M_3

Job (j)	**1**	**2**	**3**
operation	(3,1)	(3,2)	(3,3)
p_{3j}	4	6	-
r_{3j}	18	18	-
d_{3j}	28	28	-

Solving the above problem to minimize L_{max} will result in the following sequence: $\{j_1, j_2\}$.

The schedule of jobs on machine M_3 is given in following table 5.21

Table 5.21 Schedule for minimization of L_{max} for machine M_3

Job (j)	**p_{3j}**	**r_{3j}**	**S_{3j}**	**C_{3j}**	**d_{3j}**	**L_{3j}**
1	4	18	18	22	28	-6
2	6	18	22	28	28	0

Hence, the maximum lateness on machine M_3; $L_{max}(3) = 0$

Machine 4

1 | rj | L_{max} Problem

Table 5.22 1 | r_j | L_{max} problem for machine M_4

Job (j)	**1**	**2**	**3**
operation	(4,1)	(4,2)	(4,3)
p_{4j}	-	5	3
r_{4j}	-	13	25
d_{4j}	-	22	28

Solving the above problem to minimize L_{max} will result in the following sequence: $\{j_2, j_3\}$.

The schedule of jobs on machine M_4 is given in following table 5.23

Table 5.21 Schedule for minimization of L_{max} for machine M_4

Job (j)	**p_{4j}**	**r_{4j}**	**S_{4j}**	**C_{4j}**	**d_{4j}**	**L_{4j}**
2	5	13	13	18	22	-4
3	3	25	25	28	28	0

Hence, the maximum lateness on machine M_4; $L_{max}(4) = 0$

Since, Maximum Lateness for M_3 and M_4 is zero. Optimal C_{max} value does not change. Sequence of jobs on machines is as follows:

Machine	Job Sequence	Machine	Job Sequence
M_1	$\{j_1, j_2, j_3\}$	M_2	$\{j_2, j_1, j_3\}$
M_3	$\{j_1, j_2\}$	M_4	$\{j_2, j_3\}$

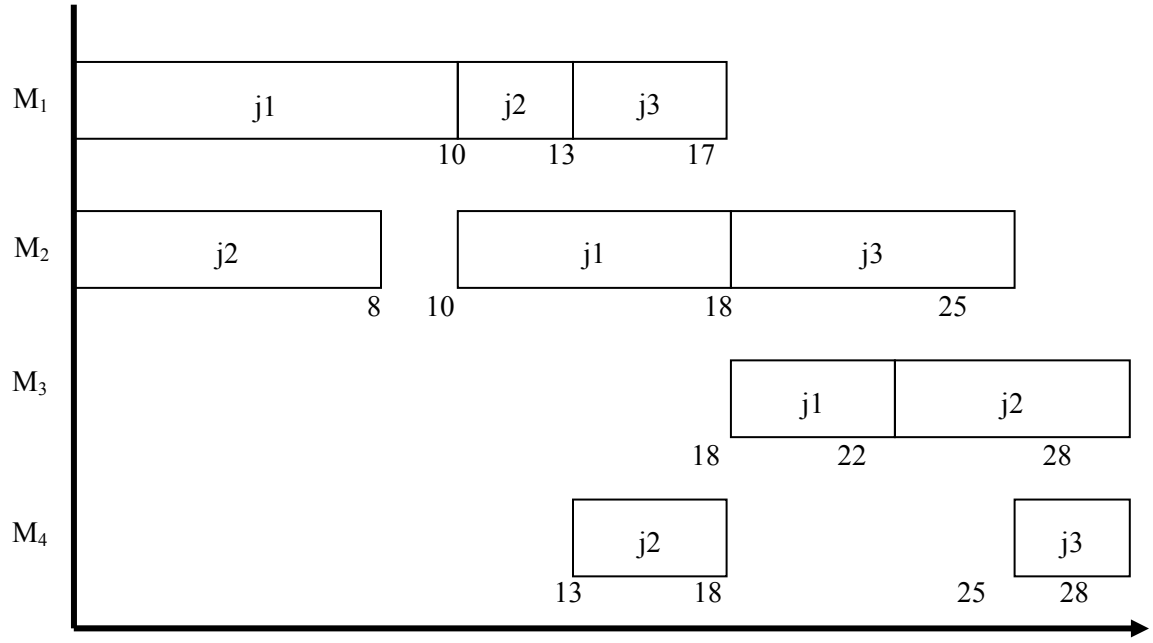


Figure 5.8 Gantt chart for final schedule

EXERCISES

5.1 Find a feasible active schedule for the job shop problem using the following data:

Job	Jobs routing	Processing times
1	1 - 2 - 3	5 - 10 - 12
2	1 - 3 - 2	4 - 3 - 8
3	3 - 2 - 1	9 - 6 - 7
4	2 - 3 - 1	7 - 5 - 11

Using a common due date of 27 for all jobs, determine,

A. Makespan
B. Total flow time
C. Total Tardiness
D. Total Lateness

5.2 Consider $J_4 \mid \mid C_{max}$ problem with the following data:

Job	Jobs routing	Processing times
1	1 - 2 - 4	6 - 8 - 5
2	2- 3 - 4	4 - 4 - 3
3	4 - 2 - 1	8 - 6 - 4
4	2 - 3 - 4	5 - 10 - 15

Find the best makespan using the shifting bottleneck algorithm. Then, compute total flow time, total waiting time, and the utilization.

5.3 Consider $J_4 \mid \mid C_{max}$ problem using the following data:

Job	Jobs routing	Processing times
1	1 - 2 - 3	5 - 10 - 12
2	1 - 3 - 2	4 - 3 - 8
3	3 - 2 - 1	9 - 6 - 7
4	2 - 3 - 1	7 - 5 - 11

Find the best makespan using the shifting bottleneck algorithm

5.4 Using the SPT rule, develop a feasible schedule that minimizes makespan for the following jobs in a job shop problem:

Job	Jobs routing	Processing times
1	2 - 4 - 5	7 - 11 - 2
2	5 - 3	3 - 6
3	4 - 1 - 5 - 2	15 - 3 - 4 - 6
4	3 - 2 - 1 - 4 - 5	3 - 1 - 6 - 4 - 2
5	5 - 1 - 2 - 3 - 4	3 - 7 - 6 - 1 - 3
6	3 - 1 - 5 - 4	4 - 6 - 5 - 3

5.5 Using the EDD rule, develop a feasible schedule that minimizes the total lateness penalty for the following jobs in a job shop problem $J_5 \mid \mid \Sigma x_j$ problem:

Job	Job routing	Processing times	Due date	Late Penalty (PT)	Early Penalty (PE)
1	2-4-5	7-11-2	33	11	4
2	5-3	3-6	11	4	1
3	4-1-5-2	15-3-4-6	32	1	0
4	3-2-1-4-5	3-1-6-4-2	47	2	2
5	5-1-2-3-4	3-7-6-1-3	27	1	0
6	3-1-5-4	4-6-5-3	28	4	3

Where the total lateness penalty $\sum_{j=1}^{6} X_j$ is computed as follows:

$$x_j = \begin{cases} (C_j - d_j) * PT_j & if \ldots C_j - d_j > 0 \\ (d_j - C_j) * PE & if \ldots C_j - d_j \le 0_j \end{cases} \cdots j = 1 \cdots 6$$

Where:

PT_j is the late penalty for job j
PE_j is the early penalty for job j.

Then, construct the Gantt chart to show your solution and then compute the total lateness penalty.

5.6 Consider $J_4 \mid\mid C_{max}$ problem and use the most work remaining (MWKR) dispatching rule to find the best makespan using the following data:

Job	Job routing	Processing times
1	1 - 2 - 3 - 4	6 - 8 - 13 - 5
2	1 - 2 - 3 - 4	4 - 1 - 4 - 3
3	4 - 2 - 1 - 3	3 - 8 - 6 - 4
4	2 - 1 - 3 - 4	5 - 10 - 15 - 4
5	1 - 2 - 4 - 3	3 - 4 - 6 - 4
6	3 - 1 - 2 - 4	4 - 2 - 4 - 5

5.7 Consider the following instance of the $J_3 \mid\mid C_{max}$ problem.

Job	Machine Sequence	Process Times
1	1-2-3	p11=5, p21=4, p31=6
2	2-1	p22=5,p_{12}=7
3	2-3	p23 =5, p_{33} =5

Apply the Shifting bottleneck heuristic for this instance and find C_{max}. Also, draw Gantt chart for your solution.

5.8 Consider $J_3 \mid \mid C_{max}$ with the following data:

Job	M/C routing	Processing times	Due date
1	1 - 2 - 3	$P_{11} = 4, P_{12} = 3, P_{13} = 2$	9
2	2 - 1 - 3	$P_{22} = 4, P_{21} = 1, P_{23} = 4$	9
3	3 - 2 - 1	$P_{33} = 3, P_{32} = 2, P_{31} = 4$	9
4	2 - 3 - 1	$P_{42} = 3, P_{43} = 1, P_{41} = 5$	9

Use the data above to solve the job shop problem according to SPT, Slack time, and CR rules. In each case, show your feasible solution on Gantt chart and compute C_{max}, T_{max} $\overline{\mathbf{F}}$, and $\overline{\mathbf{T}}$. According to our objective function which schedule is better?

6

Project Management and Scheduling

CHAPTER CONTENTS

Project management is a set of principles, methods, and techniques that people use to effectively plan and control project work. It establishes a sound basis for effective planning, scheduling, resourcing, decision making, controlling, and re-planning. The objective of project management is to ensure that projects meet agreed goals of time, cost, and scope. Today, modern project management has emerged as a premier solution in business operations. Large and small organizations recognize that a structured approach to planning and controlling projects is a necessary core competency for success.

6.1 INTRODUCTION

Project work and traditional functional work differ in significant ways. Functional work is routine, ongoing work. A manager is assigned to the specific function and provides worker training and supervision. In contrast, a project is "*a temporary endeavor undertaken to create a unique product or service*." A project manager is responsible for the approved objectives of a project, such as budget, schedule, and scope.

The need for project management is apparent in the world today as speed, quality, and cost control are becoming increasingly important. Implementing a project management system requires a long-term commitment and management support. It is important to understand how your organization is structured so you can decide how to fit project management techniques into it. Organizational structures typically span the spectrum from functional to project, with a variety of matrix structures in between. A functional organization is a hierarchy in which people are grouped into functional divisions, such as marketing or production. Each employee has one clear superior. In a project organization, projects are centralized in a separate division of skilled project managers that serves the project management needs of all divisions of the company. This is often referred to as a project office. Matrix organizations are a blend of functional and project organizations. A weak matrix has many of the characteristics of a functional organization and the project manager role is more that of a coordinator or expediter with limited authority. A strong matrix organization has many of the characteristics of a project organization, with a full-time project manager who has significant authority and a project administrative staff. In a matrix organization, the project team has a dual reporting role to a project manager, coordinator, or expediter (who provides project management skills) and a functional manager (who provides technical and functional skills). In a strong matrix organizational structure, the project manager has more power than the functional manager. In a weak matrix structure, the balance of power leans toward the functional manager.

It is important to set up a formal planning and control system that is flexible enough to operate in the real world, but still rigorous enough to provide control. A

project management system must allow for adjustments to the plan as needed throughout the project's life. The system helps you define the problem or opportunity, establish project objectives, develop the project plan, begin project work, monitor and control the work, and then close the project.

6.2 PLANNING THE PROJECT

Some people put a minimum of effort into planning. They argue that since things invariably change during the life of the project, it is a waste of effort to make extensive up-front plans. The average organization spends only 5 percent of the total project effort on planning. More successful organizations spend up to 45 percent. A good rule of thumb is to spend 25 percent of the project effort in concept and development and 75 percent in implementation and termination.

Although it is true that factors might be introduced during the life of the project that necessitate minor or major adjustments to the plan, it is still important to have a solid plan in place. Without one, a project becomes even more chaotic in the face of change. If plans are made using project management software, it is easy to make adjustments to the plan as needed.

In an ideal world, a project would be planned and approved, and then work would start and be completed according to the plan. In actual practice, however, you might have to adjust the plan throughout the life of the project. Therefore, any good planning and control system must be flexible enough to operate in the real world, and yet be rigorous enough to provide control.

Some projects are managed in pieces. Because of time constraints or other factors, the project manager might have to develop a plan for only part of the project, get it approved, and begin that portion while other parts of the project are still in the planning stage.

Often, planning continues to some extent throughout the life of the project. Recognizing this reality, the successful project manager establishes a project management system that allows for adjustments to the plan as needed. Figure 6.1 shows how a project management system allows a project to react to changing conditions.

The key steps in planning are as follows:

- *Define the problem or opportunity* that this project addresses.
- *Establish project objectives* in terms of time, cost, and scope.
- *Perform project reviews* to ensure the project is needed, feasible, and practical.
- *Define the work* (*activities*) that must be done to complete the project.
- *Estimate the cost and time* needed to accomplish each activity.

- *Sequence the activities* into a logical order, considering the dependencies between activities.
- *Calculate the critical path* to determine the longest sequence of activities.
- *Schedule the activities* by applying calendar dates.
- *Prepare resource plans* by assigning specific personnel and equipment to each activity.
- *Prepare budget plans* to determine what funds are needed at what times.
- *Plan for risk* to be ready to respond to events that may effect the project for better or worse.
- *Get approvals and compile a formal project plan.*

6.3 EXECUTING THE PROJECT

When all plans are in place, approved, and communicated to project personnel, project work can begin.

6.3.1 Monitor

As project work progresses, the project manager gather status information and compare it to the plan to determine variances. Deviations from the plan are then analyzed to determine if corrective action should be taken.

6.3.2 Control

When necessary, the project manager takes corrective action to get the project back on track. Some deviations might require re-sequencing activities, rescheduling, re-budgeting, or reallocating resources. Larger deviations can necessitate renegotiating the basic project objectives of cost, time, and scope. In some cases, the situation might be serious enough to warrant readdressing the problem or opportunity to determine if it has been identified correctly and if the organization has the resources, expertise, and commitment necessary to handle it.

Planning, monitoring, and controlling are not one-time events. They continue throughout the life of the project to refine and adjust to current conditions (see Figure 6.1).

6.3.3 Closing

A good project management methodology includes formal steps to close the project. The purpose of project closure is to verify that all work has been accomplished as agreed and that the client or customer accepts the final product.

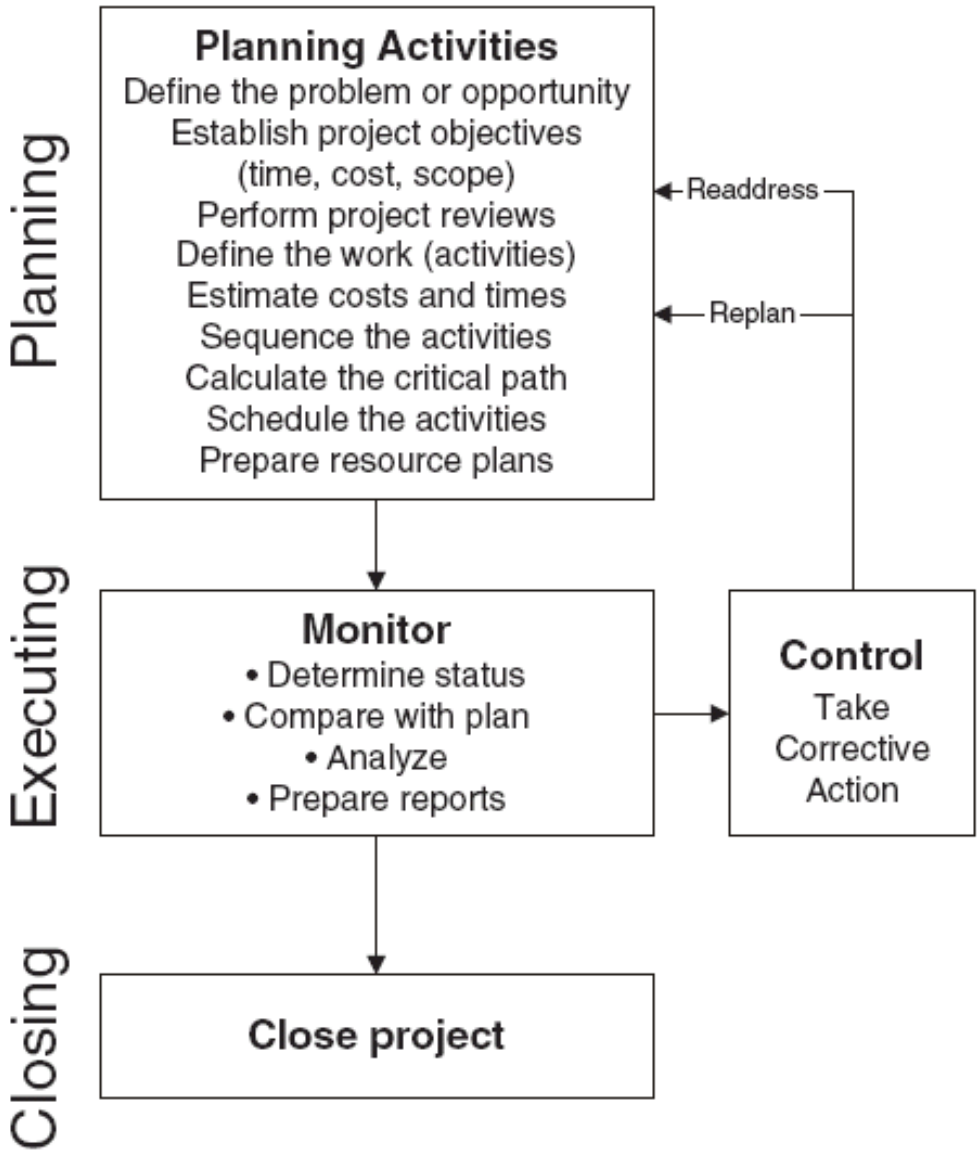


Figure 6.1 Project Management System

6.4 PROJECT SCHEDULING

In terms of classical scheduling theory, project scheduling is concerned with execution of m jobs using infinite number of machines with precedence constraints. However, scope of project scheduling is highly expanded and applies to vast variety of activities in real world. It includes every sphere of discipline not merely manufacturing. The term project connotes a synchronized and well coordinated effort by a multitude of inputs (men, machines, money) to achieve a well defined target in a specific amount of time while living within the resources. Building of a hospital, construction of super highway, installation of petroleum refining plant, laying of a gas

pipe line from one continent to another continent and development of computer network on a university campus are some examples of term *project.* Typically every project is unique as it comprises distinct set of activities (jobs) with specific time and resource constraints. The tasks inter-relationship plays a decisive role in project scheduling. If resources constraint is not an issue, it is termed as unconstrained project scheduling. In this scheduling scenario, scheduling decision problem requires these two basic parameters as input;

- Processing times (durations) of the activities
- Precedence relationship of the activities in the project

However, activities do require resources for their execution. Often the same resources are required by different activities at the same time resulting in resource-use conflict situation. Scheduling methodology has to be modified in such scenarios. A class of problems called Resource-constrained scheduling has emerged over years to tackle these situations. CPM and PERT has gained wide spread use to solve unconstrained project scheduling problems. In this chapter, we present these two methodologies (CPM & PERT) to deal with unconstrained project scheduling. In next chapter, detailed account of Resource-constrained scheduling will be presented.

6.5 CRITICAL PATH METHOD (CPM)

Critical Path Method (CPM) is a project management technique, which has been created out of the need of industrial and military establishments to plan, schedule and control complex projects. CPM was the discovery of M. R. Walker of E.I.Du Pont de Nemours & Co. and J. E. Kelly of Remington Rand, circa 1957. The computation was designed for the UNIVAC-I computer. The first test was made in 1958, when CPM was applied to the construction of a new chemical plant. In March 1959, the method was applied to maintenance shut-down at the Du Pont works in Louisville, Kentucky. CPM helped the company to reduce unproductive time from 125 to 93 hours.

CPM provides an integrated frame work for planning, scheduling and control of project management. The scheduling of a project includes answers to important questions, like;

- How long will the entire project take to be completed? What are the risks involved?
- Which are the critical activities or tasks in the project which could delay the entire project if they were not completed on time?
- Is the project on schedule, behind schedule or ahead of schedule?
- If the project has to be finished earlier than planned, what is the best way to do this at the least cost?

A project is defined by a set of distinguishable, indivisible and distinct set of actions called activities. The activities have dependency relationship among themselves. Typically, an activity will have some activities preceding it and some activities following it. Then each activity will require time for its execution and incur costs. In addition, an activity would use specific types of resources (man power, equipment) for its completion.

Five useful questions to ask when collecting data about activities are;

- Is this a Start Activity?
- Is this a Finish Activity?
- What Activity Precedes this?
- What Activity Follows this?
- What Activity is Concurrent with this?

Some activities are serially linked. The second activity can begin only after the first activity is completed. In certain cases activities are concurrent, because they are independent of each other and can start simultaneously. This is especially the case in organizations which have supervisory resources so that work can be delegated to various departments which will be responsible for the activities and their completion as planned.

The information collected about activities for project planning and scheduling is well depicted by constructing precedence network diagram. The network diagram uses nodes and arcs to portray information about the activities. Two types of precedence network diagrams are constructed for presenting project activities data.

Activities-on-arcs (AOA) network diagrams use directed arcs to present an activity. Nodes represent start and terminal occurrence of various project activities. Besides representing activities, the arcs also contain information about precedence relationships and duration of activities.

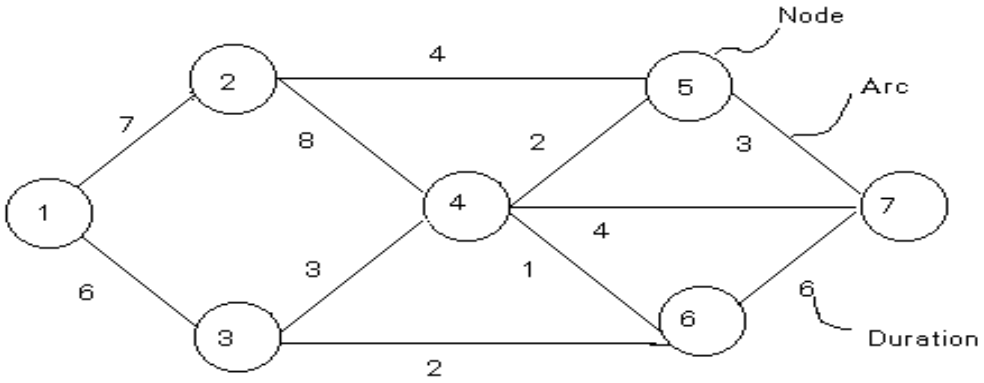


Figure 6.2 Activity-On-Arc Precedence Network Diagram

The precedence network diagram in Figure 6.2 presents 11-activity AOA project network with activities and their attributes as provided in Table 6.1

Table 6.1 Project network data in AOA and AON format

Sr. No.	Activity	Duration	Immediate Predecessor
1	(1, 2)	7	--
2	(1 , 3)	6	--
3	(2, 4)	8	(1, 2)
4	(3, 4)	3	(1, 3)
5	(2, 5)	4	(1, 2)
6	(3, 6)	2	(1, 3)
7	(4, 5)	2	(2, 4), (3, 4)
8	(4 , 7)	4	(2, 4), (3, 4)
9	(4, 6)	1	(2, 4), (3, 4)
10	(5, 7)	3	(2, 5), (4, 5)
11	(6, 7)	6	(3, 6), (4, 6)

Activity-on-Node (AON) precedence network diagram is the second way of presenting project data. The nodes present activities. Directed arcs present precedence relationship which join the nodes. The duration of activities is written inside or above the node. The data in Table is converted to Activity-on-node (AON) network format from activity-on-arc (AOA) network format and is shown in Table 6.2.

Table 6.2 Project network data in AOA and AON format

AOA Format	AON Format	Duration	Immediate Predecessor For AOA Format	Immediate Predecessor For AON Format
(1 ,2)	A	7	--	--
(1 , 3)	B	6	--	--
(2 , 5)	C	4	(1 , 2)	A
(2 , 4)	D	8	(1 , 2)	A
(3 , 4)	E	3	(1 , 3)	B
(3 , 6)	F	2	(1 , 3)	B
(4 , 5)	G	2	(2 , 4) , (3 , 4)	D , E
(4 , 7)	H	4	(2 , 4) , (3 , 4)	D , E

AOA Format	AON Format	Duration	Immediate Predecessor For AOA Format	Immediate Predecessor For AON Format
(4 , 6)	I	1	(2 , 4) , (3 , 4)	D , E
(5 , 7)	J	3	(2 , 5) , (4 , 5)	C , G
(6 , 7)	K	6	(3 , 6) , (4 , 6)	F , I

The precedence network diagram for the data in Table 6.2 using AON format is shown in Figure 6.2. Note two extra nodes have been added in the diagram. Both start node and end nodes are dummy nodes and, present starting as well as terminating activities of a project.

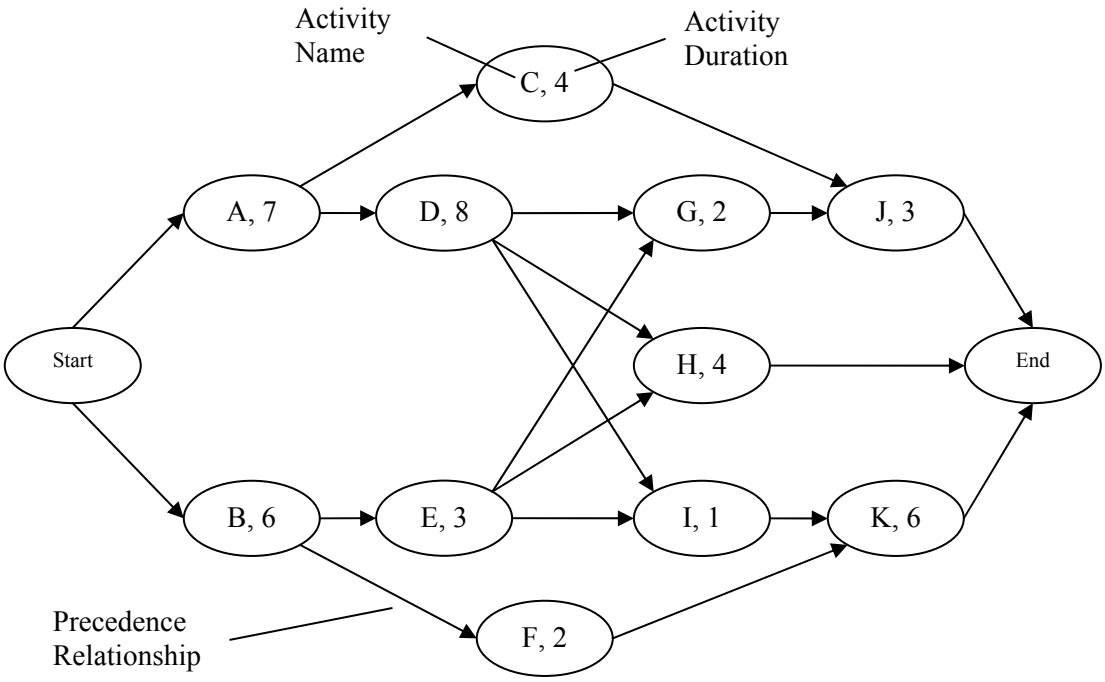


Figure 6.2 Activity-on-node (AON) precedence network diagram

After setting up the network, CPM technique finds the longest path through the activity network. The longest path comprises a set of activities which are designated as "critical" activities. These are the activities which must start and finish on exact dates and time, because the entire project completion is dependent upon them. CPM technique identifies these activities. When execution of project starts, these activities may be assigned to responsible persons. Management resources could be optimally used by concentrating on the few activities which determine the fate of the entire project.

Non-critical activities can be re-planned, rescheduled. Resources for these activities can be reallocated flexibly, without affecting the whole project.

6.5.1 CPM Calculations

Critical path method (CPM) minimizes the project completion time (C_{max}) of the project by finding longest path from source to sink node in the precedence network diagram. Let's define mathematical notations for activity-on-arc (AOA) precedence network as follows;

$p_{i,j}$ = Duration for activity (i, j)

E_i = Earliest occurrence time for event i

C_i = Latest occurrence time for event i

$S'_{i,j}$ = Earliest start time for activity (i, j)

$C'_{i,j}$ = Earliest completion time for activity (i, j)

$S''_{i,j}$ = Latest start time for activity (i, j)

$C''_{i,j}$ = Latest completion time for activity (i, j)

ψ_t = Total slack (float) for activity (i, j)

ψ_f = Free slack for activity (i, j)

The computations are carried out in two stages. First stage comprises forward pass calculations to compute earliest event times. The steps are as follows:

1. Set the start time of the initial event to zero.
2. Start each activity as soon as its predecessor events occur.
3. Calculate early event times as the maximum of the earliest completion time of activities terminating at the event.

In mathematical terms;

$$E_1 \equiv 0$$

And, $$E_j = \max \left\{ E_{i_1} + p_{i_1,j}, E_{i_2} + p_{i_2,j}, ..., E_{i_n} + t_{i_n,j} \right\}$$

Where,

$i_1, i_2, ..., i_n$ indicate the preceding events of the n activities that terminate at event j.

Earliest start and completion time for an activity (i, j) is calculated from the following expressions.

$$S'_{i,j} = E_i$$

$$C'_{i,j} = E_i + p_{i,j}$$

Project completion time,

$$C_{max} = E_n$$

Second stage comprises backward pass calculations to compute latest occurrence of event. The steps are as follows:

1 Set the latest time for the terminal event equal to the earliest time for that event.
2 Start each activity at the latest time of its successor event less the duration of the activity.
3 Determine event times as the minimum of the latest start times of all activities emanating from the event.

Set, $L_n = E_n$

And,

$$L_i = \min(L_{j_1} - t_{i,j_1},\ L_{j_2} - t_{i,j_2}\ ,\ \ldots\ldots\ldots\ldots,\ L_{j_v} - t_{i,j_v}) \qquad i < n$$

Where;

$j_1, j_2, \ldots\ldots, j_v$ indicate the successor events of v activities that emanate from event i.

Latest start and completion time for an activity (i, j) is calculated from the following expressions.

$$C''_{i,j} = L_j$$

$$S''_{i,j} = L_j - p_{i,j}$$

Total slack time for activity (i, j) is computed from;

$$\psi_{t\,i,j} = L_j - E_i - p_{i,j}$$

Free slack is the amount of time that activity completion can be delayed without affecting the early start ($S'_{i,j}$) time for any other activity in the network.

$$\psi_{f\,i,j} = E_j - E_i - p_{i,j}$$

Critical path consists of all activities with zero slack on the network.

Example 6.1

The data for an 8-activity project is given in following table with activities, their durations and corresponding precedence relationships.

Activity (i , j)	Duration $p_{i,j}$	Immediate Predecessor
(1 , 2)	4	---
(2 , 4)	7	(1 , 2)
(2 , 3)	8	(1 , 2)
(2 , 5)	6	(1 , 2)
(4 , 6)	15	(2 , 3) , (2 , 4)
(3 , 5)	9	(2 , 3)
(5 , 6)	12	(2 , 5) , (3 , 5)
(6 , 7)	8	(4 , 6) , (5 , 6)

Draw activity-on-arc (AOA) precedence network diagram, find project completion time and identify critical path and find slack times.

Solution:

An Activity-on-arc (AOA) network uses arcs to represent activities and nodes to represent occurrence of events. The precedence network diagram is shown in Figure 6.3

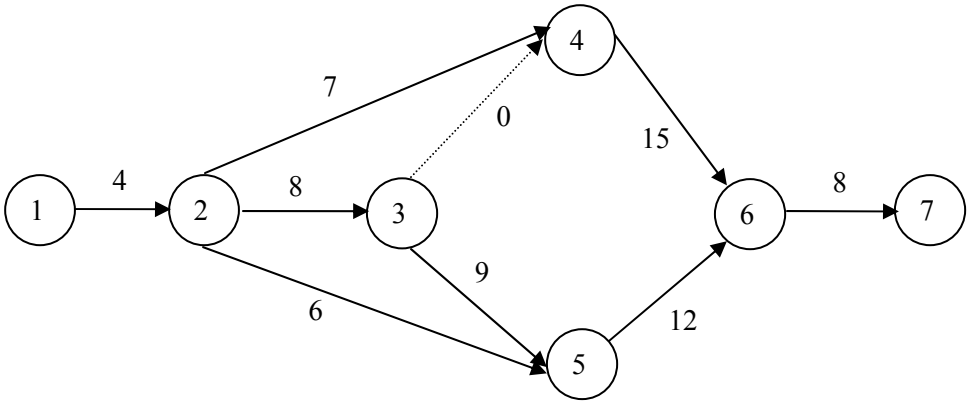


Figure 6.3 AOA Precedence network diagram

Early & late occurrence times calculation of events is shown in Table 6.4.

Table 6.4 Early and late occurrence times of Events

Event	E	L
1	0	0
2	4	4
3	12	12
4	12	18
5	21	21
6	33	33
7	41	41

Computations of various parameters pertaining to activities including early start time $S'_{i,j}$, early completion time $C'_{i,j}$, late start time $S''_{i,j}$, late completion time $C''_{i,j}$, total slack ψ_t and ψ_f are shown in Table 6.5.

Table 6.5 Calculated Parameters for Example 6.1

Activity	p (i,j)	$S'_{i,j}$	$C'_{i,j}$	$S''_{i,j}$	$C''_{i,j}$	ψ_t	ψ_f	Status
(1 ,2)	4	0	4	0	4	0	0	Critical
(2 , 4)	7	4	11	11	18	7	7	Slack
(2 , 3)	8	4	12	4	12	0	0	Critical
(2 , 5)	6	4	10	15	21	11	11	Slack
(4 , 6)	15	12	27	18	33	6	6	Slack
(3 , 5)	9	12	21	12	21	0	0	Critical
(5 , 6)	12	21	33	21	33	0	0	Critical
(6 , 7)	8	33	41	33	41	0	0	Critical

The schedule generated by CPM technique in Table 6.5 is shown on Gantt hart in Figure 6.4.

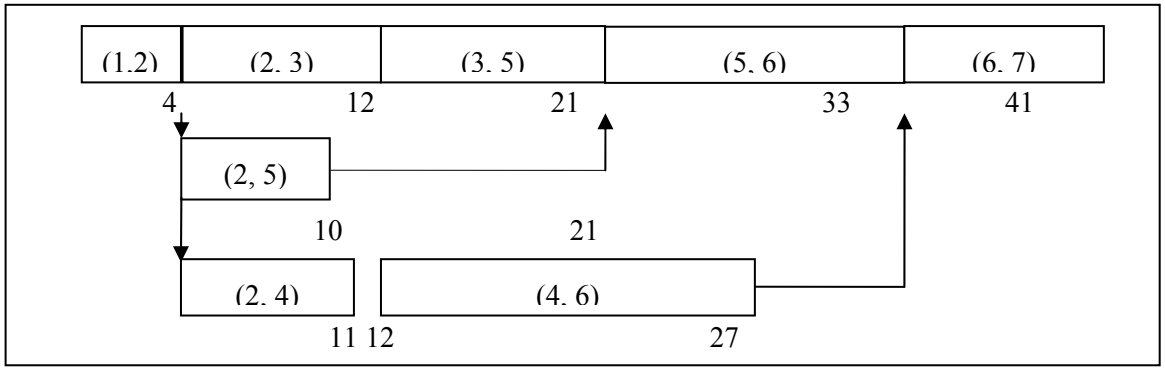


Figure 6.4 Gantt chart for Schedule

6.5.2 AON Network Computations

Activity-on-node (AON) network calculations are also carried out in two stages. Forward pass calculations are carried out in following steps.

1. Set early start time for source activity (node) equal to zero. Then, early completion time for source node will be equal to

$$C'_{source} = S'_{source} + p_{source} = 0 + p_{source} = p_{source}$$

2. For activity i, calculate early completion time by the expression;

$$C'_i = \max\left\{ C'_{j_1} , C'_{j_2} , ... , C'_{j_v} \right\} + p_i$$

Where,

Activities j_1, j_2, j_3,……, j_v are the predecessor activities of activity i and v < n as shown in Figure 6.5.

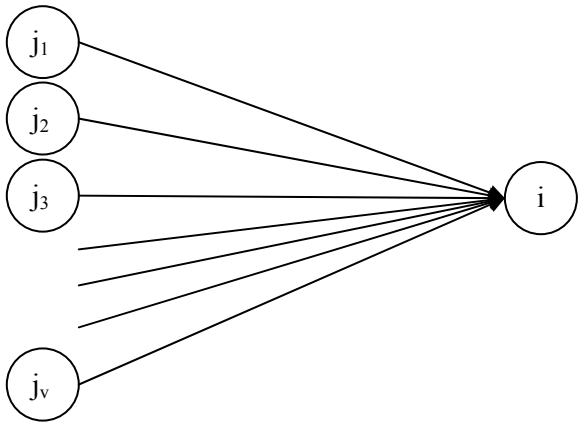


Figure 6.5 Set of predecessor activities for activity i

3. Then, project completion time; $C_{max} = C'_{sink}$

In second stage, backward pass calculations are carried out to find late start and finish times of activities.

1. Set late completion time for last (sink) activity equal to early completion time;

$$C''_n \equiv C'_n$$

2. For any activity k (k < n), find late completion time from the expression;

$$C''_k = \min\left\{ C''_{j_1} - p_{j_1} , C''_{j_2} - p_{j_2} , ... , C''_{j_v} - p_{j_v} \right\}$$

Where,

Activities j_1, j_2, j_3, …., j_v are the successor activities of activity k and v < n as shown in Figure 6.6.

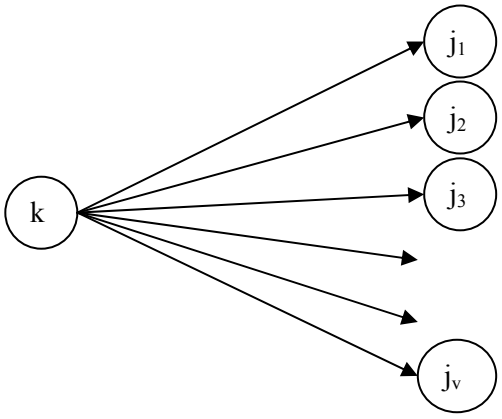


Figure 6.6 Set of successor activities for activity k

3. Activities on the critical path have zero slack time. Slack time for any activity j is found from the expression;

$$\psi_{t_j} = C_j'' - C_j'$$

Example 6.2

Change the data in Example 6.1 to represent activity-on-node (AON) frame work. Draw precedence network diagram and find critical path.

Solution:

The AON version of the data of Example 6.1 is shown in Table 6.6

Table 6.6 Activity-on-node (AON) data format

Activity Name	Activity (i , j)	Duration $p_{i,j}$	Immediate Predecessor
A	(1 , 2)	4	---
B	(2 , 4)	7	(1 , 2) → A
C	(2 , 3)	8	(1 , 2) → A
D	(2 , 5)	6	(1 , 2) → A
E	(4 , 6)	15	(2 , 3) , (2 , 4) → B , C
F	(3 , 5)	9	(2 , 3) → C
G	(5 , 6)	12	(2 , 5) , (3 , 5) → D , F
H	(6 , 7)	8	(4 , 6) , (5 , 6) → E , G

The precedence network diagram is developed and shown in Figure 6.7

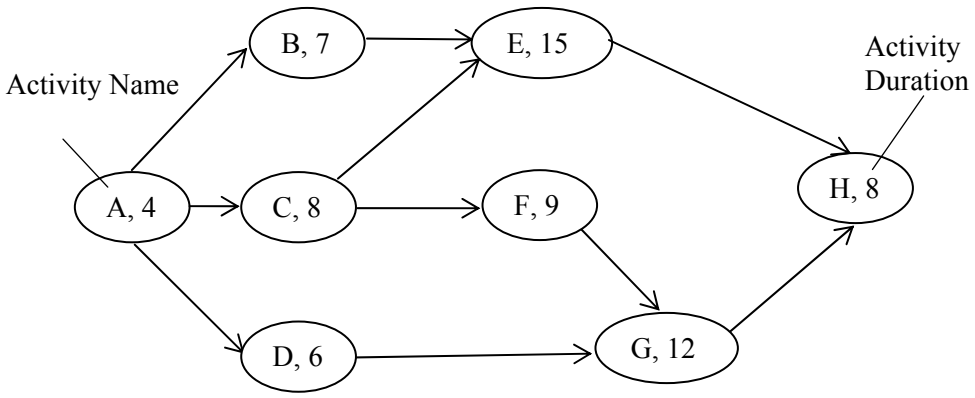


Figure 7.7 Activity-on-node (AON) Precedence network diagram

The time attributes of the activities including early start $S'_{i,j}$, early completion $C'_{i,j}$, late start $S''_{i,j}$ and late completion $C''_{i,j}$ are shown in Table 6.7

Table 6.7 Calculated parameters for Example 6.2

Activity	p (i,j)	$S'_{i,j}$	$C'_{i,j}$	$S''_{i,j}$	$C''_{i,j}$	ψ_t	ψ_f	Status
A	4	0	4	0	4	0	0	Critical
B	7	4	11	11	18	7	7	Slack
C	8	4	12	4	12	0	0	Critical
D	6	4	10	15	21	11	11	Slack
E	15	12	27	18	33	6	6	Slack
F	9	12	21	12	21	0	0	Critical
G	12	21	33	21	33	0	0	Critical
H	8	33	41	33	41	0	0	Critical

The generated schedule in Table 6.7 is shown on the Gantt chart in Figure 6.8

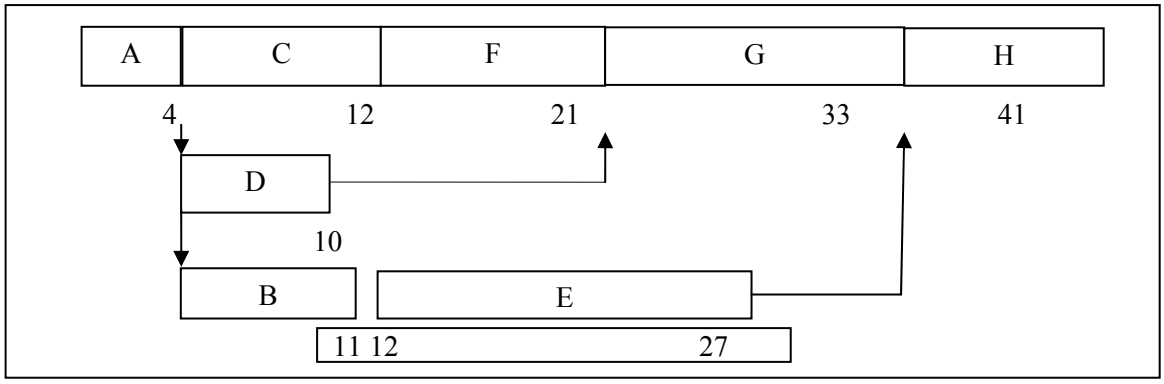


Figure 6.8 Gantt chart for Schedule in Table 6.7

6.6 PERT METHODOLOGY

PERT (Program Evaluation Review Technique) was devised in 1958 for he POLARIS missile program by the Program Evaluation Branch of the Special Projects office of the US Navy, helped by the Lockheed Missile Systems division and the Consultant firm of Booz-Allen & Hamilton. The calculations were so arranged so that they could be carried out on the IBM Naval Ordinance Research Computer (NORC) at Dahlgren, Virginia.

CPM technique deals with projects, where there is high certainty about the outcomes of activities. When there is learning process involved, degree of uncertainty is much higher and duration of activities involve considerable degree of estimation. In such situations, the PERT approach is useful, because it can accommodate the variation in activities completion times, based on an expert's or an expert committee's estimates. The activities durations are estimated using three horizons, namely; optimistic, most likely and pessimistic. Let's assign notations to activities durations for PERT methodology as follows:

$p_j^{'}$ = Optimistic (minimum) duration of j^{th} activity

p_j^{n} = Most likely (normal) duration of j^{th} activity

$p_j^{''}$ = Pessimistic (maximum) duration of j^{th} activity

$\bar{P}_j$ = Expected duration of j^{th} activity, and is calculated by following;

$$\bar{P}_j = \frac{p_j^{'} + 4p_j^{n} + p_j^{''}}{6}$$

$\bar{\mu}_j$ = Expected duration of j_{th} activity which is on the critical path

(Note that, $\bar{\mu}_j = \bar{P}_j$)

The CPM technique is applied on the problem data using $\bar{P}_j$ values to find critical activities as well as critical path. Let A_C is the set of activities on the critical path. Then, an estimate of the project completion time is

$$\bar{E}(C_{max}) = \sum_{j \in A_c} \bar{\mu}_j$$

Variance of the duration of j^{th} activity is given by

$$\sigma_j = \left(\frac{p_j'' - p_j'}{6} \right)^2$$

To obtain an estimate of the variance of the project completion time (C_{max}), consider only the activities on the critical path in the network. Since these 'critical' activities occur one after the other, add the variance of process durations of the critical activities to obtain an estimate of variance of the project completion time

$$\overline{V}(C_{max}) = \sum_{j \in A_c} \sigma_j^2$$

The distribution of the project completion time (C_{max}) is assumed to be normal with a mean of $\overline{E}(C_{max})$ and variance $\overline{V}(C_{max})$.

Example 6.3

Consider a PERT network problem. The optimistic, most likely and pessimistic task durations (in DAYS) are given in Table 6.8 below. Also, the Table contains precedence relationships.

Table 6.8 Problem data for Example 6.3

Activity	Activity Duration			Immediate Predecessor
	Optimistic	Most Likely	Pessimistic	
1	4	6	8	
2	6	8	10	
3	7	9	14	
4	3	6	9	1
5	4	6	8	1,2
6	2	3	4	2,3
7	3	8	10	3
8	10	12	14	4,5
9	9	12	15	5,6
10	8	10	18	6,7
11	2	6	10	8
12	4	9	14	9
13	3	8	13	9,10

Activity	Activity Duration			Immediate Predecessor
	Optimistic	Most Likely	Pessimistic	
14	4	8	12	10
15	7	10	13	11
16	2	6	10	12,13
17	3	7	11	14

a) What is the explicit distribution (mean, variance) of the project completion time?
b) What is the probability of finishing project no later than time 41 days?
c) What is the probability of finishing no earlier than time 38 days?
d) Suppose tasks 2 and 3 finish at time 10. What is the probability of finishing the project by time 41 days?

Solution:

The precedence network diagram is shown in Fig. 6.9

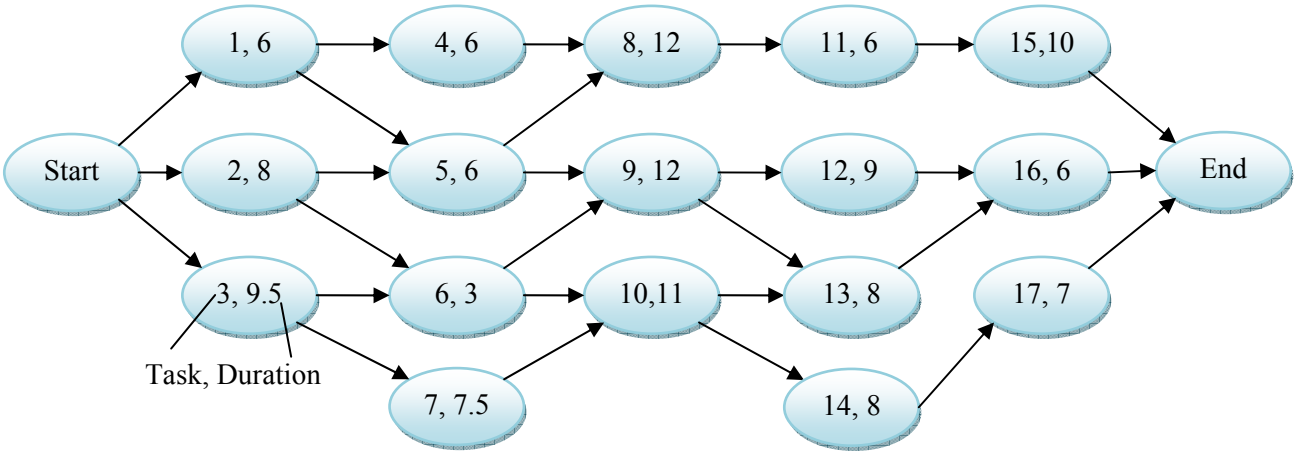


Figure 6.9 Precedence network diagram for Example 6.3

Project calculations are shown in the following Table 6.9.

Table 6.9 Calculated parameters for Example 6.3

Activity (j)	$\bar{P}_j$	S'_j	C'_j	S''_j	C''_j	Status	$\bar{\mu}_j$	σ_j^2
1	6	0	6	3	9	SLCK		
2	8	0	8	1	9	SLCK		
3	9.5	0	9.5	0	9.5	*CRTCL*	9.5	1.36
4	6	6	12	9	15	SLCK		
5	6	8	14	9	15	SLCK		
6	3	9.5	12.5	13	16	SLCK		
7	7.5	9.5	17	9.5	17	*CRTCL*	7.5	1.36
8	12	14	26	15	27	SLCK		
9	12	14	26	16	28	SLCK		
10	11	17	28	17	28	*CRTCL*	11	2.78
11	6	26	32	27	33	SLCK		
12	9	26	35	28	37	SLCK		
13	8	28	36	29	37	SLCK		
14	8	28	36	28	36	*CRTCL*	8	1.78
15	10	32	42	33	43	SLCK		
16	6	36	42	37	43	SLCK		
17	7	36	43	36	43	*CRTCL*	7	1.78

The critical path on the network is shown in Figure 6.10.

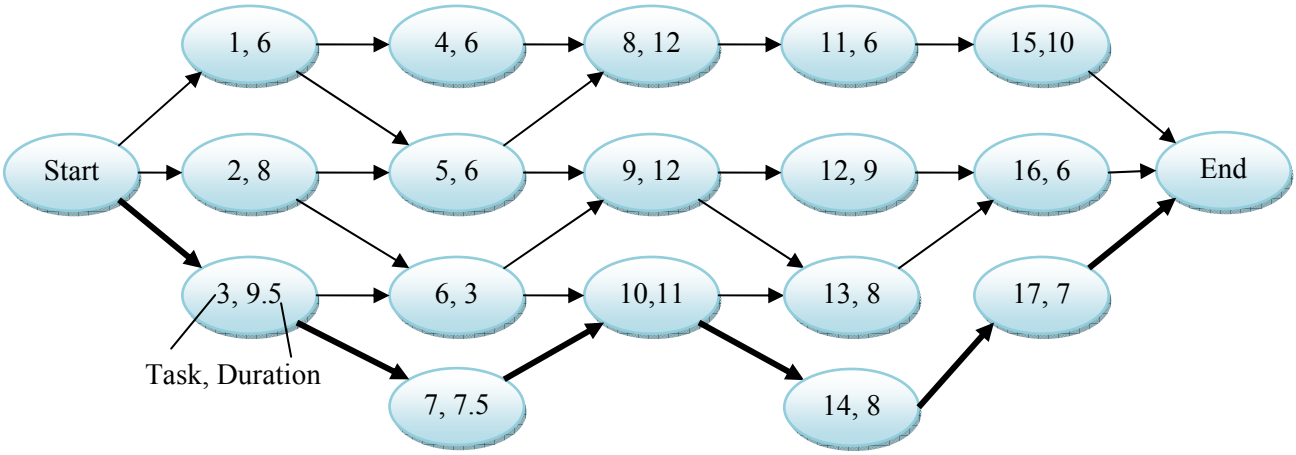


Figure 6.10 Critical Path on the network (Example 6.3)

a) Expected project completion time:

Critical Path = {3→7→10→14→17}

$$\overline{E}(C_{max}) = \overline{\mu}_3 + \overline{\mu}_7 + \overline{\mu}_{10} + \overline{\mu}_{14} + \overline{\mu}_{17} = 43 \text{ days}$$

Expected variance of the project completion time; $\overline{V}(C_{max})$ is found from;

$$\overline{V}(C_{max}) = \sum_{j \in A_c} \sigma_j^2 = \sigma_3^2 + \sigma_7^2 + \sigma_{10}^2 + \sigma_{14}^2 + \sigma_{17}^2 = 9.06$$

b) Probability that project is completed no later than 41 days is:

Probability (X <= 41 days)

$$= P\left(\frac{x-\mu}{\sigma} \le \frac{41-43}{\sqrt{9.054}}\right) = P(z \le -0.664) = 0.2546$$

Area under the normal curve from -∞ to -0.664 is 0.2546 as shown in Figure 6.11

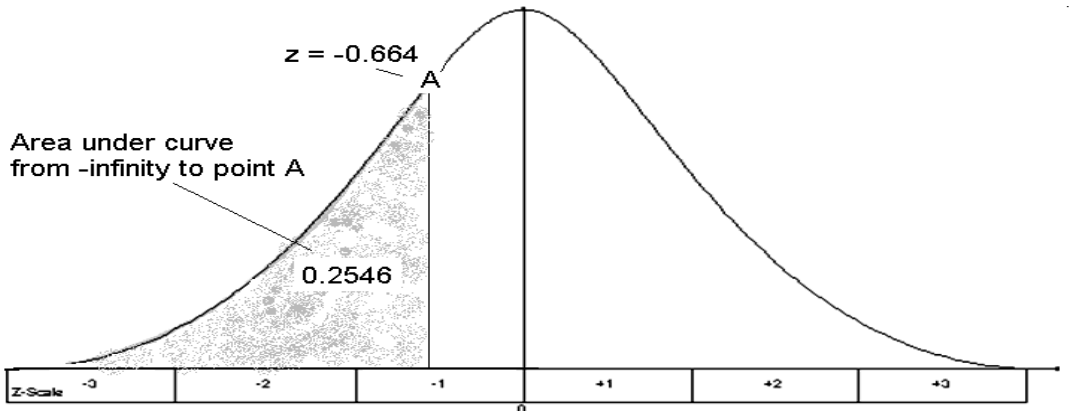


Figure 6.11 Area under normal curve $P(z \le -0.664)$

c) Probability that project is completed no earlier than 38 days is:

Probability (X >= 38 days) = 1 – P(X <= 38)

$$= 1 - P\left(\frac{x-\mu}{\sigma} \le \frac{38-43}{\sqrt{9.054}}\right) = 1 - P(z \le -1.66) = 1 - 0.0485 = 0.9515$$

Area under the normal curve from -∞ to -1.66 is 0.0485 as shown in Figure 6.12

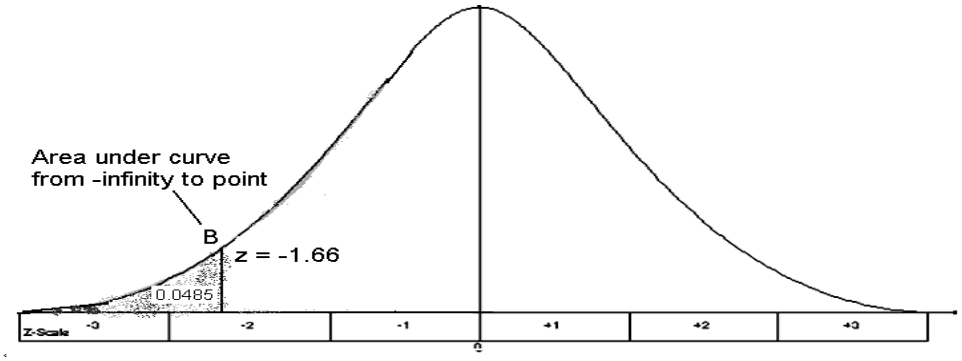


Figure 6.12 Area under normal curve $P(z \le -1.66)$

d) If Tasks 2 and 3 finishes at time 10, what is the probability that project will be completed by 41 days.

When Task 2 and Task 3 finish at time 10, the network is revised. The new values of earliest and latest completion times are shown in the network (Fig 6.13)

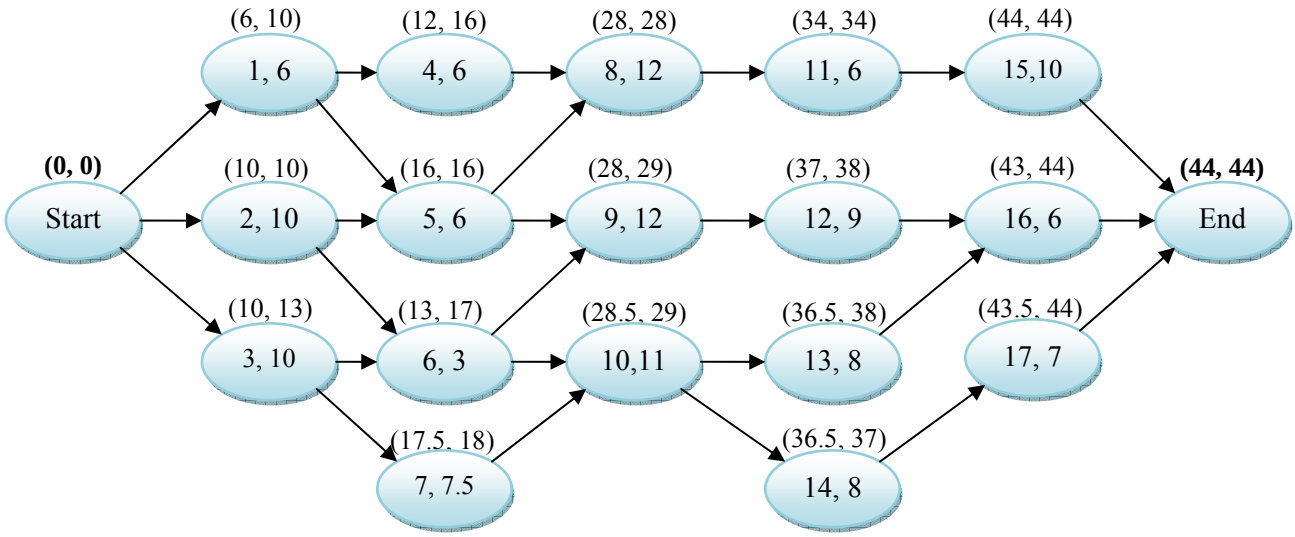


Figure 6.13 Revised precedence network with updated calculations

i) The new value of $C_{max} = 44$.

ii) New critical path is {2→ 5→ 8→ 11→15}.

iii) Expected Duration (μ) = 44 and, Variance = 3(0.44) +1.78+ 1= 4.11

Probability (X <= 41 days)

$$= P\left(\frac{x-\mu}{\sigma} \le \frac{41-44}{2.03}\right) = P(z \le -1.48) = 0.0694 .$$

Hence, there is 7% probability that project will complete by 41 days. Area under the normal curve from -∞ to -1.48 is 0.0694 as shown in Figure 6.14

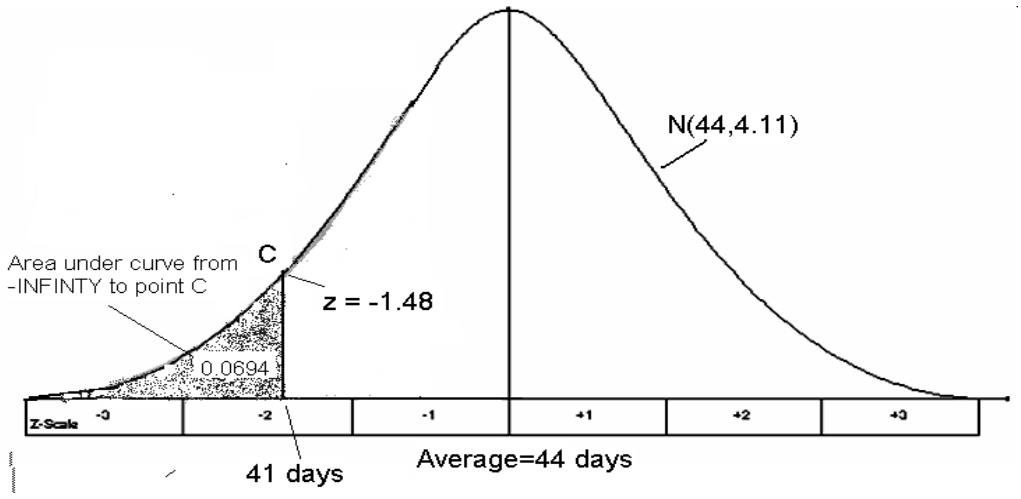


Figure 6.14 Area under normal curve $P(z \leq -1.48)$

Whereas PERT is used to estimate durations of activities, the method has certain limitations. The activity time estimates are somewhat subjective and depend on judgment. For inexperienced planners, duration of activities may be only a guess. In other cases, if the person or group performing the activity estimates the time there may be bias in the estimate. Even if the activity times are well-estimated, PERT assumes a beta distribution for these time estimates, but the actual distribution may be different. Even if the beta distribution assumption holds, PERT assumes that the probability distribution of the project completion time is the same as that of the critical path. Because other paths can become the critical path if their associated activities are delayed, PERT consistently underestimates the expected project completion time. The underestimation of the project completion time due to alternate paths becoming critical is perhaps the most serious of these issues. To overcome this limitation, Monte Carlo simulations can be performed on the network to eliminate this optimistic bias in the expected project completion time.

6.7 TIME / COST TRADE-OFF

One of the major objectives in project scheduling is minimization of project completion time (C_{max}). There is a relationship between a project's completion time (C_{max}) and its cost. The relationship depends upon the type of costs. For some types of costs, the relationship is direct proportion; i.e., as C_{max} increases, the total project cost increases. In order to accelerate the pace of the project (minimize C_{max}), more resources are acquired increasing the cost of the project. Because of these two types of costs, there is an optimal project completion time (C^*_{max}) for minimal cost. By understanding the time-cost relationship, one is better able to predict the impact of a schedule change on project cost.

6.7.1 Types of Costs

The costs associated with a project can be classified as direct costs or indirect costs. Direct costs are those directly associated with project activities, such as salaries, travel, and direct project materials and equipment. If the pace of activities is increased (to decrease duration of activities) thereby decreasing project completion time, the direct costs generally increase since more resources must be allocated to accelerate the pace.

Indirect costs are those overhead costs that are not directly associated with specific project activities such as office space, administrative staff, and taxes. Such costs tend to be relatively steady per unit of time over the life of the project. As such, the total indirect costs decrease as the project duration decreases. The project cost is the sum of the direct and indirect costs.

6.7.2 Crashing of Project

Crashing the project schedule refers to the acceleration of the project activities in order to complete the project earlier than normal time. Since project completion time (C_{max}) is determined by the activities on the critical path, so to crash a project schedule one must focus on critical path activities.

A procedure for determining the optimal project time is to determine the normal completion time (duration) for each critical path activity and a crash time (duration). The crash time (duration) is the shortest time in which an activity can be completed. The direct costs then are calculated for the normal and crash durations of each activity.

Let,

C_j^n = Normal duration cost of j^{th} activity

C_j^c = "Crash" duration cost of j^{th} activity

p_j^n = Normal duration of j^{th} activity

p_j^c = "Crash" duration of j^{th} activity

C_j = cost of reducing duration by one time unit of j^{th} activity

C_o = Fixed cost of project per unit time

C_p^k = Total project cost at k^{th} iteration.

Slope of each activity's cost versus time trade-off (C_j) can be found by

$$\text{Slope} = \frac{(\text{Crash Cost} - \text{Normal Cost})}{(\text{Normal Duration} - \text{Crash Duration})}$$

In terms of mathematical notations, C_j is determined by;

$$C_j = \frac{C_j^c - C_j^n}{P_j^n - P_j^c}$$

The graphical determination of slope C_j is shown in Figure 6.15.

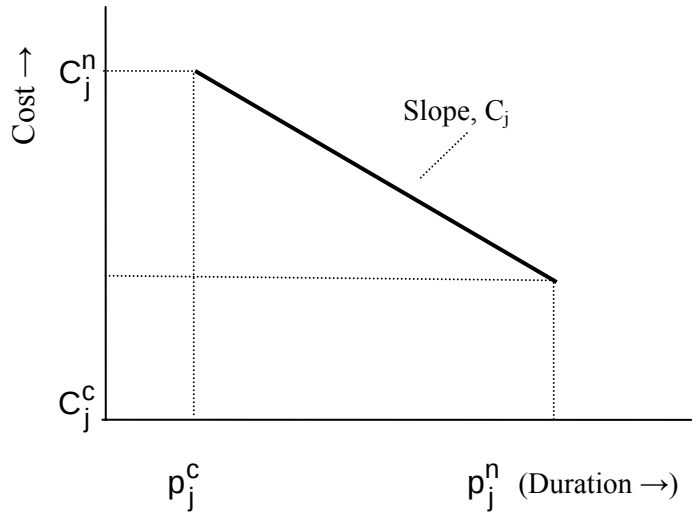


Figure 6.15 Slope calculations (C_j) for Time-Cost Trade-Off

To obtain an optimal value of project completion time (C_{max}^*), the activities having the lowest values of C_j should be shortened first. In this way, one can step through the critical path activities and create a graph of the total project cost versus the project time. The indirect, direct, and total project costs then can be calculated for different project durations. The optimal point is the duration resulting in the minimum project cost. Attention should be given to the critical path to make sure that it remains

the critical path after the activity duration is reduced. If a new critical path emerges, it must be considered in subsequent time reductions.

To minimize the cost, those activities that are not on the critical path can be extended to minimize their costs without increasing the project completion time.

6.7.3 Time-Cost Trade-off Assumptions

The time-cost model described above relies on the following assumptions:

- The normal cost for an activity is lower than the crash cost.
- There is a linear relationship between activity time and cost.
- The resources are available to shorten the activity.

A formal methodology is described by taking following steps to reduce project completion time yielding minimum total project cost.

i. Use normal durations of all activities and solve project network problem by CPM/PERT technique. Find C_{max} and, identify critical path/s on the network.

ii. For normal duration, total cost of the project will be equal to

$$C_p^0 = C_o \text{ x } C_{max}.$$

iii. Set iteration counter k = 1. To reduce project cost, find among the *unmarked* activities on the critical path having minimum value of C_j.

iv. Pick minimum cost *unmarked* activity on the critical path with least cost, and reduce duration by one unit of time. *Mark* this activity if its duration has reached p_j^c value.

Find C_{max} and new critical path/s for revised project network. Project cost for the revised network will be equal to;

$$C_p^k = C_0 \times C_{max} + \sum_{j \in U} C_j$$

Where,

U is the set of least cost activities in the present network.

IF $C_p^K \le C_p^{k-1}$ THEN go to next step, otherwise STOP.

v. Set k = k + 1. From the new critical paths found in step (iv), find if all jobs are at their p_j^c value. If yes, STOP. There is no further possibility of reduction in C_{max} value. If No, go to step (ii).

The procedure to reduce C_{max} by single unit of time in every iteration may require enormous amount of calculations. It is possible to speed up the calculations by making large reductions in an activity's duration in one iteration. However, new critical paths may emerge in this approach. The activities on the critical paths may hit their minimum, making the critical path irrelevant. Special algorithms have to be devised for such approaches.

Example 6.4

Consider the precedence graph in Figure 6.16.

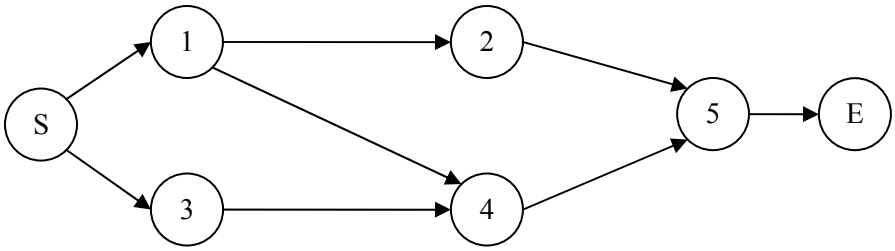


Figure 6.16 Precedence network diagram for Example 6.4

The time / cost data for the problem is as under.

Table 6.11 Normal and 'Crash' Data for Example 6.4

Activity (j)	1	2	3	4	5
p_j^{max}	3	6	4	3	2
p_j^{min}	2	5	3	2	1
C_j	12	6	10	8	16

Take C_o = 12 and find minimum project completion time and corresponding cost using Time-Cost Trade-Off.

Solution:

i) Computation of C_{max} is shown in Figure 6.17, when activities are executed at normal duration (p_j^{max}) value,

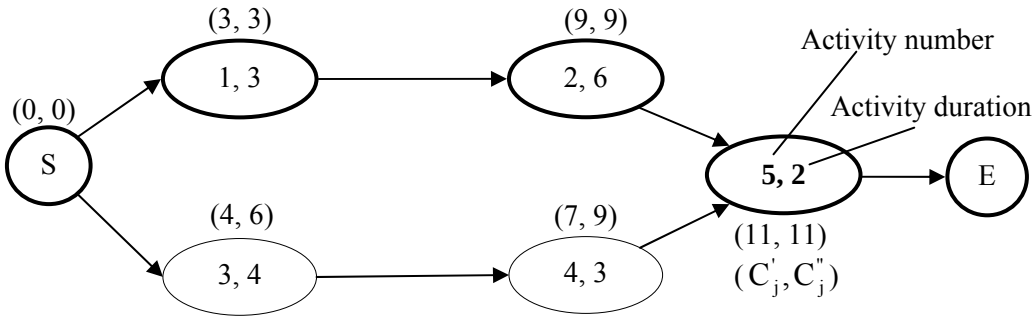


Figure 6.17 Computations of Early and Late completion times

$C_{max} = 11$, Critical Path: $1 \rightarrow 2 \rightarrow 5$

Cost of Project, $C_P^0 = C_o \times C_{max} = 12 \times 11 = 132$

ii) The activities on the critical path along with crash costs are;

Activities	1	2	5
C_j	12	6	16

Since, activity 2 has minimum Cj value, reduce duration of activity 2 by one time unit in the project network. So, $p_2^c = 5$ The network with $p_2^c = 5$ is shown in precedence network diagram (Figure 6.18)

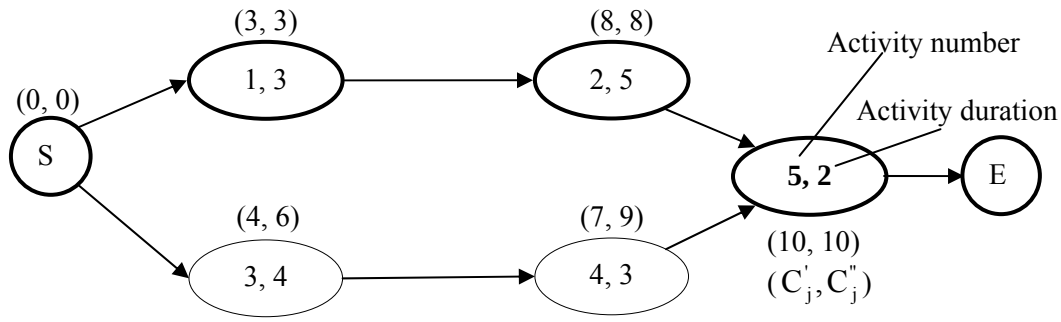


Figure 6.18 Computations of completion times with activity 2 being performed at crash duration

For this network, $C_{max} = 10$. Critical Path: $1 \rightarrow 2 \rightarrow 5$

Cost of Project, $C_P^1 = C_o \times C_{max} + C_2 = 12 \times 10 + 6 = 126$

iii) *Unmarked* Activities on critical path have the following features;

Activity	1	2	5
p_j^n	3	6	2
p_j^c	2	5	1
C_j	12	6	16

Activity 2 is hitting its minimum value. Hence, there is no further reduction in duration of activity 2. So activity 2 is removed from candidate's list. Out of the remaining two activities, activity 1 has lower value of C_j. So, reduce duration of

activity 1 by one time unit in the project network. Set, $p_1^c = 2$. Re-compute completion times with $p_1^c = 2$ as shown in Figure 6.19. The C_{max} value of this network is 9:

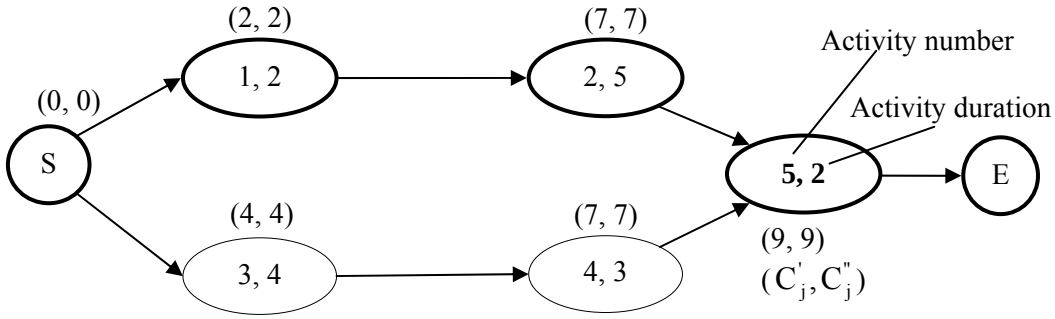


Figure 6.19 Computations of completion times with activity 1 being performed at crash duration

There are two critical paths now; $1 \rightarrow 2 \rightarrow 5$
$3 \rightarrow 4 \rightarrow 5$

Cost of Project, $C_P^2 = C_o \text{ x } C_{max} + C_2 + C_1 = 12 \times 9 + 6 + 12 = 126$

There are two critical paths in the above network. The common activity is 5.

Activities	1	2	3	4	**5**
p_j^n	-	-	4	3	**2**
p_j^c	2	5	3	2	**1**
C_j	12	6	10	8	**16**

Set $p_5^c = 1$ in the new network.

The computations of C_{max} are shown below in Figure 6.20.

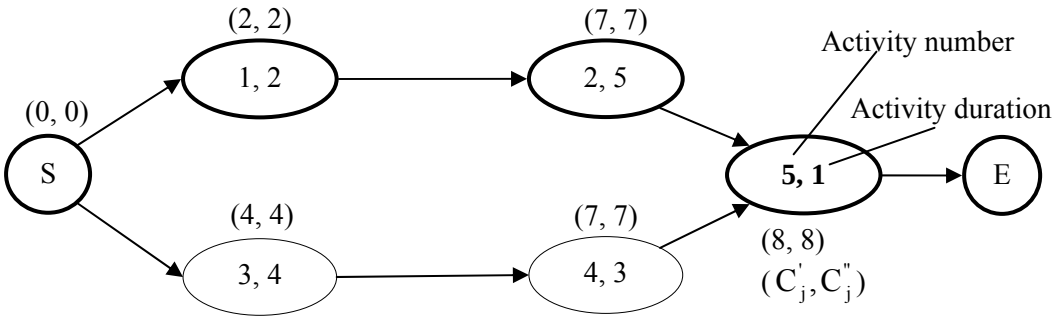


Figure 6.20 Computations of completion times with activity 5 being performed at crash duration

$C_{max} = 8$ Two critical paths are; $1 \rightarrow 2 \rightarrow 5$, $3 \rightarrow 4 \rightarrow 5$

Cost of Project, $C_P^3 = C_o \times C_{max} + C_2 + C_1 + C_5 = 12 \times 8 + 6 + 12 + 16 = 130$

The calculations are summarized in table 6.11

Table 6.11 Summary of Calculations

Iteration (k)	C_{max}	Total Cost
0	11	12 x 11 = 132
1	10	12 x 10 + 6 = 126
2	9	12 x 9 + 6 + 12 = 126
3	8	12 x 8 + 6 + 12 + 16 = 130

The optimal value of project cost is 126. There are two alternative critical paths with C_{max} values of 9 and 10.

6.7.4 Additional Considerations

There are other considerations besides project cost. For example, when the project is part of the development of a new product, time-to-market may be extremely important and it may be beneficial to accelerate the project to a point where its cost is much greater than the minimum cost.

In contract work, there may be incentive payments associated with early completion or penalties associated with late completion. A time-cost model can be adapted to take such incentives and penalties into account by modeling them as indirect costs.

EXERCISES

6.1 Assume we are to construct a pump station. The following table lists the activities to construct a pump station.

Activity No.	Activity Description	Duration in Days	Preceding Activity
1	Start	0	
2	Mobilize	2	1
3	Survey	1	2
4	Grade site	2	2
5	Trench footings	5	3, 4
6	Form and pour concrete	5	5, 8
7	Cure concrete	8	6
8	Concrete and material design	5	1
9	Spec prefab metal building	4	1
10	Plumbing materials, pump	5	1
11	Electrical materials, lights, panel	5	1
12	Install pump	7	7, 9, 10
13	Erect structural steel	4	7, 9, 10
14	Install roofing and siding	5	13
15	Install lights and panels	3	11, 14
16	Test pump	2	12
17	Paint	3	15
18	End	0	16, 17

a. Draw the network diagram for construction of a pump station and perform the required computations. When constructing the network diagram use the following format for each node:

Activity Name	
Activity Number	Activity Duration
Start Date	Finish Date
Early start Time	Early finish Time
Late start Time	Late finish Time

b. Determine critical path.
c. Based on the early start and early finish times for each activity and assuming that the project will start on March 1st, 2007, construct calendar dates for construction of a pump station. (i.e., assign a calendar date to the beginning of the first activity and convert the time durations on each activity to calendar date.).
d. Draw the Gantt chart for construction of pump station.

6.2 The following table gives the data for small project.

Project Phase	Preceding Phase	Normal Time	Crash Time	Normal Cost	Crash Cost
A	-	40	30	9,000	12,000
B	A	53	50	15,000	15,300
C	A	60	30	7,500	10,000
D	A	35	30	20,000	22,000
E	C, D	28	20	12,000	15,000
F	B, E	30	27	6,000	7,000

a. Draw the network diagram and perform the required computations.
b. Determine critical path.
c. Draw the Gantt chart.
d. There is a penalty of 3000 SR per day beyond the normal CPM duration. Perform crashing analysis to compare total normal cost to total crash cost. First crash only the critical activities and then crash all activities.
e. Assume that the variances of the durations of the project phases are as given below:

Phase	A	B	C	D	E	F
Duration Variance	10	9	600	25	40	30

Find the probability that total lateness penalty will be less than 3000 SR.
f. Repeat part e above for probability that total lateness penalty will be less than 9000 SR.

6.3 For a specific project to be accomplished, you are given the following time and predecessor for each activity in the project:

Activity	a	m	b	Immediate Predecessor
A	2	3	4	-
B	1	2	3	-
C	4	5	12	A
D	1	3	5	B&E
E	1	2	3	A

a. Draw Network Diagram.
b. Perform forward schedule.
c. Perform backward schedule.
d. In table format, determine the ES, EF, LS, LF, and slack for all activities.
e. Draw Gantt chart for the Early Start Schedule for the project (5 points).
f. Find the Critical Path.
g. What is the variance in completion time for the critical path found?

6.4 Development of a new deluxe version of a particular software product is being considered. The activities necessary for the completion of this project are listed in the table below.

Activity	Normal Time (week)	Crash Time (week)	Normal Cost	Crash Cost	Immediate Predecessor
A	4	3	2,000	2,600	-
B	2	1	2,200	2,800	-
C	3	3	500	500	-
D	8	4	2,300	2,600	A
E	6	3	900	1,200	B
F	3	2	3,000	4,200	C
G	4	2	1,400	2,000	D, E

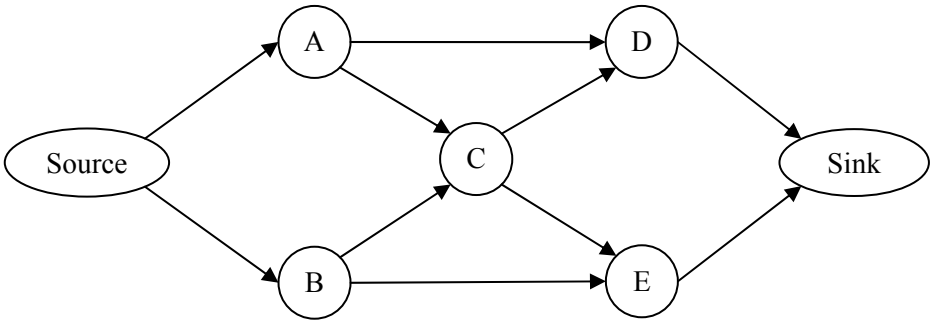
a. What is the project expected completion date?
b. What is the total cost required for completing this project on normal time?

c. If you wish to reduce the time required completing this project by 1 week, which activity should be crashed, and how much will this increase the total cost?

6.5 A project has an expected completion time of 40 weeks and a standard deviation of 5 weeks. It is assumed that the project completion time is normally distributed.

a. What is the probability of finishing the project in 50 weeks or less?
b. What is the probability of finishing the project in 38 weeks or less?
c. The due date for the project is set so that there is a 90% chance that the project will be finished by this date. What is the due date?

6.6 Consider the following precedence graph shown below:



The time/cost data for the problem is shown in the following table:

Activity (j)	A	B	C	D	E
$p_j^{\max}$	5	8	3	8	4
$p_j^{\min}$	5	7	1	7	3
C_j	8	5	9	5	11

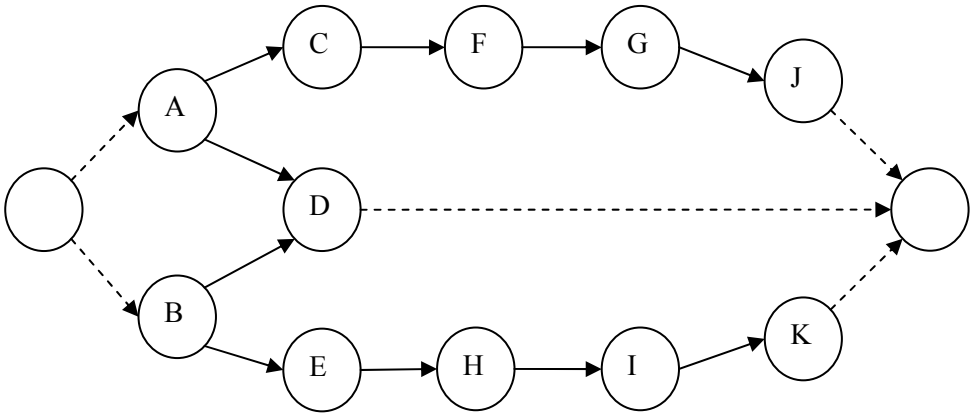
Given that C_O is 15 do the following:

a. Find project completion time ($C_{\max}$) as well as project cost if all activities are performed at $p_j^{\max}$
b. Find project completion time ($C_{\max}$) as well as project cost if all activities are performed at $p_j^{\min}$
c. Use time-cost trade-off method, and find Minimum Cost project Schedule.

6.7 Precedence graph of a small project comprising 11 jobs/tasks is shown in Fig.1. The time/cost trade-off data including maximum process time (p_j^{max}), minimum process time (p_j^{min}) and cost of the jobs/tasks for reducing process time from maximum to minimum value per day (C_j) is shown in Table. Take fixed cost of the project per time unit (C_0) as equal to 12.

Job (j)	A	B	C	D	E	F	G	H	I	J	K
p_j^{max}	5	3	9	4	2	8	11	12	6	12	7
p_j^{min}	4	2	8	3	2	7	6	12	5	11	7
C_j	4	3	5	4	9	1	3	2	4	3	5

Find the optimal project completion time to provide **minimum total costs**. Also show the critical path/s for this schedule.



Precedence Graph

6.8 For a specific project to be completed, you are given the following activities and their predecessor, durations, and cost in the table below:

Activity	Immediate Predecessor	Optimistic time	Most likely time	Pessimistic time	Cost
A	-	5	9	13	0
B	-	6	14	34	2000
C	A	8	13	30	0
D	C	4	11	18	15000
E	D	9	10	11	4000
F	A	19	38	63	40000

Activity	Immediate Predecessor	Optimistic time	Most likely time	Pessimistic time	Cost
G	E & F	46	75	80	260000
H	B	10	30	50	15000
I	H	9	16	17	10000
J	F & I	10	17	30	150000
K	F & I	16	22	28	15000
L	K	8	22	24	100000
M	J & K	15	23	43	18000
N	M	8	10	18	7000
O	A	30	36	60	12000
P	D & O	41	47	77	90000
Q	P	4	5	6	80000
R	G & Q	3	6	21	26000

Then, perform the following:

a. Draw network diagram and do the forward and the backward schedule for the project.
b. Determine the expected completion time and the critical path of this project.
c. In table format, determine the early start, early finish, late start, late finish, and slack for all activities,
d. Draw Gantt chart for the early start schedule for the project and show the critical path.
e. What is the probability of finishing the project in 120 days?

6.9 Assume that the status of the project given in the question 6.8 on the 58 days in the project time is as follows:

Activity	Percent completed	Activity	Percent completed	Activity	Percent completed
A	100	G	0	M	0
B	100	H	100	N	0
C	100	I	100	O	50
D	100	J	75	P	0
E	50	K	25	Q	0
F	100	L	0	R	0

The total expenditures to date are 195000. Then, perform the following:

a. Analyze the progress of the project from both a budget and time point of view.
b. If the project is not completed on the expected completion time, then, the event of celebrating the end of the project will be cancelled and this will cost 100000. Thus, there are two suggestions to reduce the times of the remaining activities as follows:
    - Reduce the most likely and pessimistic times for activity L by 50% and this will cost 8000.
    - Reduce activity N by three days and this will cost 10000.

Then, which suggestion or combination of suggestion should you consider adopting.

6.10 Perform the CPM analysis for the given project using the following data in the table below:

Activity	Predecessor	Duration
A	-	2
B	A	4
C	B	6
D	A	3
E	B	2
F	E	1
G	C,D	2
H	G	3
I	H	4

When performing the analysis assume the following:

a. The project cannot start until 3 days from time zero (i.e. ES at start node = 3).
b. There is deadline of 30 days (i.e. LC at finish node = 30).

Draw the CPM network and perform the required computations. For every activity, show in a table ES, LS, EC, LC, and TS. Also, in the same table for every activity show whether the activity is critical or non-critical.

6.11 Perform the PERT analysis for the project using the following data in the table below:

Activity No	a	m	b	Preceding activity
1	1	4	5	-
2	2	3	4	-
3	6	10	13	1
4	6	6	7	1
5	2	2	2	2
6	1	2	3	3
7	5	8	9	4,5
8	12	16	19	2

For every activity, compute the average activity duration and the standard deviation of activity duration. Then, draw the PERT network and perform the required computations. For every activity, list in a table format the following: ES, LS, EC, LC, and TS. Also, for every activity show in the same table whether the activity is critical or non-critical. Then, find the probabilities of finishing the project in 15 unit of time. Also, find the probabilities of finishing the project in 21 unit of time.

7

Resource Constrained Project Scheduling

CHAPTER CONTENTS

7.1 INTRODUCTION

The CPM/PERT methodology presented in previous chapter emphasized on time factor for project scheduling problem. However, both time and resources are equally important in any real world scheduling problem. But, often, resources are limited. Scarcity of resources necessitate that these should be utilized optimally. The project scheduling problem with limited resources has two fold objectives;

1. Minimize project completion time (C_{max})
2. Assign available resources so to ensure their optimal utilization

Optimal solutions to these problems are very difficult to obtain, especially for large scale networks. This section is deal with the issues, techniques, and tools for resource management in projects. Objectives are achieved through the assignment of resources to areas of need. Consequently, resource management is essential to achieve successful operations. Resource management strategies will depend on the specific resources that are to be managed. Some resources possess special skills. Some are in very limited supply. The relative importance of different resource types should be considered for resource management purposes. Topics covered in this chapter include resource allocation in project networks; resource sharing, human resource management, resource utilization analysis.

Project goals are achieved through the utilization of resources. Resource refers to the manpower, tools, equipment, and other physical items that are available to achieve project goals. Not all resources are necessarily tangible. Conceptual knowledge, intellectual property, and skill can be classified as resources. The lack or untimely availability of resources is a major impediment to manufacturing and automation efforts. Resource management is a complex task that is affected by several constraints. These constraints include:

- Resource interdependencies
- Conflicting resource priorities
- Mutual exclusivity of resources
- Limitations on resource availability
- Limitations on resource substitutions
- Variable levels of resource availability
- Limitations on partial resource allocation

The above factors determine the tools and techniques that can be used for resource management.

7.2 RESOURCE ALLOCATION IN PROJECT

Basic CPM and PERT approaches assume unlimited resource availability in project network analysis. In this section, both the time and resource requirements of activities are considered in developing schedules. Projects are subject to three major constraints: time limitations, resource constraints, and performance requirements. Since these constraints are difficult to satisfy simultaneously, trade-offs must be made. The smaller the resource base, the longer the project schedule. The quality of work may also be adversely affected by poor resource allocation strategies.

Good planning, scheduling, and control strategies must be developed to determine what the next desired state of a project is, when the next state is expected to be reached, and how to move toward that next state. Resource availability as well as other internal and external factors will determine the nature of the progress of a project from one state to another. Network diagrams, Gantt charts, progress charts, and resource loading graphs are visual aids for resource allocation strategies. One of the first requirements for re-source management is to determine what resources are required versus what resources are available. Table 7.1 shows a model of a resource availability data. The data is essential when planning resource loading strategies for resource-constrained projects.

Table 7.1 Format for Resource Availability Data

Resource Type	Description	Job Function	When Available	Duration of Availability	How many
1	Manager	Planning	3	10	1
2	Analyst	Scheduling	Now	Indefinite	3
3	Engineer	Design	Now	36	2
…	…	…	…	…	….
…	…	…	…	…	…
n	Operator	Machining	Immediate	Indefinite	12

7.3 RESOURCE-CONSTRAINED SCHEDULING

A resource-constrained scheduling problem arises when the available re-sources are not enough to satisfy the requirements of activities that can be performed concurrently. To satisfy this constraint, sequencing rules (also called priority rules, activity urgency factor, scheduling rules, or heuristics) are used to determine which of the competing activities will have priority for resource allocation. Several optimum-yielding techniques are available for generating resource-constrained schedules.

Unfortunately, the optimal techniques are not generally used in practice because of the complexity involved in implementing them for large projects.

Even using a computer to generate an optimal schedule is sometimes cumbersome because of the modeling requirements, the drudgery of lengthy data entry, and the combinatorial nature of interactions among activities. However, whenever possible, effort should be made in using these methods since they provide the best solution.

Most of the available mathematical techniques are based on integer programming that formulates the problem using and 1 indicator variables. The variables indicate whether or not an activity is scheduled in specific time periods. Three of the common objectives in project network analysis are to minimize project duration, to minimize total project cost, and to maximize resource utilization. One or more of these objectives are attempted, subject to one or more of the constraints below:

1. Limitation on resource availability.
2. Precedence restrictions.
3. Activity-splitting restrictions.
4. Non-preemption of activities.
5. Project deadlines.
6. Resource substitutions.
7. Partial resource assignments.
8. Mutually exclusive activities.
9. Variable resource availability.
10. Variable activity durations.

Instead of using mathematical formulations, a scheduling heuristic uses logical rules to prioritize and assign resources to competing activities. Many scheduling rules have been developed in recent years. Some of the most frequently used ones are presented here.

7.4 RESOURCE ALLOCATION RULES

Resource allocation heuristics facilitate ease of scheduling large projects subject to resource limitations. Some heuristics are very simple and intuitive, while others require computer implementations. Several scheduling heuristics have been developed in recent years. Many of these are widely applied to real projects. Many project management software packages use proprietary resource allocation rules that are not transparent to the user. A good scheduling heuristic should be simple, unambiguous, easily understood, and easily executable by those who must use it. The heuristic must be flexible and capable for resolving schedule conflicts. When users

trust and use a scheduling heuristic, then project scheduling becomes an effective communication tool in project management. Some of the most frequently used scheduling rules are presented below.

ACTlM (Activity Time): ACTIM (Whitehouse and Brown, 1979) is one of the earlier activity sequencing rules. The rule was developed by George H. Brooks and used in his algorithm, Brooks' algorithm (Brooks and White, 1965; Bedworth, 1973). The original algorithm considered only the single project, single resource case, but it lends itself to extensions for the source cases. The ACTIM scheduling heuristic represents the maximum time that an activity controls in the project network on any one path. This is represented as the length of time between when the activity finishes and when the project finishes. It is computed for each project activity by subtracting the activity's latest start time from the critical path time as shown below:

ACTIM = (Critical path time) - (Activity latest start time)

ACTRES (Activity Resource): ACTRES is a scheduling heuristic proposed by (1973). This is a combination of the activity time and resource requirements. It is computed as

ACTRES = (Activity time) (Resource requirement)

For multiple resources, the computation of ACTRES can be modified to account for various resource types. For this purpose, the resource requirement can be replaced by a scaled sum of resource requirements over different resource types.

TIMRES (Time Resources): TIMRES is another priority rule proposed by Bedworth (1973). It is composed of equally weighted portions of ACTIM and ACTRES. It is expressed as

TIMRES = 0.5(ACTIM) + 0.5(ACTRES)

GENRES: GENRES is a search technique proposed by Whitehouse and Brown (1979) as an extension of Brooks' algorithm (Brooks and White, 1965). It is a modification of TIMRES with various weighted combinations of ACTIM and ACTRES. GENRES is implemented as a computer search technique whereby iterative weights (w) between 0 and 1 are used in the following expression:

GENRES = (w)(ACTIM) + (1 - w)(ACTRES)

ROT (Resource Over Time): ROT is a scheduling criterion proposed by Elsayed (1982). It is calculated as the resource requirement divided by the activity time:

$$\text{ROT} = \frac{\text{Resource requirement}}{\text{Activity time}}$$

The resource requirement can be replaced by the scaled sum of resource requirements in the case of multiple resource types with different units.

CAF (Composite Allocation Factor): CAF is a comprehensive rule developed by Badiru (1988c). For each activity ***i*,** CAF is computed as a weighted and scaled sum of two components **RAF** (resource allocation factor) and **SAF** (stochastic activity duration factor) as follows:

$$CAF_i = (w)RAF_i + (1\text{-}w)SAF_i$$

where w is a weight between **0** and 1. **RAF** is defined for each activity i as

$$RAF_i = \frac{1}{t_i}\sum_{j=1}^{R}\frac{x_{ij}}{y_j}$$

where,
x_{ij} = number of resource type **j** units required by activity i
$y_j = \text{Max}\{x_{ij}\}$, maximum units of resource type ***j*** required
t_i = the expected duration of activity i
R = the number of resource types

RAF is a measure of the expected resource consumption per unit time. In the case of multiple resource types, the different resource units are scaled by the y_j component in the formula for RAF. This yields dimensionless quantities that can be summed in the formula for RAF. The **RAF** formula yields real numbers that are expressed per unit time. To eliminate the time -based unit, the following scaling method is used:

$$\text{Scaled } \mathbf{RAF}_i = \frac{\mathbf{RAF}_i}{\mathbf{Max\{RAF}_i\}}(100)$$

The above scaling approach yields unitless values of **RAF** between 0 and 100 for the activities in the project. Resource-intensive activities have larger magnitudes for RAF and therefore require a higher priority in the scheduling process. To incorporate the stochastic nature of activity times in a project schedule, **SAF** is defined for each activity *i* as

$$SAF_i = t_i + \frac{s_i}{t_i}$$

where,
t_i = expected duration for activity *i*
s_i = standard deviation of duration for activity *i*

s_i/t_i = coefficient of variation of the duration of activity *i*

It should be noted that the formula for SAF has one component (ti) with units of time and one component s_i/t_i with no units. To facilitate the required arithmetic operation, each component is scaled as shown below:

$$\text{Scaled } t_i = \frac{t_i}{\text{Max}\{t_i\}}(50)$$

$$\text{Scaled } (s_i/t_i) = \frac{(s_i/t_i)}{\text{Max}\{s_i/t_i\}}(50)$$

When the above scaled components are plugged into the formula for SAF, we automatically obtain unitless scaled SAF values that are on a scale of 0 to 100. However, the 100 weight will be observed only if the same activity has the highest scaled t_i value and the highest scaled s_i/t_i, value at the same time. Similarly, the 0 weight will be observed only if the same activity has the lowest scaled t_i value and the lowest scaled s_i/t_i value at the same time. The scaled values of SAF and RAF are now inserted in the formula for CAF as shown below:

$$CAF_i = (w)\ \{\text{scaled } RAF_i\} + (1 - w)\ \{\text{scaled } SAF_j\}$$

To ensure that the resulting **CAF** values range from 0 to 100, the following final scaling approach is applied:

$$\text{Scaled } CAF_i = \frac{CAF_i}{\text{Max}\{CAF_i\}}(100)$$

It is on the basis of the magnitudes of CAF that an activity is assigned a priority for resource allocation in the project schedule. Activities with larger values of **CAF** have higher priorities for resource allocation. An activity that last longer, consumes more resources, and varies more in duration will have a larger magnitude of CAF.

RSM (Resource Scheduling Method): RSM was developed by Brand, Meyer, and Shaffer (1964). The rule gives priority to the activity with the minimum value of d_{ij} calculated as follows:

d_{ij} = increase in project duration when activity j follows activity i

= Max $\{0, (EC_i - LS_j)\}$

where, **EC** is the earliest completion time of activity i and LS; is the latest start time of activity j**.** Competing activities are compared two at a time in the resource allocation process.

***GRD (Greatest Resource* Demand):** This rule gives priority to the activity with the largest total resource -unit requirements. The **GRD** measure is calculated as

$$g_j = d_j \sum_{i=1}^{n} r_{ij}$$

where
g_j = priority measure for activity j
d_j = duration of activity j
r_{ij} = units of resource type i required by activity j per period
n = number of resource types (resource units are expressed in common units)

GRU (Greatest Resource Utilization): The **GRU** rule assigns priority to activities that, if scheduled, will result in maximum utilization of resources or minimum idle time. For large problems, computer procedures are often required to evaluate the various possible combinations of activities and the associated utilization levels.

Most Possible Jobs: This approach assigns priority in such a way that the greatest numbers of activities are scheduled in any period.

Other Scheduling Rules
Most total successors
Most critical activity
Most immediate successors
Any activity that will finish first
Minimum activity latest start (Min LS)
Maximum activity latest start (Max LS)
Minimum activity earliest start (Min ES)
Maximum activity latest completion (Max LC)
Minimum activity earliest completion (Min EC)
Maximum activity earliest completion (Max EC)
Minimum activity latest completion (Min LC)
Maximum activity earliest start (Max ES)
Minimum activity total slack (Min TS)
Maximum activity total slack (Max TS)
Any activity that can start first
Minimum activity duration
Maximum activity duration

The project analyst must carefully analyze the prevailing project situation and decide which of the several rules will be most suitable for the resource allocation

requirements. Since there are numerous rules, it is often difficult to know which rule to apply. Experience and intuition are often the best guide for selecting and implementing scheduling heuristics. Some of the shortcomings of heuristics include subjectivity of the technique, arbitrariness of the procedures, lack of responsiveness to dynamic changes in the project scenario, and oversimplified assumptions.

7.5 CATEGORISATION OF RESOURCE SCHEDULING TECHNIQUES

In the following paragraphs, heuristic scheduling techniques are introduced. The resource constrained scheduling problems can be categorized into four types as follows;

i. Single-Resource, Single-Capacity
ii. Single-Resource, Multi-Capacity
iii. Multiple-Resource, Single-Capacity
iv. Multiple-Resource, Multiple-Capacity

Single-Resource Single-Capacity problems are equivalent to classical Single Machine scheduling problems and, are of no practical use in project scheduling. Other three types of problems are described in this chapter.

7.6 SINGLE-RESOURCE MULTI-CAPACITY SCHEDULING

In this type of scheduling, only one type of resource is considered in multiple quantities. For example, each activity of a construction project requires services of labor force. The management has a fixed number of workers in its pool. City transport system requires the services of bus drivers. The management has specific number of drivers in its staff. A telephone company requires the services of technicians to install telephone exchange. The company has a certain number of technicians available on its list. When project schedule is prepared, both precedence as well as resource constraints have to be taken care off. A heuristic methodology called ACTIM is presented to develop project schedule for this type of resource constrained problem.

ACTIM Procedure:

i) Develop project network to find C_{max}
ii) Determine for each activity, the maximum time it controls through network. Call it ACTIM score. Rank the activities in decreasing ACTIM sequence.
iii) Compute S_j for all unscheduled tasks. Schedule tasks according to ACTIM rank.
iv) If all tasks are finished, STOP. Otherwise go to step (3).

Example 7.1

Consider following project network in Figure 7.1. Develop project schedule. The resource requirements for this project are workers. Each activity requires one worker for the execution. At present two workers are available.

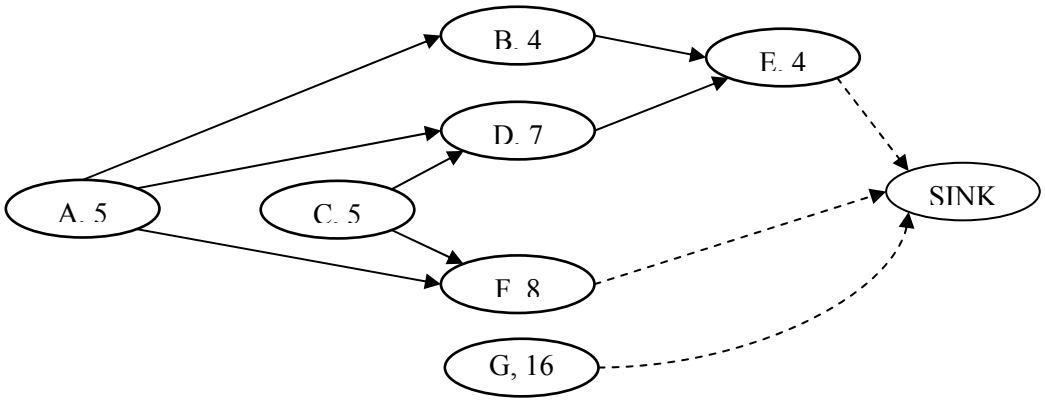


Figure 7.1 Project Network For Example 7.1

Solution:

Find ACTIM for each activity

Job	A	B	C	D	E	F	G
ACTIM	16	11	16	11	4	8	16

Rank activities according to ACTIM. Hence, ordered list of activities is {A, C, G, B, D, F, E}

Find earliest start time S_j for all activities from figure 7.2

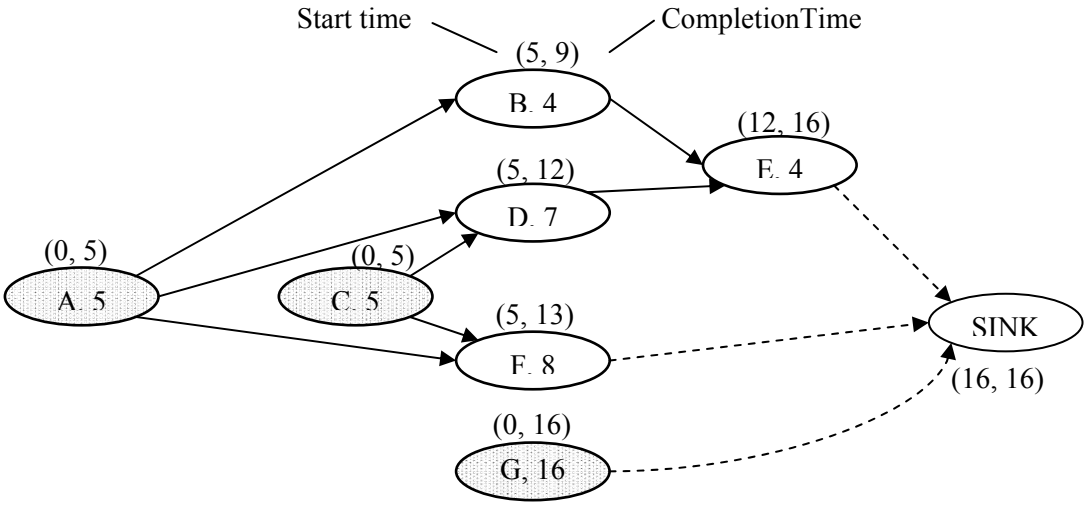


Figure 7.2 Project Network with Early star and completion time

The activities with zero earliest start time are {A, C, G} as shown below in Table 7.2

Table 7.2 Activities with Zero Earliest Start time

Activity	A	C	G	B	D	F	E
P_j	5	5	16	4	7	8	4
S'_j	0	0	0	5	5	5	12
C'_j	5	5	16	9	12	13	16

At time zero {TNOW = 0}, three activities with equal rank are schedulable. These activities are {A, C, G}. However, two workers are available. Hence, schedule activities A and G at time zero. Gantt chart is shown below in Figure 7.3.

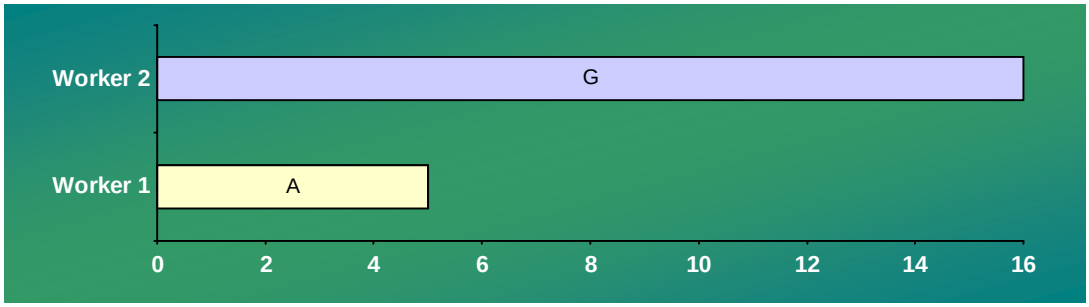


Figure 7.3 Gantt chart for partial schedule

Note, worker 1 will be free at time 5, and, worker 2 will be free at time 16.

Update S_j for unscheduled activities in the network as shown in Figure 7.4. (Add an arc from activity A to activity C)

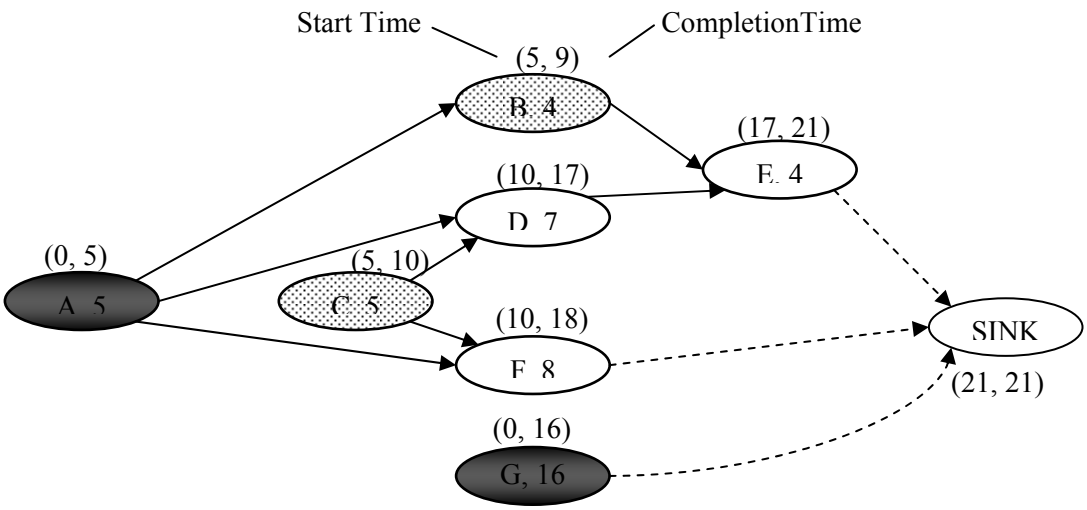


Figure 7.4 Project network with new early start and completion time

New S_j values for unscheduled activities are;

Note, activities B and C have earliest start time of 5. Since, Worker 1 is available at time, t = 5, schedule next activity at time 5. So, TNOW = 5. Schedulable activities are {C, B}. Since, activity C has higher ACTIM rank than activity B, schedule C at time 5.

Updated Gantt chart is shown below.

Activity	C	B	D	F	E
P_j	5	4	7	8	4
S'_j	5	5	10	10	17
C'_j	**10**	**9**	17	18	21

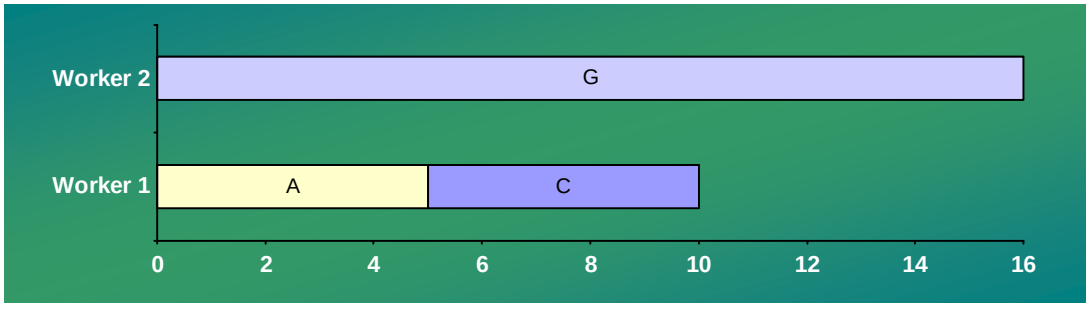


Figure 7.5 Gantt chart for partial schedule.

Note that worker 1 will be free now at time 10. Update S_j values for unscheduled activities as shown in Figure 7.6. (Add arc from C to B)

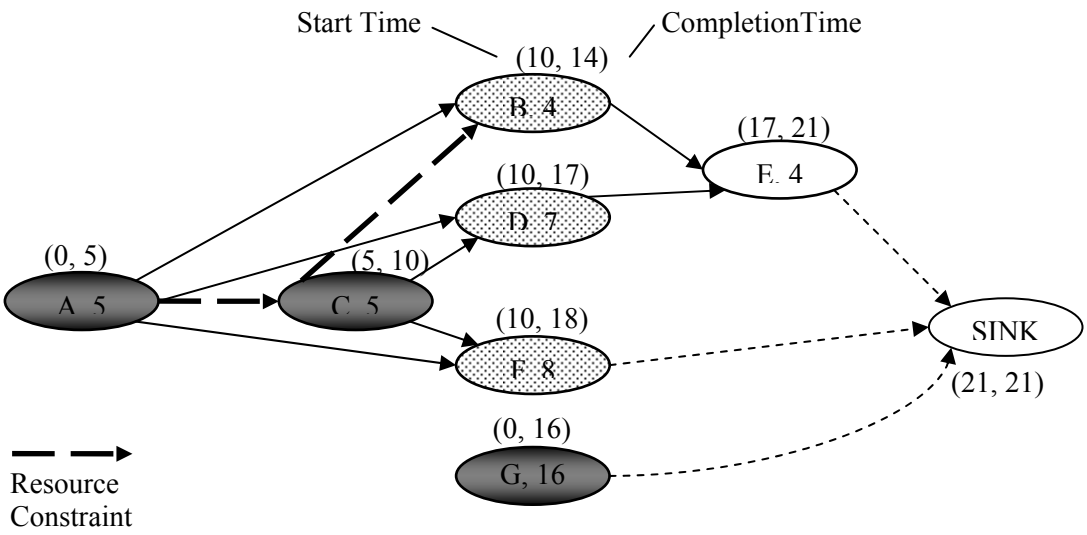


Figure 7.6 Project network with new early start and completion time

New S_j values for unscheduled activities are;

Activities B, D and F are schedulable at time t=10. Only worker 1 is available at this time. Since activity B is higher in ACTIM rank, schedule activity B at time t = 10. The updated Gantt chart is shown below.

Activity	B	D	F	E
P_j	4	7	8	4
S'_j	10	10	10	17
C'_j	**14**	**17**	**18**	21

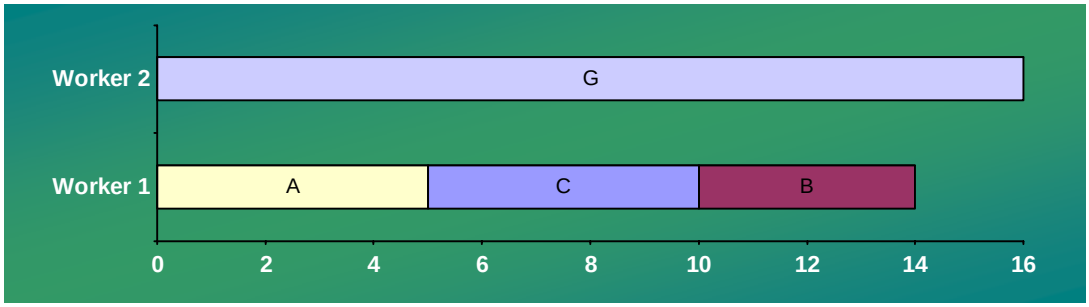


Figure 7.7 Gantt chart for partial schedule.

Update S_j values of unscheduled tasks as shown in Figure 7.8 (add arcs from B to D and, from B to F)

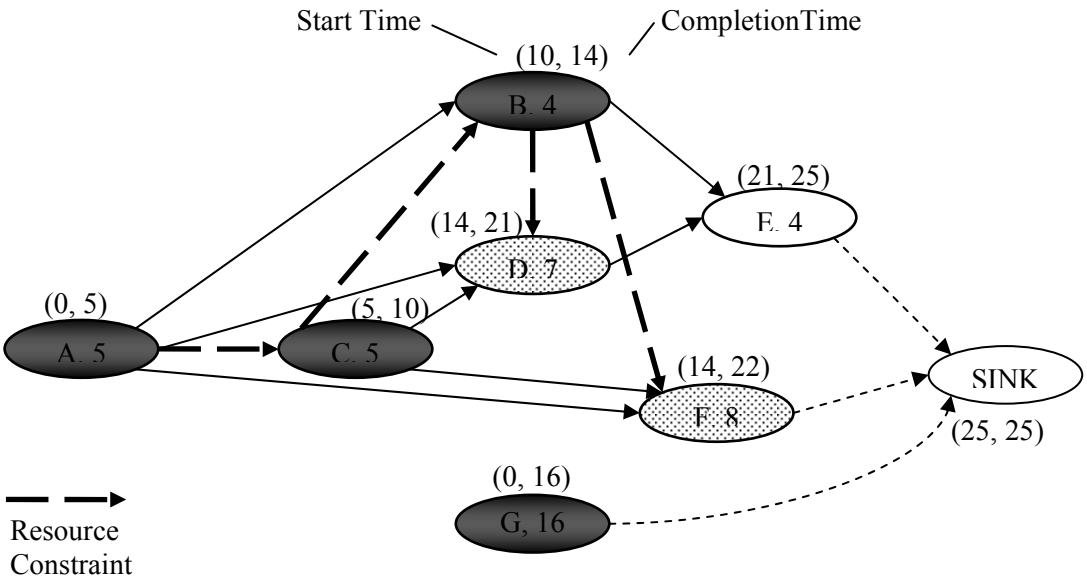


Figure 7.8 Project network with new early start and completion time

Calculate S_j' values for unscheduled activities;

Activity	D	F	E
P_j	7	8	4
S_j'	14	14	21
C_j'	**21**	**22**	25

Activities D and F are schedulable at time 14. Only Worker 1 is available at time 14.

Hence, TNOW = 14. Since activity D is higher in ACTIM rank, schedule D at time 14. Update Gantt chart as follows:

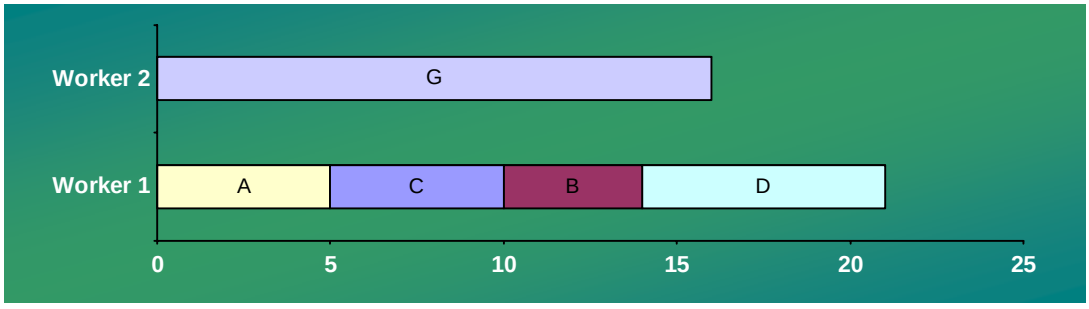


Figure 7.9 Gantt chart for partial schedule.

Note, worker 2 will be free at time t = 16.
Calculate new values of early start and completion times (Figure 7.10)

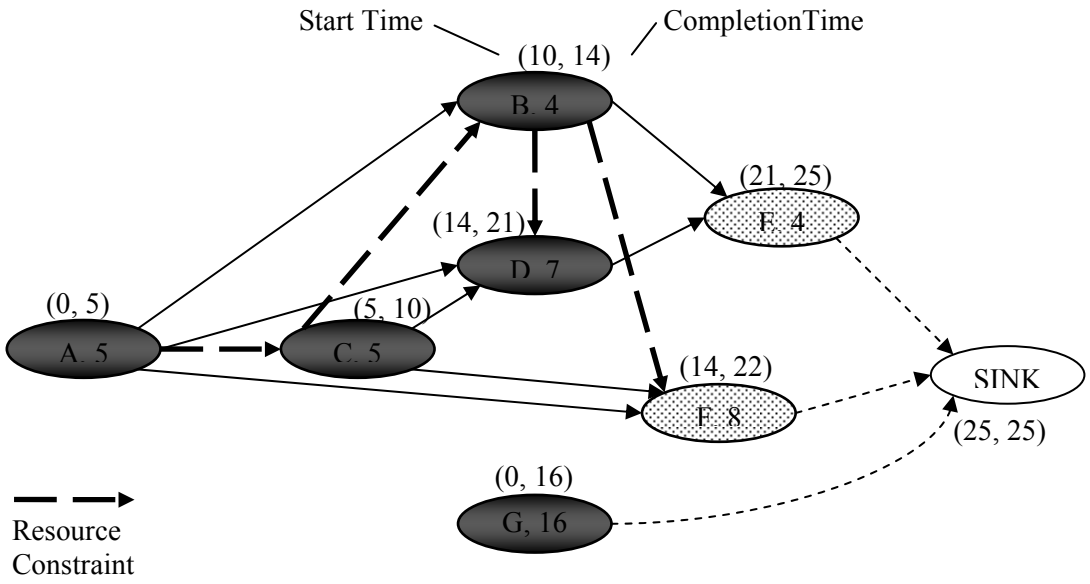


Figure 7.10 Project network with new early start and completion time

Calculate $S_j^{'}$ values for unscheduled activities;

Activity	F	E
P_j	8	4
$S_j^{'}$	14	21
$C_j^{'}$	**22**	25

Worker 2 is available at t = 16. Schedule activity F at time 16 and, engage worker 2. TNOW = 16, Schedule F.
Updated Gantt chart is shown below.

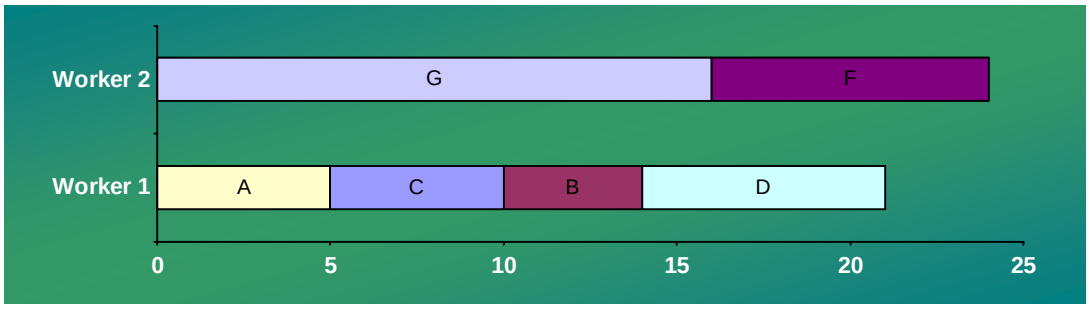


Figure 7.11 Gantt chart for partial schedule.

Worker 1 is available at time 21. Schedule E at time 21 and engage worker 1. The complete schedule is shown by Gantt chart in Figure 7.12. Project completion time; $C_{max} = 25$

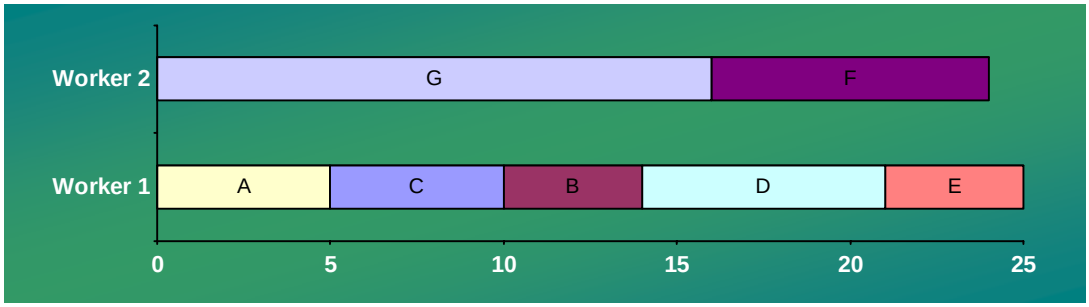


Figure 7.12 Gantt chart for complete project schedule

Example 7.2

Revise project schedule for Example 7.1 if **three** workers are available.

Solution:

The ranked list of activities according to ACTIM is as under;
{A, C, G, B, D, F, E}

The earliest start time S_j' for all activities is shown in table 7.3;

Table 7.3 Activities with Zero Earliest Start time

Activity	A	C	G	B	D	F	E
P_j	5	5	16	4	7	8	4
S_j'	0	0	0	5	5	5	12
C_j'	5	5	16	9	12	13	16

At time zero {TNOW = 0}, three activities with equal rank are schedulable. These activities are {A, C, G}. Since three workers are available, schedule activities A, C and G at time zero. Gantt chart is shown below in figure 7.13.

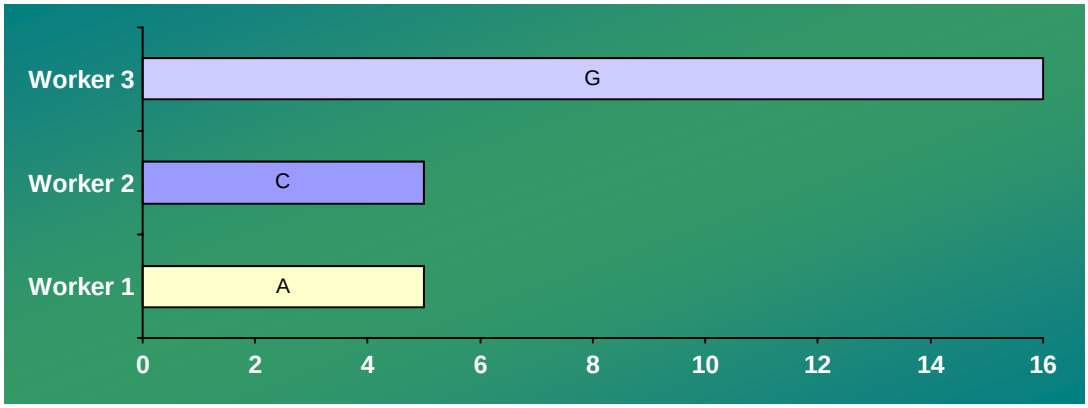


Figure 7.13 Gantt chart for partial schedule

Note, workers 1 and 2 will be free at time 5, and, worker 3 will be free at time 16. The new available times for workers are as follows:

Worker 1	Worker 2	Worker 3
5	**5**	**16**

Update S_j' for unscheduled activities in the network as shown in Figure 7.14 (Add an arc from activity C to activity B)

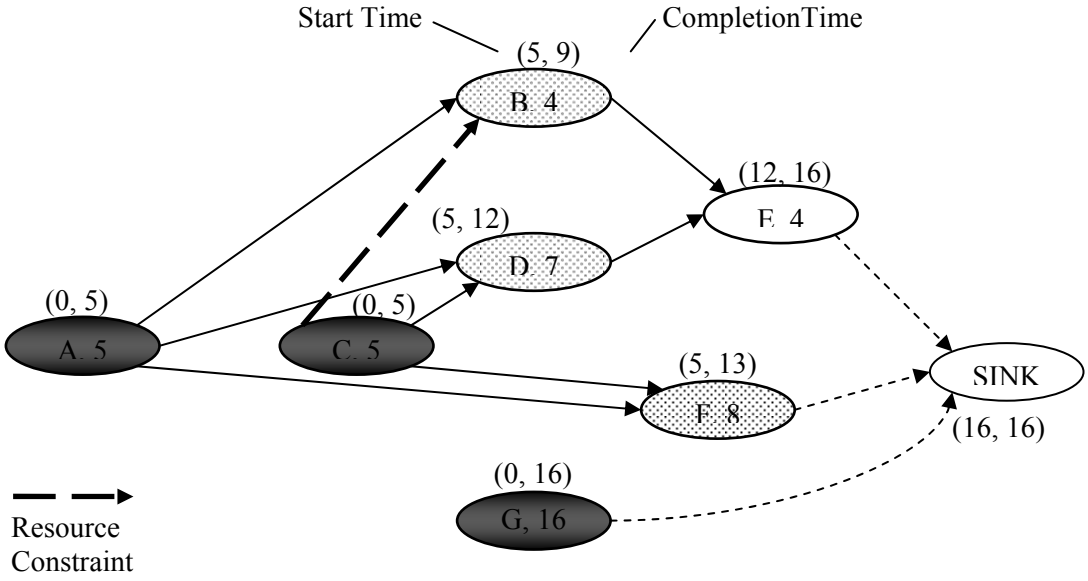


Figure 7.14 Project Network with early start and completion time

New S_j' values for unscheduled activities are;

Activity	B	D	F	E
P_j	4	7	8	4
S_j'	5	5	5	12
C_j'	**9**	**12**	**13**	16

Note, activities B, D and F have earliest start time of 5. Since, Workers 1 and 2 are available at time, t = 5, schedule next activity at time 5. So, TNOW = 5. Schedulable activities are {B, D, and F}. Since, activities B and D have higher ACTIM rank than activity F, schedule activities B and D at time 5. Updated Gantt chart is shown in Figure 7.15.

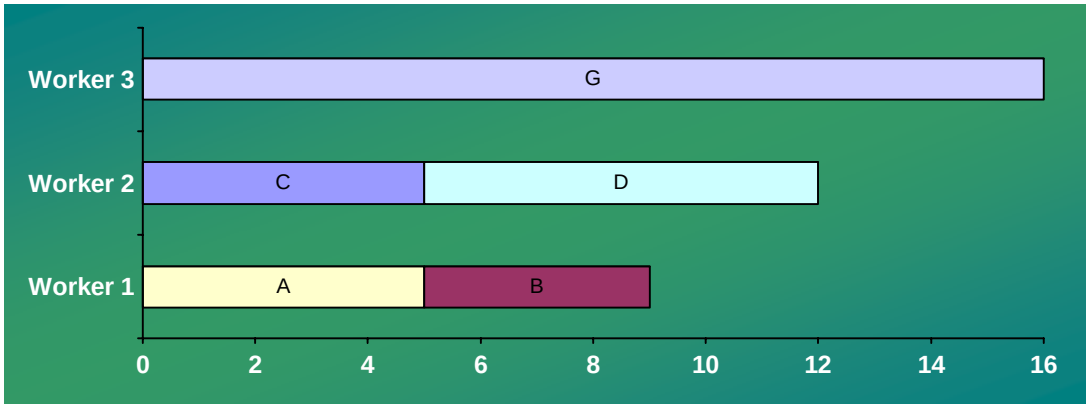


Figure 7.15 Gantt chart for three workers assignment.

The new available times for workers are as follows:

Worker 1	Worker 2	Worker 3
9	**12**	**16**

Note, worker 1 will be free now at time 9. Add arc from B to F in the network diagram and find new S_j' values for unscheduled activities as shown in Figure 7.16.

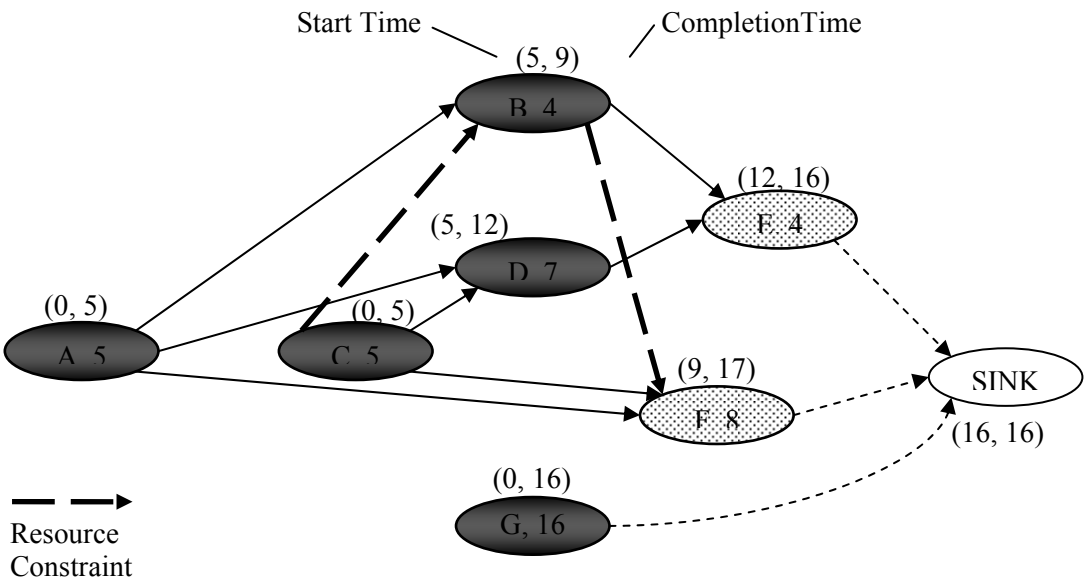


Figure 7.16 Project Network with early start and completion time

New $S_j^{'}$ values for unscheduled activities are;

Schedule activity F at time t = 9 and, engage worker 1. Similarly, schedule activity E at time t = 12, and, engage worker 2. Final schedule is shown in completed Gantt chart below. Note that $C_{max} = 17$.

Activity	F	E
P_j	8	**4**
$S_j^{'}$	9	**12**
$C_j^{'}$	**17**	**16**

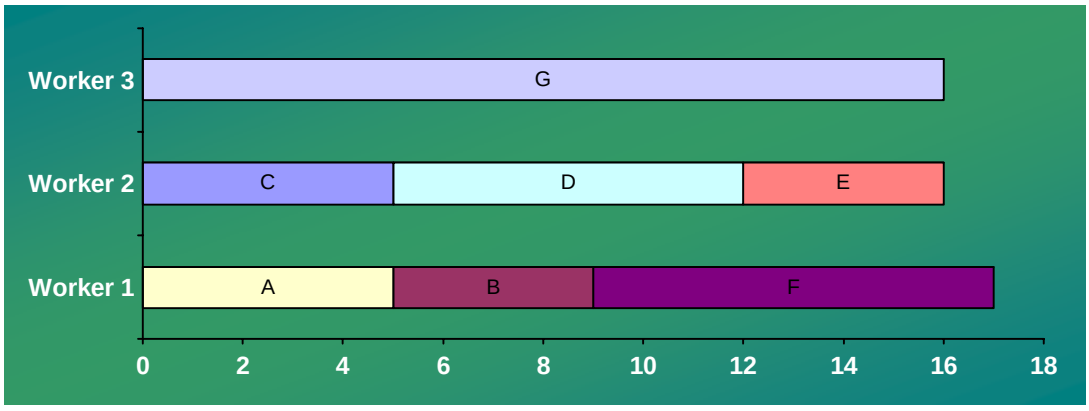


Figure 7.17 Gantt chart for complete project schedule three workers assignment

7.7 MULTIPLE-RESOURCES SINGLE-CAPACITY SCHEDULING

This type of scheduling problem is encountered in project management where more than one type of resource is included in planning. For example; housing construction company will require services of various categories of work force; masons, plumbers, electricians, carpenters, steel fixers and ordinary labors. For this class of problems, it is assumed that quantity of each resource required is exactly one. An algorithm for generating active schedule is presented below to solve this class of problems.

7.2.1 Active Schedule Generation

i. From the project data, find early completion times of all activities (C_j')

ii. Let's define;

β = set of unscheduled activities
k = iteration index
j = activity number
C_j^k = early completion time of j^{th} activity on k^{th} iteration
$\overline{R}$ = Particular resource group.

Determine Δ where

$$\Delta = \min_{j \in \beta} (C_j^k) \qquad k = 0, 1, 2, ...$$

iii. Check for any conflicts.

Let's the activity whose completion time is Δ [from step (ii)] be called j* and, let j* $\in$ $\overline{R}$. For each activity j in resource group $\overline{R}$, compute $C_{j*}^k + p_j$. The activity j must satisfy these properties;

a) $j \in \overline{R} \cap \beta$

b) $j \neq j^*$

If $C_{j*}^k + p_j \leq C_j^k$, then activities j and j* are not in <u>conflict</u>.

iv. If all the unscheduled activities j in resource group $\overline{R}$ are not in <u>conflict</u> with activity j*, then schedule j*. If β is { }, then STOP, otherwise go to step (ii).

v. If there is a conflict between j* and any activity j, then arbitrarily select a conflicting activity and schedule it. Update all completion times. If β is { }, then STOP, otherwise go to step (ii).

Example 7.3

The precedence graph of a four activities is given in Figure 7.18. The activities belong to the following two resource groups.

$R_1 = \{T_1, T_4\}$ $R_2 = \{T_2, T_3\}$

Generate all possible schedules. Indicate active, semi-active and infeasible schedules also.

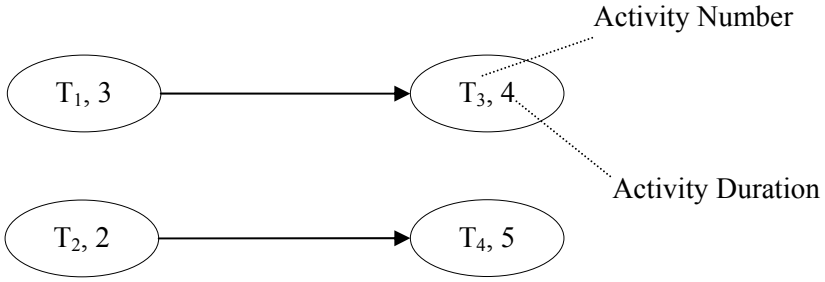


Figure 7.18 Precedence Graph

Generate all possible schedules and, find minimum C_{max} value schedule.

Solution:

The following four schedules are possible.

1: $\begin{Bmatrix} T_1 \rightarrow T_4 \\ T_2 \rightarrow T_3 \end{Bmatrix}$ 2: $\begin{Bmatrix} T_1 \rightarrow T_4 \\ T_3 \rightarrow T_2 \end{Bmatrix}$ 3: $\begin{Bmatrix} T_4 \rightarrow T_1 \\ T_2 \rightarrow T_3 \end{Bmatrix}$ 4: $\begin{Bmatrix} T_4 \rightarrow T_1 \\ T_3 \rightarrow T_2 \end{Bmatrix}$

Precedence graph for schedule 1 is shown in Figure 7.19.

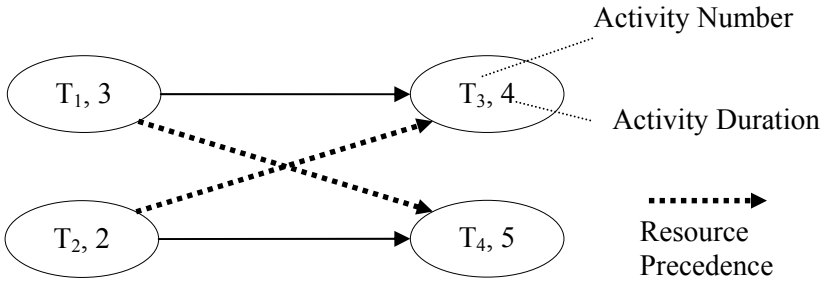


Figure 7.19 Precedence Graph for schedule 1

The Gantt chart for schedule 1 is shown in Figure 7.20. It is an active-schedule. The value of $C_{max} = 8$

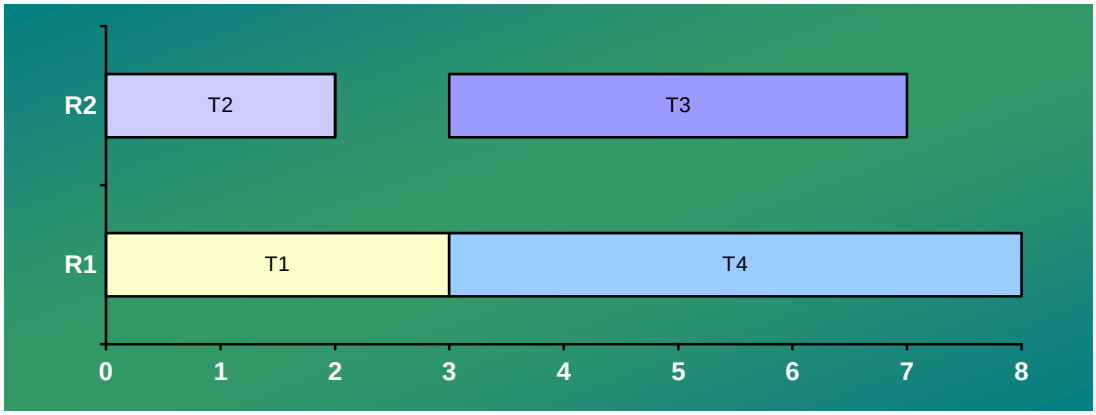


Figure 7.20 Gantt chart for schedule 1

Precedence graph for schedule 2 is shown in Figure 7.21

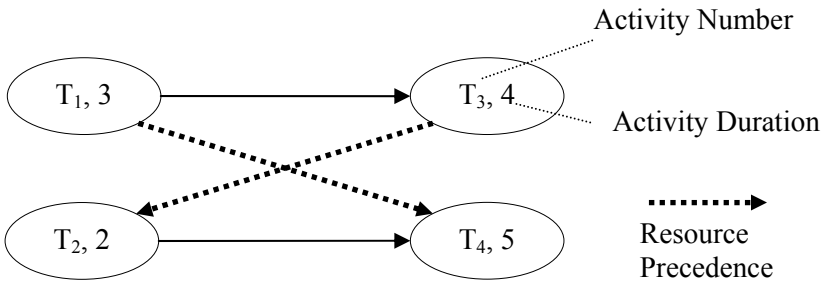


Figure 7.21 Precedence Graph for schedule 2

The Gantt chart for schedule 2 is shown in Figure 7.22. It is semi-active schedule. The value of $C_{max} = 14$

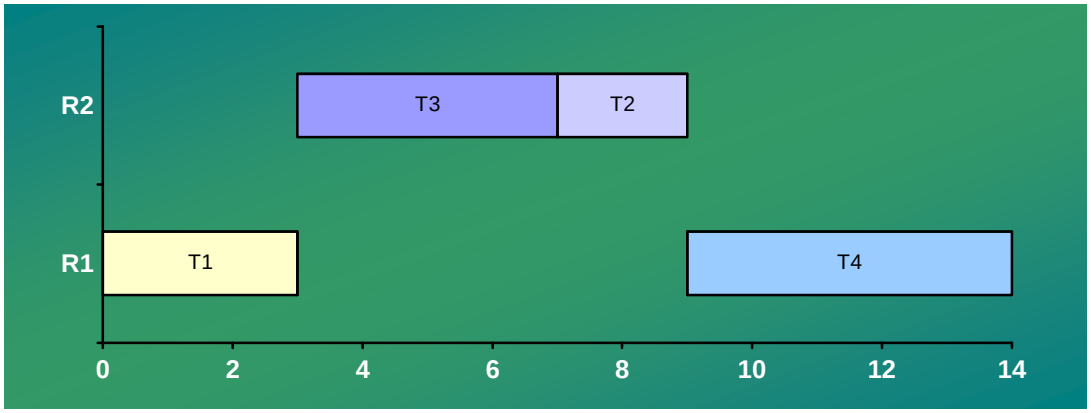


Figure 7.22 Gantt chart for schedule 2

Precedence graph for schedule 3 is shown in Figure 7.23.

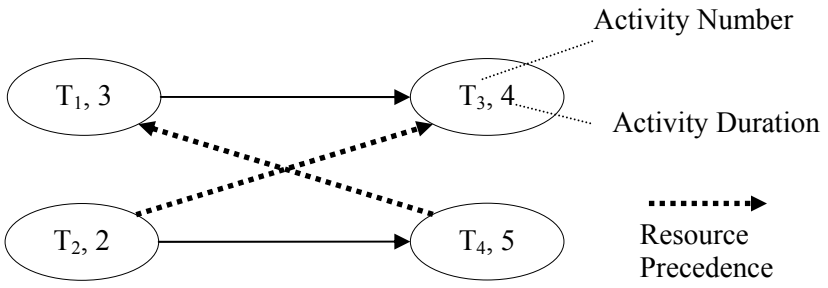


Figure 7.23 Precedence Graph for schedule 3

The Gantt chart for schedule 3 is shown in Figure 7.24. It is semi-active schedule. The value of $C_{max} = 14$

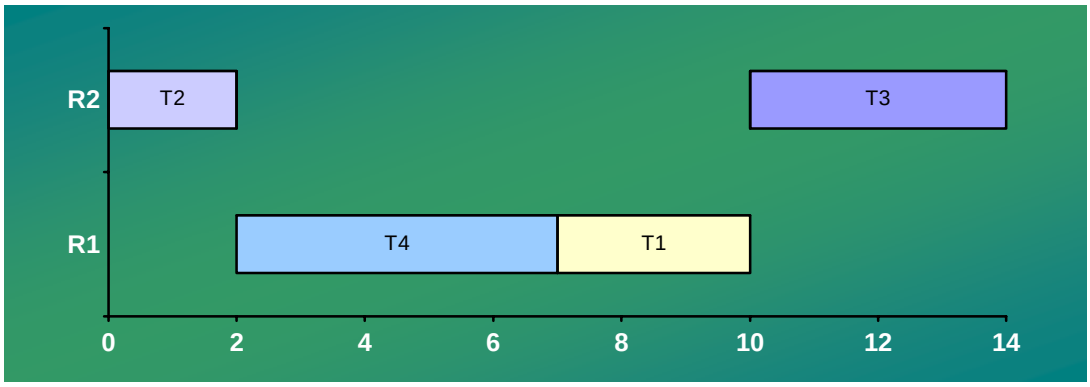


Figure 7.24 Gantt chart for schedule 3

Precedence graph for schedule 4 in figure 7.25 is infeasible because it forms a cycle.

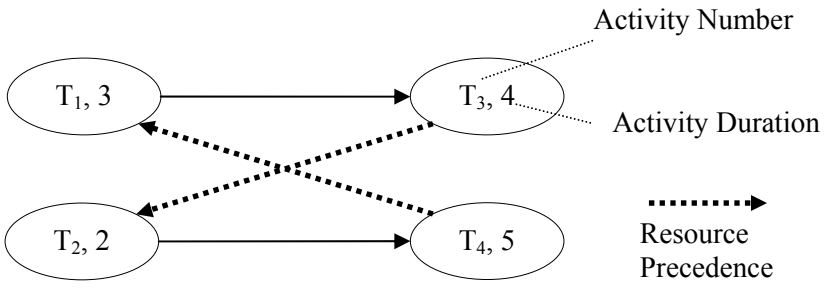


Figure 7.25 Precedence Graph for schedule 3

Example 7.4

Solve Example 7.3 using active schedule generation algorithm.

Solution:

Iteration 1

Step (i) Find early completion time of all activities.

$$C^0 = \begin{bmatrix} 3 & 7 \\ 2 & 7 \end{bmatrix}$$

Step (ii) Minimum completion time is 2; $\Delta = 2 = C^0_{T_2}$

Step (iii) Check conflicts in Resource group 2 for activity T_2 with other activities.

$$C^0_{T_2} + t_{T_3} \le C^0_{T_3} \rightarrow 2 + 4 \le 7 \rightarrow \text{yes}$$ (No conflict between T_2 & T_3)

Step (iv) Schedule activity T_2 t = 0, $S_{T_2} = 0,\ C_{T_2} = 2$

So, scheduled activities set = $\{T_2\}$, $\beta = \{T_2, T_3, T_4\}$

Iteration 2

Step (ii) Select next job with minimum value of C^0_i; $\Delta = 3 = C^0_{T_1}$

Step (iii) Check for conflict between activities T_1 and T_4 in resource group R_1.

$$C^0_{T_1} + t_{T_4} \le C^0_{T_1} \rightarrow 3 + 5 = 8 \le 7 \rightarrow \text{No}$$ (Conflict between T_1 and T_4)

Step (iv) (b) Let's schedule activity T_1 ; $S_{T_1} = 0,\ C_{T_1} = 3$

Scheduled activities set = $\{T_2, T_1\}$, $\beta = \{T_3, T_4\}$

Iteration 3

Add resource constraint arc from T_1 to T_4 and, T_2 to T_3 as shown in Figure 7.26.

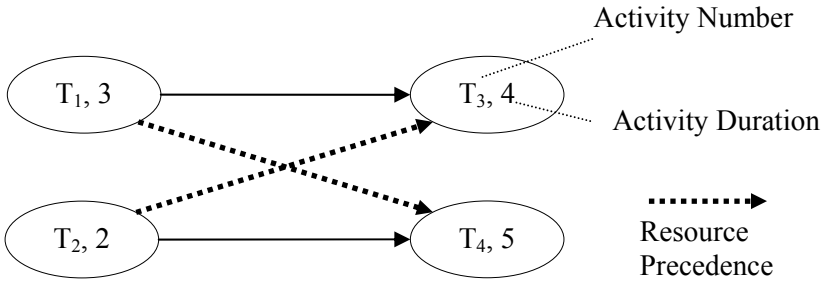


Figure 7.26 Resource constraint arcs between T_1/T_4 and T_2/T_3

Compute new completion times for unscheduled activities

$$C^1 = \begin{bmatrix} \underline{\mathbf{3}} & 7 \\ \underline{\mathbf{2}} & 8 \end{bmatrix}$$

Step (ii) Minimum completion time for unscheduled activities is 7; $\Delta = C^1_{T_3} = 7$. Activity T_3 is the last activity in its group, so schedule T_3; $S_{T_3} = 5,\ C_{T_3} = 7$

Last unscheduled activity is T4; schedule it. $S_{T_4 = 3}, C_{T_4} = 8$.

Comment: Active schedule generation algorithm generates 'good' solution.

Example 7.5

A precedence network with activities and durations is shown in Figure 7.27.

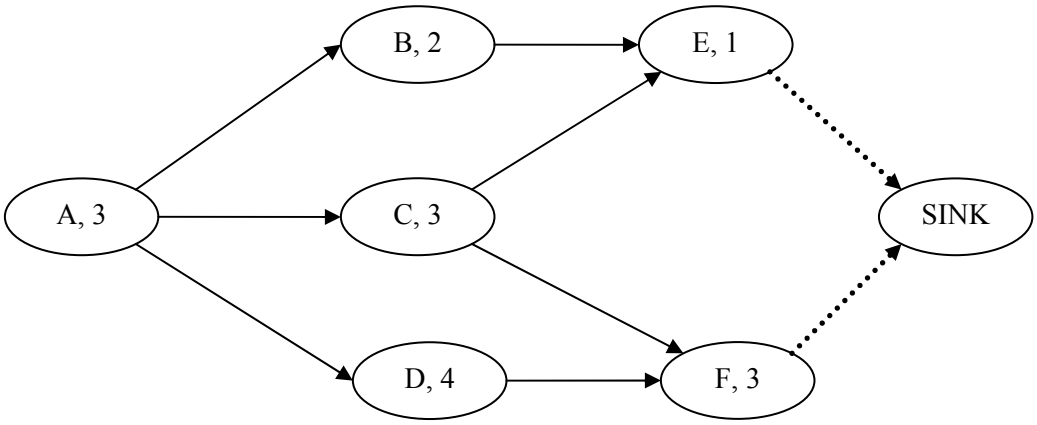


Figure 7.27 Project Network for Example 7.5

The activities belong to two resource groups as follows:

$R_1 = \{A, E, F\}$ $R_2 = \{B, C, D\}$

Generate active schedule using early completion time approach.

Solution:

Iteration 1

Step (i) Find early completion time of all activities.

$$C^0 = \begin{Bmatrix} - & 5 & 7 \\ 3 & 6 & - \\ - & 7 & 10 \end{Bmatrix}$$

Step (ii) Minimum completion time is 3; $\Delta = 3 = C_A{}^0$

Step (iii) check for conflicts in Resource group 1 for activity A with other activities.

$C_A^0 + t_E \le C_E^0 \rightarrow 3 + 1 \le 7 \rightarrow$ yes No conflict between A & E

$C_A^0 + t_F \le C_F^0 \rightarrow 3 + 3 \le 10 \rightarrow$ yes No conflict between A & F

Step (iv) Schedule activity A at t = 0, $S_A = 0$, $C_A = 3$

Scheduled activities set = {A}, $\beta = \{B, C, D, E, F\}$

Iteration 2

Step (ii) Select next job with minimum value of C_i^0 ; $\Delta = 5 = C_B^0$

Step (iii) Check for conflict between jobs B , C and D in resource group R_2.

$$C_B^0 + t_C \le C_C^0 \rightarrow 5 + 3 = 8 \le 6 \rightarrow \text{No}$$

$$C_B^0 + t_D \le C_D^0 \rightarrow 5 + 4 = 9 \le 7 \rightarrow \text{No}$$

} Conflict between B, C, D

Step (iv) (b) Let's schedule activity B ; $S_B = 2$, $C_B = 5$.
Scheduled activities set = {A, B}, β = {C, D, E, F}

Iteration 3

Add resource constraints from B to C and, from B to D (Figure 7.28)

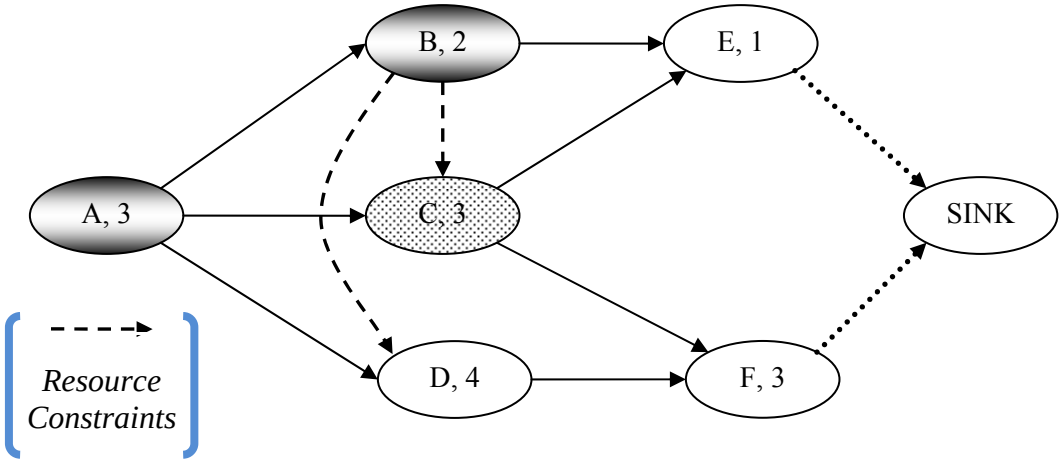


Figure 7.28 Project Network with new arcs emanating from node B

Compute new completion times

$$C^1 = \begin{Bmatrix} - & \underline{\mathbf{5}} & 9 \\ \underline{\mathbf{3}} & 8 & - \\ - & 9 & 12 \end{Bmatrix}$$

Step (ii) Minimum completion time for unscheduled activities is 8; $\Delta = C_c^1 = 8$

Step (iii) Check conflict between unscheduled activities in group 2;

$C_C^1 + t_D \le C_D^1 \rightarrow 8 + 4 = 12 \le 8 \rightarrow \text{No}$ {Conflict between C & D}.

Step (iv) (b) Schedule activity C arbitrarily; $S_C = 5$, $C_C = 8$
Scheduled activities set = {A, B, C}, β = {D, E, F}

Iteration 4

Revise the project network for unscheduled activities. Add a resource constraint from C to D. Remove the constraint from B to D (Figure 7.29). Compute new completion times for unscheduled activities.

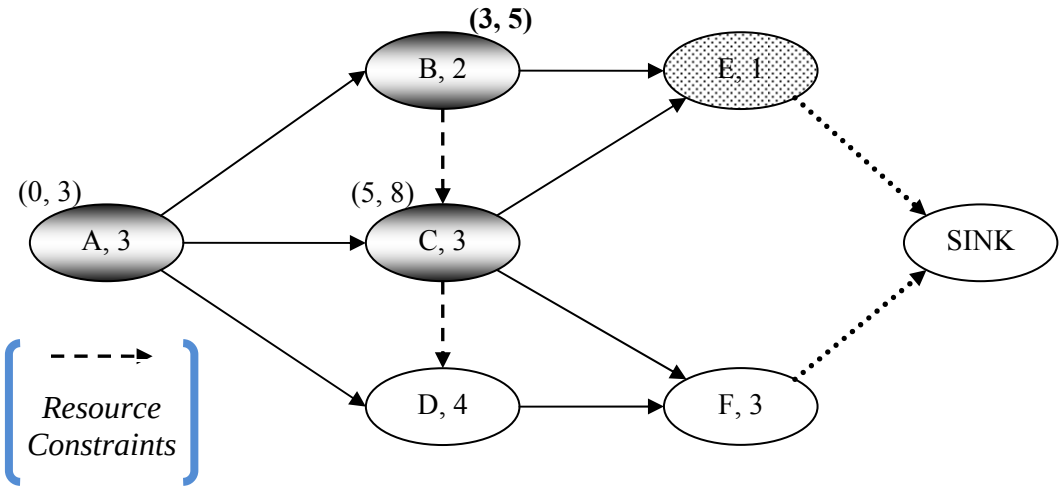


Figure 7.29 Project Network with new arcs emanating from node C

New completion times for unscheduled activities are;

$$C^2 = \begin{Bmatrix} - & \underline{\mathbf{5}} & 9 \\ \underline{\mathbf{3}} & \underline{\mathbf{8}} & - \\ - & 12 & 15 \end{Bmatrix}$$

Step (ii) Minimum completion time from among unscheduled activities is 9; $\Delta = C_E^2 = 9$ for activity E.

Step (iii) Activity E belongs to resource group 1. Activity F is the other unscheduled activity in this group. Check conflict between activities E and F as follows:

$$C_E^2 + t_F \le C_F^2 \rightarrow 9 + 3 = 12 \le 15 \rightarrow \text{No}$$ { No conflict between E & F }

Step (iv) (a) Schedule activity E; $S_E = 8$, $C_E = 9$

Scheduled activities list = { A , B , C , E }, β = { D , F }

Iteration 5

Step (ii) Find next lower minimum C_j value in C^2.

Next lower minimum value in C^2 is 9. $\Delta = C_D^2 = 12$. Since, activity D is the last activity in resource group 2, schedule D; $S_D = 8$, $C_D = 12$.

Last remaining activity is F , schedule F, $S_F = 12$, $C_F = 15$

Gantt chart for resources R_1 and R_2 is shown in Figure 7.30

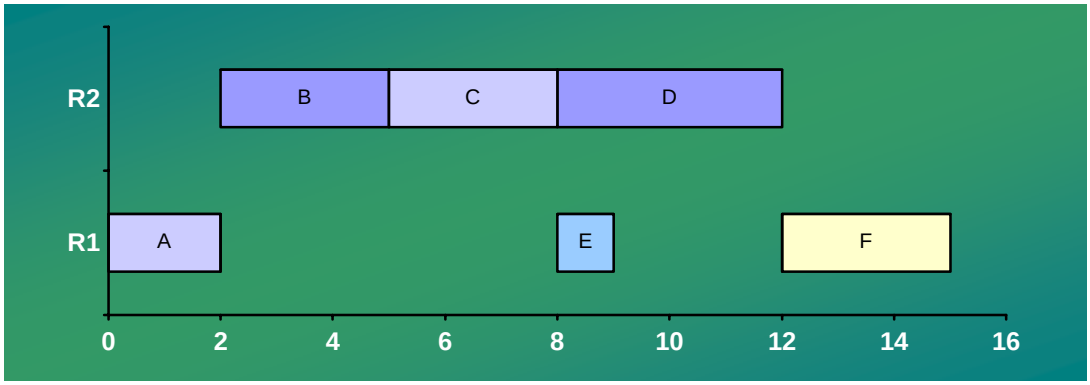


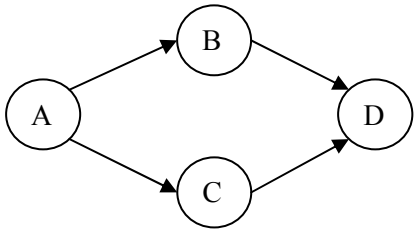
Figure 7.30 Final Gantt chart

7.8 MULTIPLE-RESOURCES MULTI-CAPACITY SCHEDULING

Often, project managers are asked to execute large projects where resources are of diverse variety. Various types of resources including man power, equipment, machine, tools, transportation vehicles, information processing resources, expert knowledge and supervisory staff are required with different levels at different time. This type of scheduling problem presents real world scheduling scenario. The optimal solution for this problem is very difficult. The solution methodology aims at minimizing C_{max} while keeping resource utilization within feasible bounds.

Example 7.6

Consider a 4-activity network as follows:



Activities process times and resource requirements are given in Table 7.4. Find C_{max} and draw Gantt chart.

Table 7.4 Resources data For Example 7.6

Activity (j)	p_j	R_α	R_β
A	5	2α	
B	6		3β
C	7	4α	β
D	3	2α	
Maximum Resource		5α	4β

Solution:

Apply CPM technique and find S_j', C_j' of all activities as shown in Figure 7.31.

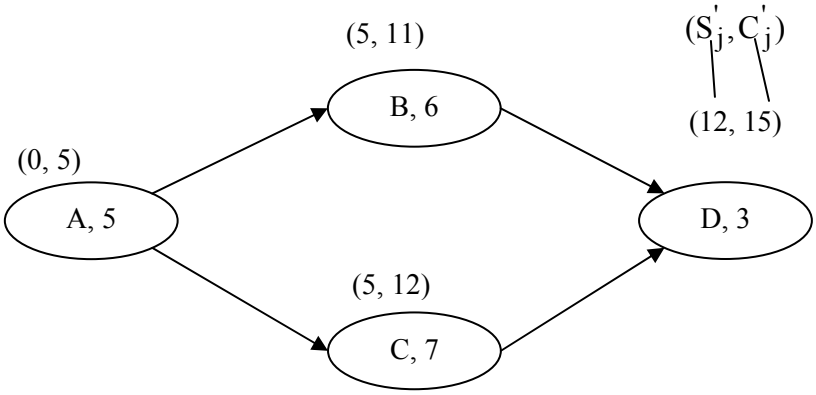


Figure 7.31 Project Network with early start and completion time

The earliest start and completion times of the activities are shown in Table 7.5.

Activity	pj	S_j'	C_j'
A	5	0	5
B	6	5	11
C	7	5	12
D	3	12	15

Table 7.5 Early start and completion Times

At beginning, $\beta = \{A, B, C, D\}$,
$R_\alpha = \{A, C, D\}$
$R_\beta = \{B, C\}$

Activity A is schedulable at time, t = 0. The resource requirements for activity A are shown in Table 7.6.

Table 7.6 Resource requirements for activity A

			Resource Required		Resource Available	
Activity	p_j	S_j'	R_α	R_β	R_α	R_β
A	5	0	2α	---	5α	4β

The resources for activity A are available. Schedule activity A at time, t = 0. The Gantt chart is shown in Figure 7.32.

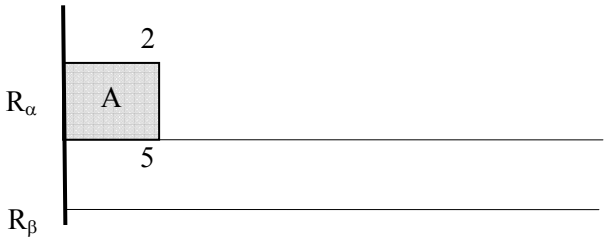


Figure 7.32 Initial Gantt chart

Updated information on unscheduled activities is;

$\beta = \{B, C, D\}$,

$R_\alpha \cap \beta = \{C, D\}$,

$R_\beta \cap \beta = \{B, C\}$

Activities B and C can be started as early as at time, t = 5 as shown in Table 7.7 below.

Table 7.7 Resource data for activities B & C

			Resource Required		Resource Available	
Activity	p_j	S'_j	R_α	R_β	R_α	R_β
B	6	5	---	3β	5α	4β
C	7	5	4α	β		

Resources required for activities B and C are available at time, t =5. Schedule activities B and C at time, t = 5. Update Gantt chart as shown in Figure 7.33.

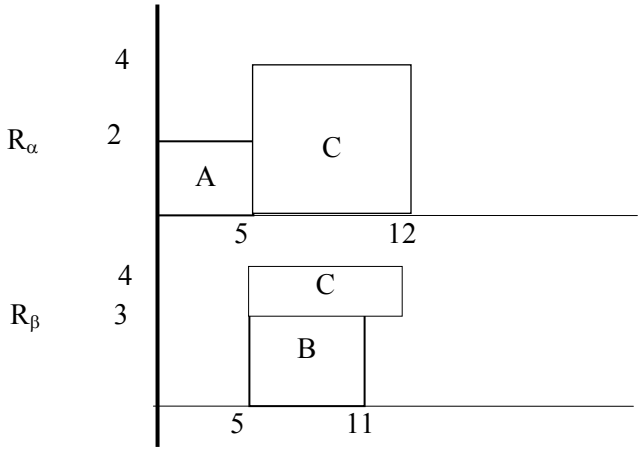


Figure 7.33 Gantt chart with activities A, B and C scheduled.

Update information about unscheduled activities

$\beta = \{D\}$,
$R_\alpha \cap \beta = \{D\}$,
$R_\beta \cap \beta = \emptyset$

The remaining activity is D. Activity D has earliest start time at t =12 (Table 7.8).

Table 7.8 Resource data for activity D

			Resource Required		Resource Available	
Activity	p_j	S_j'	α	β	α	β
D	3	12	---	4β	5α	4β

Resources required for activity D are available at time, t = 12. Schedule activity D at time, t = 12. The updated Gantt chart is shown in Figure 7.34.

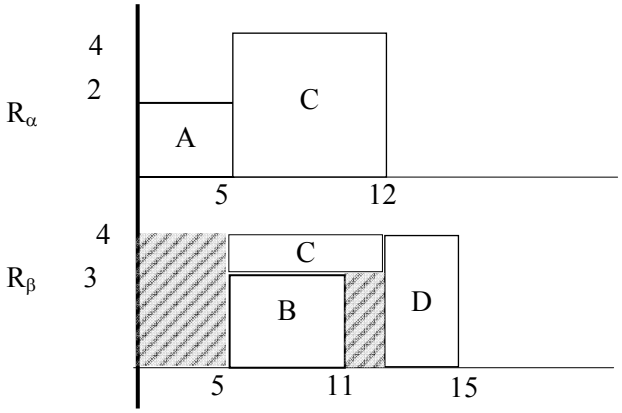


Figure 7.34 Gantt chart for complete schedule.

7.9 PRIORITY BOUNDS FOR SCHEDULING

When more than one activity competes for scheduling on resource, there is scheduling conflict due to resource scarcity. One activity among the competing set of activities is to be assigned to the resource. Selection of an activity will be either done randomly or using some type of criteria. Three selection criteria are presented here.

7.9.1 Precedence Based Bound: $\beta_1(T_j)$

A first criterion uses precedence-based bound approach. This bound represents the length of the critical path in project network if activity T_j is scheduled. An example is cited here to demonstrate the bound calculations.

Example 7.7

A 6-activity project network is shown in Figure 7.35.

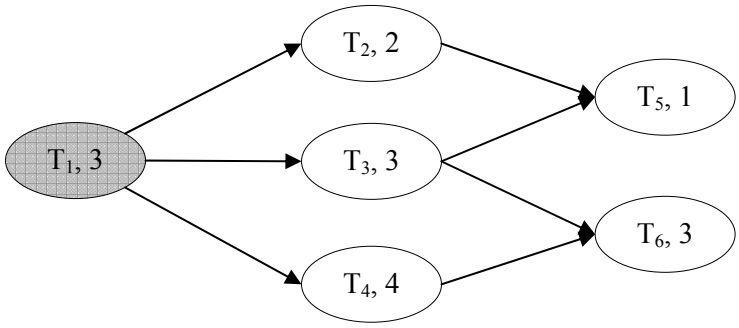


Figure 7.35 Precedence Network with T_1 already scheduled

The activities belong to resource groups as under:

$R_1 = \{T_1, T_5, T_6\}$ $R_2 = \{T_2, T_3, T_4\}$

Find the bounds $\beta_1(T_2)$, $\beta_1(T_3)$ and $\beta_1(T_4)$ for the activities in resource group R_2. Assume activity T_1 is already scheduled.

Solution:

Let β is the set of unscheduled activities. Then, $\beta = \{T_2, T_3, T_4, T_5, T_6\}$

To find $\beta_1(T_2)$, add arcs from T_2 to T_3 and from T_2 to T_4; since activities T_2 ,

T_3 and T_4 belong to the same group. The updated network is shown in Figure 7.36.

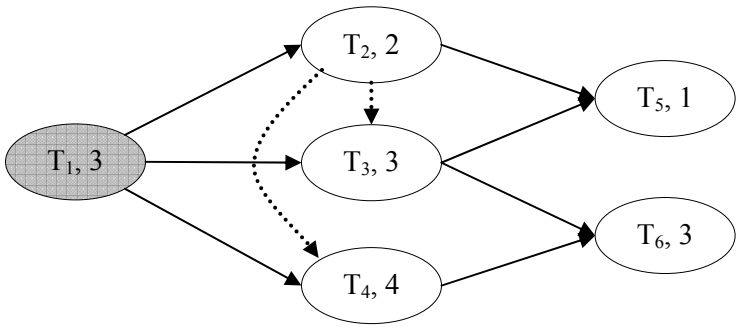


Figure 7.36 Precedence Network with arcs from T_2 to T_3 and from T_2

Critical path length calculations are shown with earliest start and completion times in Table 7.9.

Activity (j)	Early Start Time S_j'	Early Finish Time C_j'	Activity Significance
T_1	0	3	Critical
T_2	3	5	Critical
T_3	5	8	
T_4	5	9	Critical
T_5	8	9	
T_6	9	12	Critical

Table 7.9 Early start and completion time of activities

Hence, $\beta_1(T_2) = 12$. Similarly, it can be proved that $\beta_1(T_3) = 13$, and, $\beta_1(T_4)=13$

7.9.2 Resource Based Bound

This bound is used to resolve scheduling conflict using characteristics of the unscheduled activities in the resource groupings. The bound is designated as β_2, and expressed as under;

$$\beta_2(T_j) = \max_k \left\{ \hat{S}_k + \sum_{T_k \in R_k \cap \beta} p_{T_k} \right\}$$

Where,

$\hat{S}_k$ = Early start time of any unscheduled activity on resource type k. (k = 1, 2, ...)

β = Set of unscheduled activities.

T_j = Activity whose lower bound is to be computed.

T_k = Activity belonging to resource type R_K and still unscheduled $\in$ (β). $\{T_k \neq T_j\}$.

p_{T_k} = Duration of activity T_k.

Example 7.8

For the precedence network shown below, activity T_1 is already scheduled. Compute $\beta_2(T_2)$. Resource groupings are as under:

$R_1 = \{T_1, T_5, T_6\}$ $R_2 = \{T_2, T_3, T_4\}$

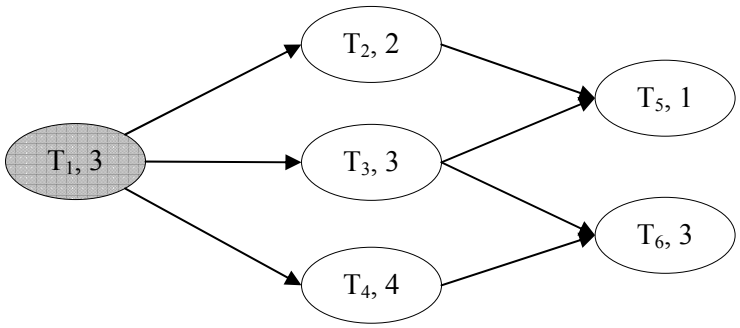


Figure 7.37 Precedence Network with T_1 already scheduled

Solution:

Since activity T_1 is scheduled already, $\beta = \{T_2, T_3, T_4, T_5, T_6\}$. Activity T_2 belongs to group R_2. Add resource constraint arcs from T_2 to T_3 and T_2 to T_4 as shown in Figure 7.38

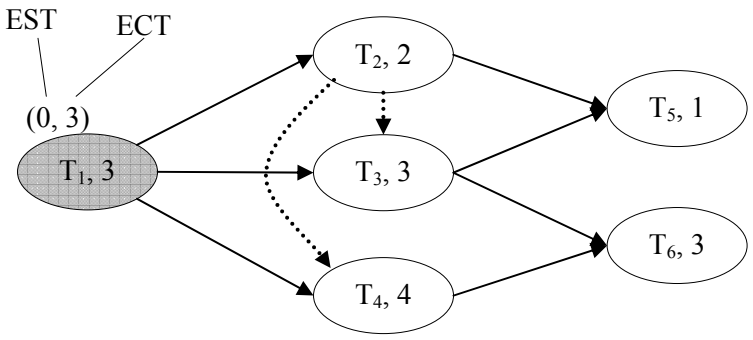


Figure 7.38 Precedence Network with arcs from T_2 to T_3 T_2 to T_4

$\hat{S}_1$ = Early start time of unscheduled activity in Resource type R_1 from Figure 7.38 is;

Activity	$\hat{S}_1 =$
T_5	8
T_6	8

$$\hat{S}_1 \cong 8$$

$\hat{S}_2$ = Early start time of unscheduled activity in Resource type R_2 from Figure 7.38 is;

Activity	$\hat{S}_2 =$
T_3	5
T_4	5

$$\hat{S}_2 \cong 5$$

Hence,

$$\beta_2(T_2) \cong \max\left\{\begin{array}{l}\hat{S}_1 + p_5 + p_6 = 8+1+3 = 12 \\ \hat{S}_2 + p_3 + p_4 = 5+3+4 = 12\end{array}\right\} = 12$$

Example 7.9

For the precedence network shown in Fig. 7.37, compute $\beta_2(T_3)$. Resource groupings are as under:

$R_1 = \{T_1, T_5, T_6\}$ $R_2 = \{T_2, T_3, T_4\}$

Solution:

Since activity T_1 is scheduled already, $\beta = \{T_2, T_3, T_4, T_5, T_6\}$. Activity T_3 belongs to group R_2. Add resource constraint arcs from T_3 to T_2 and T_3 to T_4 as shown in Figure 7.39.

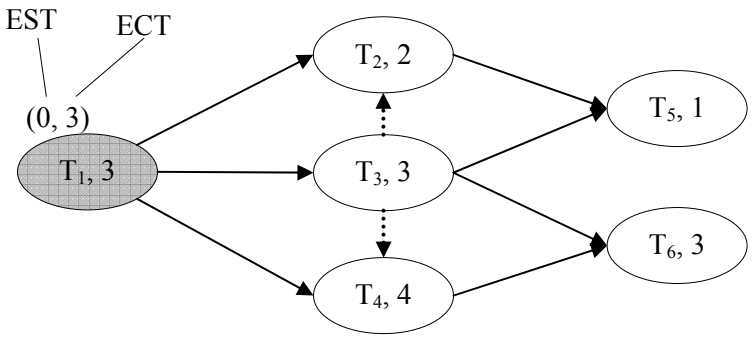


Figure 7.39 Precedence Network with arcs from T_3 to T_2 and from T_3 to T_4

$\hat{S}_1$ = Early start time of unscheduled activity in Resource type R_1 from Figure 7.39 is;

Activity	$\hat{S}_1 =$
T_5	8
T_6	10

$$\hat{S}_1 \cong \min(8, 10) = 8$$

$\hat{S}_2$ = Early start time of unscheduled activity in Resource type R_2 from Figure 7.39 is;

Activity	$\hat{S}_2 =$
T_2	6
T_4	6

$$\hat{S}_2 \cong 6$$

$$\text{Hence, } \beta_2(T_3) \cong \max\left\{\begin{matrix}\hat{S}_1 + p_5 + p_6 = 8+1+3 = 12 \\ \hat{S}_2 + p_2 + p_4 = 6+2+4 = 12\end{matrix}\right\} = 12$$

Example 7.10

For the precedence network shown in Fig 7.37, compute $\beta_2(T_4)$. Resource groupings are as under:

$R_1 = \{T_1, T_5, T_6\}$ $R_2 = \{T_2, T_3, T_4\}$

Solution:

Since activity T_1 is scheduled already, $\beta = \{T2, T_3, T_4, T_5, T_6\}$. Activity T_4 belongs to group R_2. Add resource constraint arcs from T_4 to T_2 and T_4 to T_3 as shown in Figure 7.40.

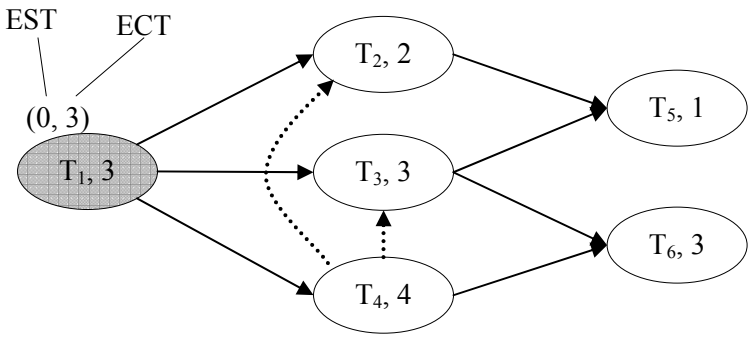


Figure 7.40 Precedence Network with arcs from T_4 to T_2 and from T_4 to T_3

$\hat{S}_1$ = Early start time of unscheduled activity in Resource type R_1 from Figure 7.40 is;

$$\hat{S}_1 \cong 10$$

Activity	$\hat{S}_1 =$
T_5	10
T_6	10

$\hat{S}_2$ = Early start time of unscheduled activity in Resource type R_2 from Figure 7.40 is;

$$\hat{S}_2 \cong 7$$

Activity	$\hat{S}_2 =$
T_2	7
T_3	7

$$\beta_2(T_4) \cong \max\left\{\begin{matrix}\hat{S}_1 + p_5 + p_6 = 10+1+3 = 14 \\ \hat{S}_2 + p_2 + p_3 = 7+2+3 = 12\end{matrix}\right\} = 14$$

The bound β_2 for three activities in resource group R_2 are shown in the tree in Figure 7.41. Hence, either activity T_2 or T_3 can be selected for scheduling.

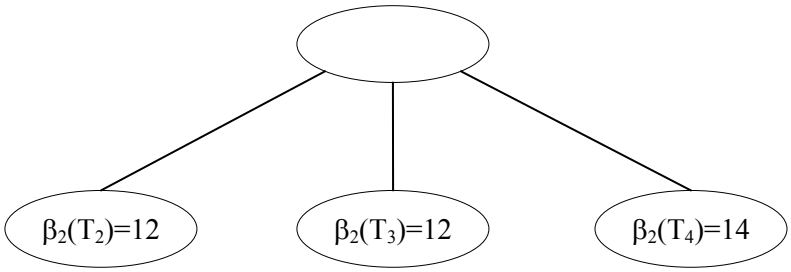


Figure 7.41 Resource based bound β_2 values for R_2 group activities

7.9.3 Hybrid Bound

It is a combination of resource as well as precedence based bounds and, is denoted by $\beta_3(T_j)$. To compute $\beta_3(Tj)$, proceed as follows:

Calculate π_j for each activity T_j,

Where,

$\pi_j \cong$ Length of the longest path in the sub network consisting of all successors of T_j

Order the unscheduled activities in order of non-decreasing values of π in each resource grouping R_k such that

$$R_k \cap \beta \neq \phi$$

Iteratively, calculate λ_j;

$$\lambda_j = \max(\lambda_{j-1} + \pi_{[j]}) + p_{[j]} \quad ; \quad j = 1, 2, ..., n_k$$

Where,

$p_{[j]}$ = duration of activity j

$[j]$ = sequence position of activity j in Resource grouping R_k

n_k = no of unscheduled activities in Resource grouping R_k $[R_k \cap \beta]$

Let the last λ_j computed be equal to W_k,

i.e.; $W_k \equiv \lambda_{n_k}$

Then, $$\beta_3(T_j) = \max_k \left\{ \hat{S}_k + W_k \right\}$$

Example 7.11

For the precedence network shown in Figure 7.42, activity T_1 is already scheduled. Compute $\beta_3(T_2)$. Resource groupings are as under:

$R_1 = \{T_1, T_5, T_6\}$ $R_2 = \{T_2, T_3, T_4\}$

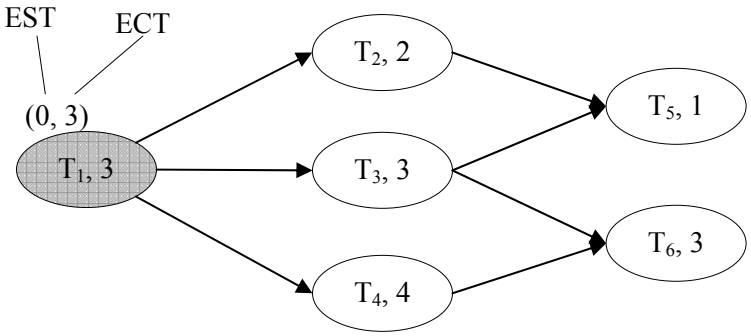


Figure 7.42 Precedence Network with T_1 already scheduled

Solution

Since activity T_1 is scheduled already, and, activity T_2 is being scheduled, $\beta = \{T_3, T_4, T_5, T_6\}$.

$R_1 \cap \beta = \{T_5, T_6\}$, $n_1 = 2$

$R_2 \cap \beta = \{T_3, T_4\}$, $n_2 = 2$

Compute π_j's

$\pi_5 = 0$, $\pi_6 = 0$, $\pi_4 = 3$,

$\pi_3 = 3$, $\pi_2 = 1$, $\pi_1 = 7$

Order the unscheduled activities in R_1 and R_2 according to non-decreasing values of corresponding π_j's.

Hence, $R_1 \cap \beta = \{T_5, T_6\}$, and, $R_2 \cap \beta = \{T_3, T_4\}$.

Calculate W_1 for $R_1 \cap \beta$;

$$\lambda_1 = \max\{\lambda_0 + \pi_5\} + p_5 = \max\{0, 0\} + 1 = 5$$

$$\lambda_2 = \max\{\lambda_1 + \pi_6\} + p_6 = \max\{1, 0\} + 3 = 4$$

$$\text{So, } W_1 \equiv \lambda_2 = 4$$

Calculate W_2 for $R_2 \cap \beta$;

$$\lambda_1 = \max\{\lambda_0 + \pi_3\} + p_3 = \max\{0, 3\} + 3 = 6$$

$$\lambda_2 = \max\{\lambda_0 + \pi_4\} + p_4 = \max\{6, 3\} + 4 = 10$$

$$\text{So, } W_2 \equiv \lambda_2 = 10$$

To find $\hat{S}_k$ values, add resource constraint arcs in the network. Since, activity T_2 belongs to group R_2, add resource constraint arcs from T_2 to T_3 and T_2 to T_4 as shown in Figure 7.43.

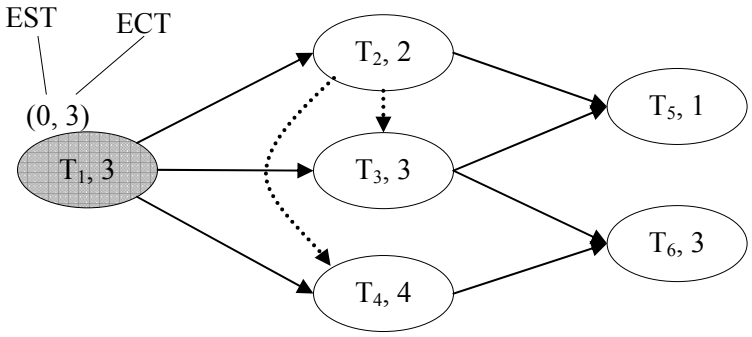


Figure 7.43 Precedence Network with arcs from T_2 to T_3 and from T_2 to T_4

$\hat{S}_1$ = Early start time of unscheduled activity in Resource type R_1 from Figure 7.43 is;

$$\hat{S}_1 \cong 8$$

Activity	$\hat{S}_1$ =
T_5	8
T_6	8

$\hat{S}_2$ = Early start time of unscheduled activity in Resource type R_2 from Figure 7.43 is;

$$\hat{S}_2 \cong 5$$

Activity	$\hat{S}_2$ =
T_3	5
T_4	5

Compute $\beta_3(T_2)$ by formula;

$$\beta_3(T_2) = \max_k \left\{ \hat{S}_k + W_k \right\}$$

$$= \max \begin{Bmatrix} \hat{S}_1 + W_1 = 8 + 4 = 12 \\ \hat{S}_2 + W_2 = 5 + 10 = 15 \end{Bmatrix} = 15$$

Example 7.12

A 9-activities project network is shown in Figure 7.44.

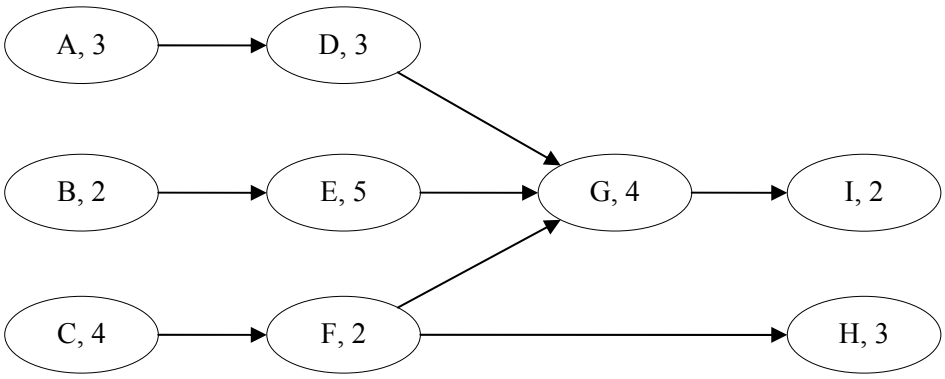


Figure 7.44 Precedence Network for Example 8.13

Resource groupings are as under:

$R_1 = \{A, B, G\}$ $R_2 = \{D, E, C\}$ $R_3 = \{F, H, I\}$

At present, all activities are unscheduled. Suppose activity B is to be scheduled. Compute $\beta_3(B)$. Show all your calculations.

Solution:

Since activity B is being scheduled, $\beta = \{ A , C , D , E , F, G , H , I \}$.

$R_1 \cap \beta = \{A, G\}, n_1 = 2$
$R_2 \cap \beta = \{D, E, C\}, n_2 = 3$
$R_3 \cap \beta = \{F, H, I\}, n_3 = 3$

Compute π_j's

$\pi_A = 9,$ $\pi_B = 9,$ $\pi_C = 8,$
$\pi_D = 6,$ $\pi_E = 6,$ $\pi_F = 6$
$\pi_G = 2,$ $\pi_H = 0,$ $\pi_I = 0$

Order the unscheduled activities in R_1, R_2 and R_3 according to non-decreasing values of corresponding π_j's.

Hence,

$R_1 \cap \beta = \{G, A\}$, and,
$R_2 \cap \beta = \{D, E, C\}$, and,
$R_3 \cap \beta = \{H, I, F\}$

Calculate W_1 for $R_1 \cap \beta = \{G, A\}$

$$\lambda_1 = \max\{\lambda_0, \pi_G\} + p_G = \max\{0\ , 2\} + 4 = 6$$
$$\lambda_2 = \max\{\lambda_1, \pi_A\} + p_A = \max\{6\ , 9\} + 3 = 12$$

So,

$$W_1 \equiv \lambda_2 = 12$$

Calculate W_2 for $R_2 \cap \beta = \{D, E, C\}$

$$\lambda_1 = \max\{\lambda_0, \pi_D\} + p_D = \max\{0,\ 6\} + 3 = 9$$
$$\lambda_2 = \max\{\lambda_1, \pi_E\} + p_E = \max\{9,\ 6\} + 5 = 14$$
$$\lambda_3 = \max\{\lambda_2, \pi_C\} + p_C = \max\{14,\ 8\} + 4 = 18$$

So,

$$W_2 \equiv \lambda_3 = 18$$

Calculate W_3 for $R_3 \cap \beta = \{H, I, F\}$;

$$\lambda_1 = \max\{\lambda_0, \pi_H\} + p_H = \max\{0,\ 0\} + 3 = 3$$
$$\lambda_2 = \max\{\lambda_1, \pi_I\} + p_I = \max\{3,\ 0\} + 2 = 5$$
$$\lambda_3 = \max\{\lambda_2, \pi_F\} + p_F = \max\{5,\ 6\} + 2 = 8$$

So,

$$W_3 \equiv \lambda_3 = 8$$

If activity B is scheduled first, add a resource constraint arc from activity B to activity A. The early start times of all activities are shown in Figure 7.45.

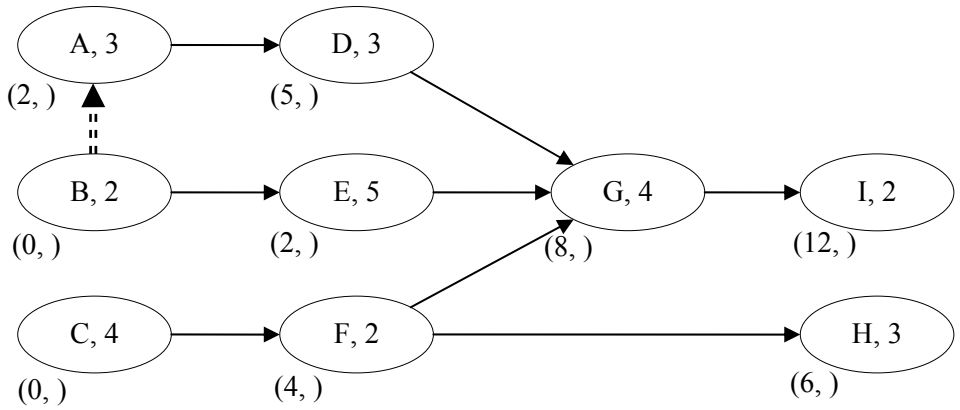


Figure 7.45 Precedence Network with resource arc from B to A

Early start times for resource groupings are;

$$\hat{S}_1 = \min \left\{ \hat{S}_A , \hat{S}_G \right\} = \min \{2, 8\} = 2$$

$$\hat{S}_2 = \min \left\{ \hat{S}_D , \hat{S}_E , \hat{S}_C \right\} = \min \{5 , 2, 0\} = 0$$

$$\hat{S}_3 = \min \left\{ \hat{S}_H , \hat{S}_I , \hat{S}_F \right\} = \min \{6 , 12, 4\} = 4$$

$$\text{So,} \quad \beta_3(B) = \max_k \left\{ \hat{S}_k + W_k \right\}$$

$$= \max \left[W_1 + \hat{S}_1 , \; W_2 + \hat{S}_2 , \; W_3 + \hat{S}_3 \right]$$

$$= \max \left[12 + 2 \; , \; 18 + 0 \; , \; 8 + 4 \right]$$

$$= \max \{ 14 , 18 , 12 \} = 18$$

So,

$\beta_3(B) = 18$

7.10 RESOURCE LEVELING

Resource leveling is a commonly used project planning technique to avoid extraordinary demands or excessive fluctuations in labor and plant resources required for a project, which could otherwise lead to a drop in productivity or an increase in production cost. After assigning resources, it is likely that at certain times there will be more work assigned than there are resources available. The main purpose of resource leveling is to create a smoother distribution of resource usage and reduce over-allocation of resources. Leveling requires delaying tasks until resources are available, thus enabling the project to be finished, though often resulting in a later project finish date.

Resource leveling aims to:

- Examine resource requirements during specific periods of the project
- Minimize the variation in resource demand from period to period during project execution.

Resource Graph

The resource graph depicts the usage rate of that resource as a histogram. A sample resource graph with sudden changes in resource requirements on consecutive days is shown in Figure 7.46. It represents workers demand on daily basis over one week time.

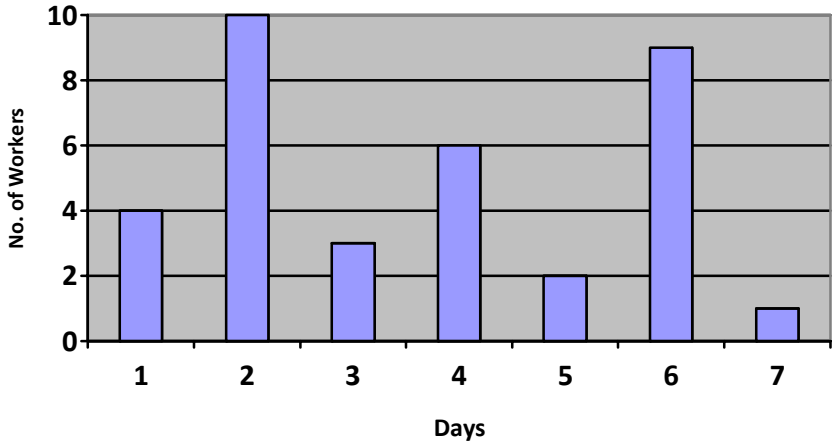


Figure 7.46 Unleveled Schedule

Peak worker requirement is on day 2 when 10 workers are demanded. The organization has maximum of 7 workers. There is abrupt change in worker's requirement for successive days with variations in resource utilization as given in the following Table 7.10.

Table 7.10 Variation in the requirement of resource

Day→	1	2	3	4	5	6	7
Workers	4	10	3	6	2	9	1
Change(+/-)	-3	+6	-7	+3	-4	+7	-8

The resource leveling effort intends to *smooth* out the demand of a resource over time. The effect is to reduce the size of the peaks and troughs on the resource graphs (histograms) - or as a more practical interpretation to improve the efficiency of resource utilization. It can be achieved by a variety of different techniques; (i) adjusting the timing of activities within their float, (ii) by the moving of allocated resources between activities, (iii) by more radical approaches - such as the hiring in of contract staff and the pre-production of certain products. Both the float and the critical path will need continual attention as the resource leveling process proceeds.

Benefits of leveling (smoothing) will include; (i) reduce management overheads of sudden changes, (ii) improves morale of human resources, (iii) implications for smooth cost profiles. A sample resource graph after resource leveling is shown in Figure 7.47. It presents minor changes in resource usage between days with maximum resource requirement within available limits of the resources.

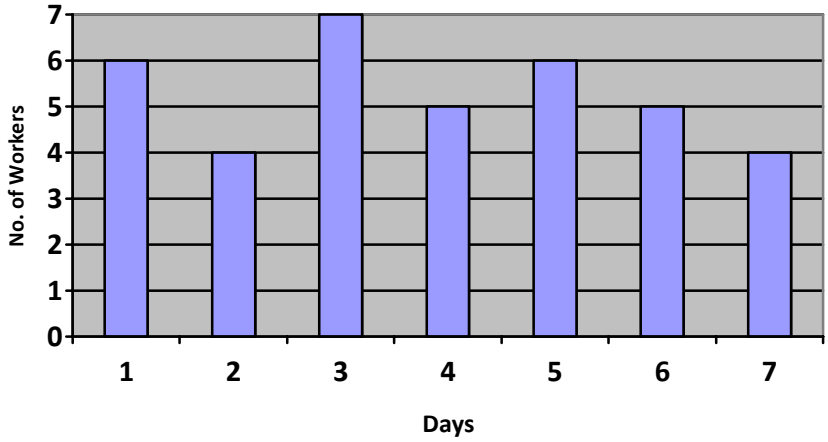


Figure 7.47 Leveled Resource Schedule

In performing resource leveling, many planners or managers would adopt standard heuristic approaches to obtain an acceptable solution. This is because mathematical methods are only considered suitable for small to medium networks due to the combinatorial non-deterministic nature of the problem. The leveling of multiple resources is also dominated by the chosen heuristic methods, e.g. whether by leveling multiple resources in series or through combined resource leveling. Although heuristic approaches are easy to understand, they are problem-dependent. Hence, it is difficult to guarantee that an optimal solution can be achieved.

One commonly used heuristic technique starts with constructing a Gantt chart using early start and completion time of all activities. A resource graph is constructed from the Gantt chart reflecting the distribution of resource usage over time. If sudden crests and troughs are found on the graph, then slack activities are rescheduled to shift the resource usage from high peak areas to low trough valleys over the resource graph. Sometimes, some activities are split to produce smooth usage of the resources. An example will demonstrate the approach.

Example 7.13

A project network is shown in Figure 7.48. Each activity requires Workers for its execution. The resource requirements for each activity are given below.

Activity	A	B	C	D	E	F	G
No of Workers	4	6	4	4	2	2	3

a) Generate project schedule. Draw Gantt chart and develop Resource Graph.
b) Is the resource graph leveled? (Why or why not!)
c) If not, apply resource leveling heuristic and obtain leveled schedule.

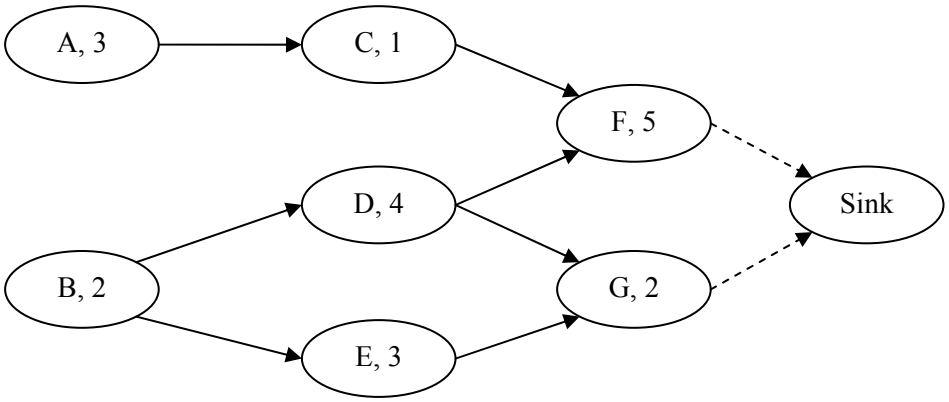


Figure 7.48 Project network for Example 7.13.

Solution

The computations of earliest C_j' and latest completion times (C_j'') are shown in the table 7.11

Activity	p_j	S_j	C_j'	S_j''	C_j''	Critical	Slack
A	3	0	3	2	5		2
B	2	0	2	0	2	Yes	-
C	1	3	4	5	6		1
D	4	2	6	2	6	Yes	-
E	3	2	5	6	9		4
F	5	6	11	6	11	Yes	-
G	2	5	7	9	11		4

Table 7.11 Earliest and Latest Completion time

The critical path over the network is shown in Figure 7.49.

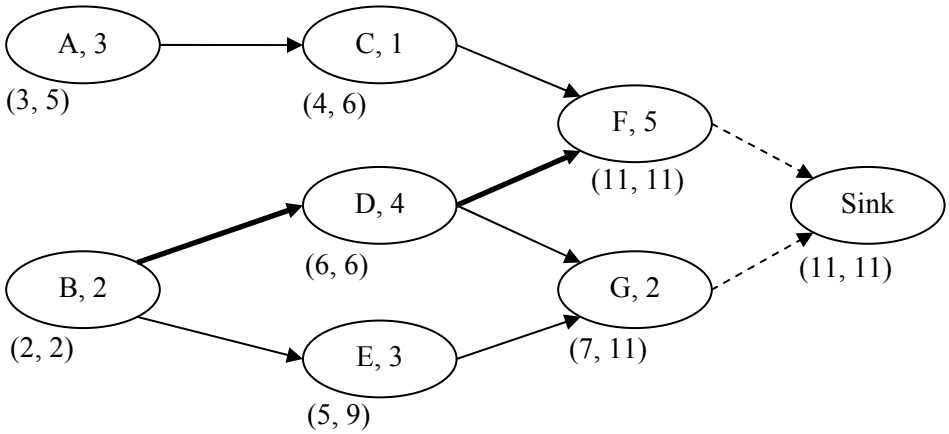


Figure 7.49 Critical Path for network (Example 7.14)

The Gantt chart based on early and late schedules is shown in Figure 7.50

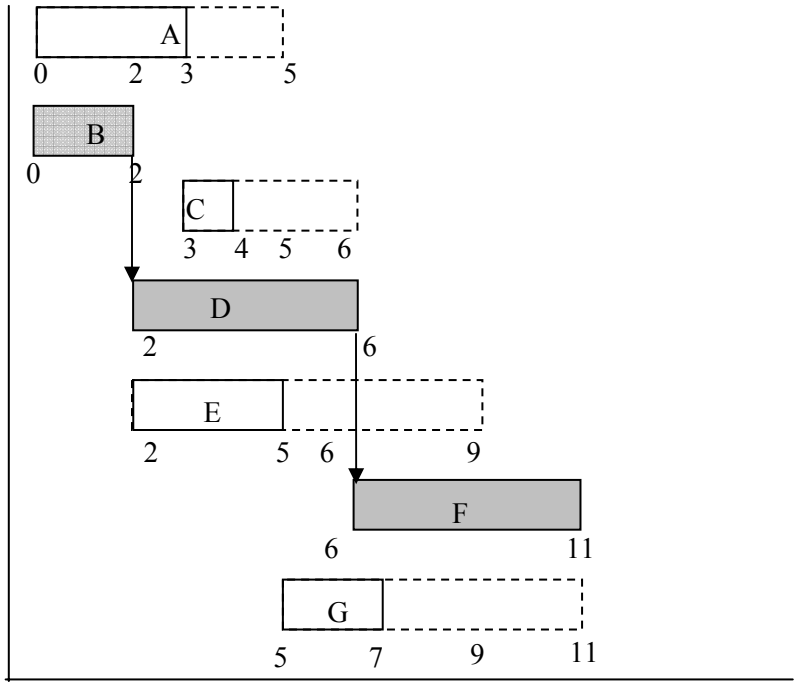


Figure 7.50 Gantt chart for solution of Example

The grey colored boxes represent activities on the critical path. The solid line rectangles represent early start and finish time of each slack activity. The dotted line rectangles represent late start and finish time of each slack activity. The following Table 7.12 shows number of units of the **resource required** in each period.

Task	p_j	1	2	3	4	5	6	7	8	9	10	11
A	3	4	4	4								
B	2	6	6									
C	1				4							
D	4			4	4	4	4					
E	3			2	2	2						
F	5							2	2	2	2	2
G	2							3	3			

Table 7.12 No. of units of resource required

The resource profile over Gantt chart is shown in Fig below. Note the Gantt chart is constructed using S_j' values. The daily resource requirements for 11 days are shown in attached data boxes. Total of the squared resource requirements is 514.

		1	2	3	4	5	6	7	8	9	10	11
A	3	4	4	4								
B	2	6	6									
C	1				4							
D	4			4	4	4	4					
E	3			2	2	2						
F	5							2	2	2	2	2
G	2							3	3			
	total	10	10	10	10	6	4	5	5	2	2	2
		100	100	100	100	36	16	25	25	4	4	4

Table 7.13 Total of the squared resource requirement ($\Sigma R^2 = 514$)

The resource usage graph for resource requirements over 11 days is shown below in Figure 7.51. The graph shows large resource requirements in the beginning periods. The requirements drop to low levels in last periods.

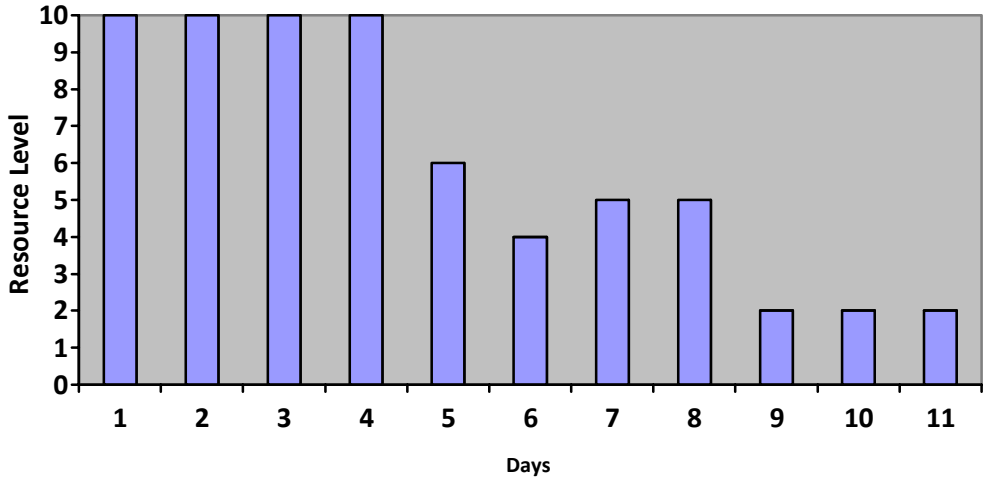


Figure 7.51 Resource usage graph

Shift the slack Task G to its late start time (S_j'') as shown in following Table 7.14.

Task	p_j	1	2	3	4	5	6	7	8	9	10	11
A	3	4	4	4								
B	2	6	6									
C	1				4							
D	4			4	4	4	4					
E	3			2	2	2						
F	5							2	2	2	2	2
G	2										3	3
	total	10	10	10	10	6	4	2	2	2	5	5
		100	100	100	100	36	16	4	4	4	25	25

Table 7.14 Total of the squared resource requirement after shifting task G ($\Sigma R^2 = 514$)

The resource usage graph after shifting task G to its new start time is shown below in figure 7.52.

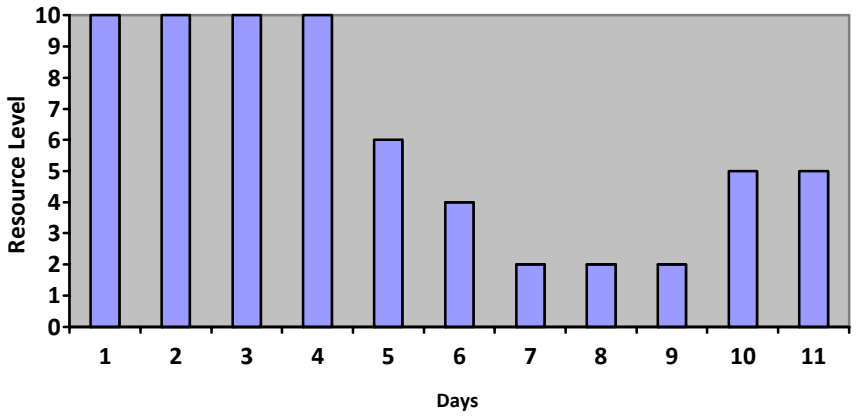


Figure 7.52 Resource usage graph after shifting task G

The lowest use of the resource shifts from left towards center. However, value of $\sum R^2$ remains equal to 514. The resource usage is still maximum at the beginning periods (i.e.; from day 1 to day 4).

Next shift Task E to its late start time (From day 7 to day 9). The value of $\sum R^2$ drops sharply from 514 to 458 is shown in Table 7.14.

Task	p_j	1	2	3	4	5	6	7	8	9	10	11
A	3	4	4	4								
B	2	6	6									
C	1				4							
D	4			4	4	4	4					
E	3							2	2	2		
F	5							2	2	2	2	2
G	2										3	3
	total	10	10	8	8	4	4	4	4	4	5	5
		100	100	64	64	16	16	16	16	16	25	25

Table 7.15 Total of the squared resource requirement after shifting task E ($\Sigma R^2 = 458$)

The resource usage graph after shifting task E to its new start time is shown below in figure 7.53.

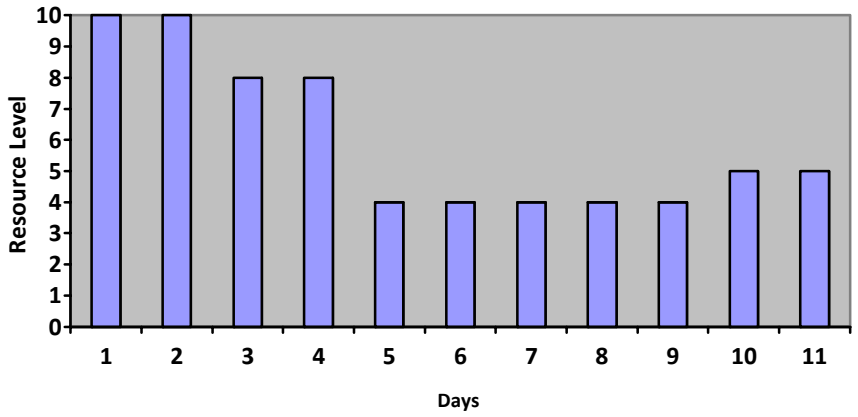


Figure 7.53 Resource usage graph after shifting task E

Next shift Task C to its late start time (From day 4 to day 6). The revised Gantt chart and the value of $\sum R^2$ are shown in Table 7.16.

Task	P_j	1	2	3	4	5	6	7	8	9	10	11
A	3	4	4	4								
B	2	6	6									
C	1						4					
D	4			4	4	4	4					
E	3							2	2	2		
F	5							2	2	2	2	2
G	2										3	3
	total	10	10	8	4	4	8	4	4	4	5	5
		100	100	64	16	16	64	16	16	16	25	25

Table 7.16 Total of the squared resource requirement after shifting task C ($\Sigma R^2 = 458$)

There is no improvement in the value of $\sum R^2$ .The resource usage graph also does not show any marked improvement in resource leveling as shown in Fig 7.54.

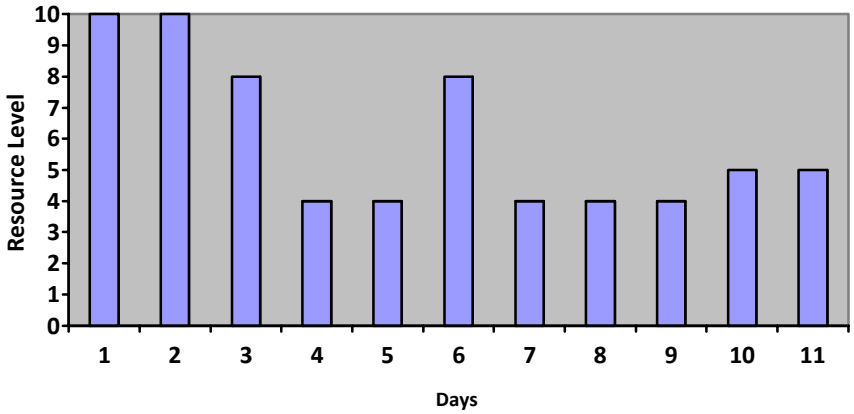


Figure 7.54 Resource usage graph after shifting task C

Finally, shift the Task A to its latest start time (From day 1 to day 3) and, compute resource requirements as below. The value of $\sum R^2$ drops down from 458 to 426 is shown in Table 7.17.

Task	P_j	1	2	3	4	5	6	7	8	9	10	11
A	3			4	4	4						
B	2	6	6									
C	1						4					
D	4			4	4	4	4					
E	3							2	2	2		
F	5							2	2	2	2	2
G	2										3	3
	total	6	6	8	8	8	8	4	4	4	5	5
		36	36	64	64	64	64	16	16	16	25	25

Table 7.17 Total of the squared resource requirement after shifting task C ($\Sigma R^2 = 458$)

The resource usage graph after shifting task E to its new start time is shown below in figure 7.53.

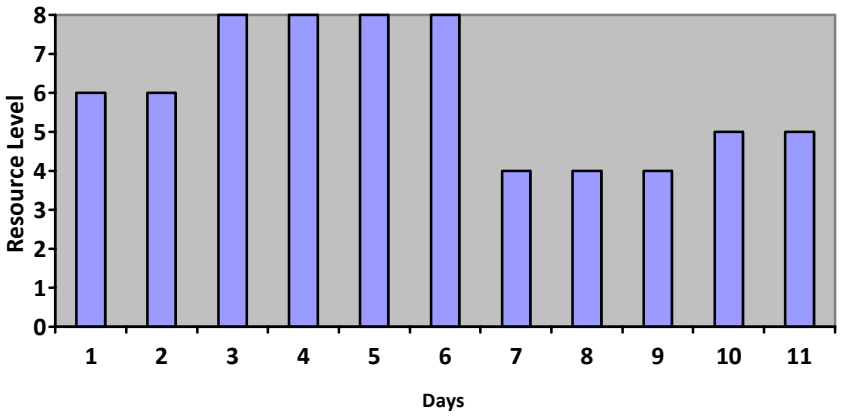


Figure 7.54 Resource usage graph after shifting task A

The Resource usage graph after shifting activity A indicates that maximum requirement for the resource in any period has dropped from 10 to 8.

EXERCISES

7.1 Consider the project data presented in the following table:

Activity	Activity time	Predecessors
A	4	-------
B	4	A
C	5	A, B
D	3	A
E	3	C
F	3	C
G	4	D
H	5	G, F, E
I	10	H,F, G
J	8	I
K	7	J

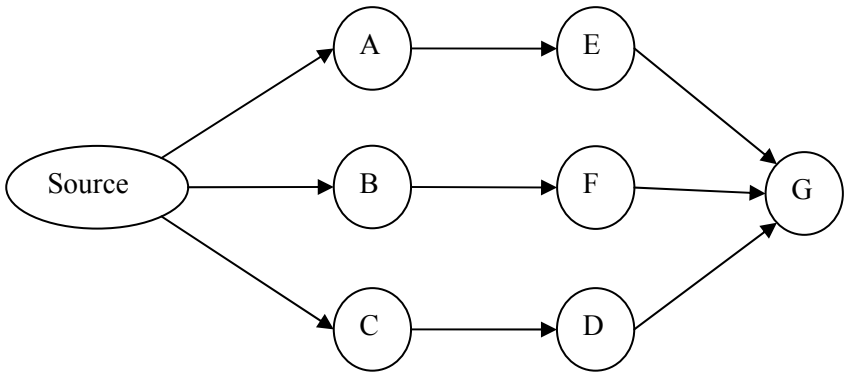
All these tasks require technicians to finish it. If there are three technicians available in the company, how much time will be required to finish it? Use ACTIM procedure to solve the problem

7.2 A maintenance project consist of ten activities labeled A, B, …, J. The duration (in days) and the precedence for the activities are given in the table below:

Activity	Number of resources required	Precedes	Resources requirements
A	7	B, C	5
B	2	D, F	5
C	4	E, G	4
D	3	H	6
E	5	I	2
F	6	I	3
G	1	J	1
H	5	--	4
I	9	--	7
J	8	--	4

a. Develop the project network.
b. Do the forward and backward calculations.
c. What is the slack for activities D, F, and J?
d. Determine all critical paths.
e. If there are ten mechanics available, draw a resource profile (Gantt chart load) using the early start schedule in which ties are broken by the smallest slack activity
f. What is the criticality index (CI) for this problem? (**Note: the criticality index is computed as follows: it is the ratio of average per-unit resource requirement divided by per-unit time resource availability**).
g. Discuss the pro and cons of using 8, 10, or 12 mechanics by comparing the three scenarios using the project completion time and resources utilization.

7.3 Project scheduling problem which uses two resources α and β, where α is the electricians and β is the plumbers. The company has two electricians and three plumbers. Hence, it is Multi-Resources/Multi-Capacity resource scheduling problem. Generate a schedule to minimize the C_{max} for following project network which is shown below:



The resource requirement for each task is given in the following table:

Resource Name	List of Tasks that require the resources
Electrician (α)	A(1) , D(1) , F(2), G(1)
Plumber (β)	B(1), C(2) , D(2),E(2),G(1)

The time for each activity is given in the following table:

Activity(j)	A	B	C	D	E	F	G
Time(Hrs)	5	3	9	4	2	8	11

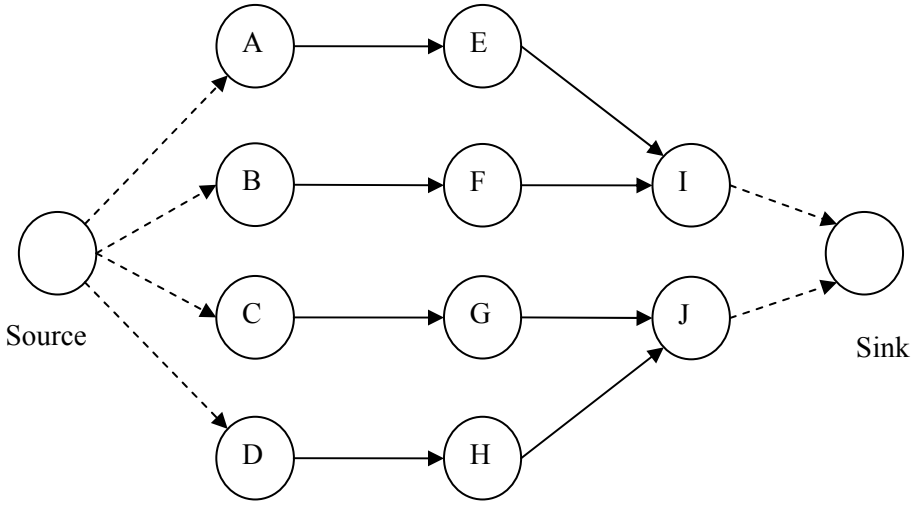
7.4 A small project network with 10 tasks is shown in precedence diagram (Fig. 1). The task durations are shown in Table as under.

Task (j)	A	B	C	D	E	F	G	H	I	J
P_i	3	6	5	4	7	3	8	7	4	2

There are two different types of resources required by tasks. The resource requirements for tasks are;

Resource Name	List of Tasks that require the resource
R_1	A , D , G, I , J
R_2	B, C , E , F , H

Exactly one unit of each resource is available. Generate schedule to minimize C_{max}.



Precedence Graph

8

General Purpose Scheduling

CHAPTER CONTENTS

8.1 INTRODUCTION

The heuristics described before in all chapters are the constructive type. They start without a schedule and gradually construct a schedule by adding one job at a time. The heuristic scheduling techniques presented in this chapter are algorithms of the improvement type. These heuristics aim to find relatively good solutions for scheduling problems with less computational effort. Algorithms of the improvement type are conceptually completely different from algorithms of the constructive type. They start out with a complete schedule, which may be selected arbitrarily, and then try to obtain a better schedule by manipulating the current schedule.

An important class of improvement type algorithms is the local search procedures. A local search procedure does not guarantee an optimal solution. It usually attempts to find a schedule that is better than the current one in the *neighborhood* of the current one. Two schedules are *neighbors,* if one can be obtained through a well defined modification of the other. At all iteration, a local search procedure performs a search within the neighborhood and evaluates the various neighboring solutions. The search process within a neighborhood can be done in a number of ways. A simple way is to select schedules in the neighborhood at random, evaluate these schedules and decide which one to accept. However, it may pay to do a more organized search and select first schedules that appear promising. One may want to consider swapping those jobs that affect the objective the most.

The procedure either accepts or rejects a candidate solution as the next schedule to move to, based on a given acceptance-rejection criterion. The acceptance-rejection criterion is usually the design aspect that distinguishes a local search procedure the most. The difference between the two procedures discussed in the remaining part of this section, simulated annealing and tabu-search, lies mainly in their acceptance-rejection criteria. In simulated annealing the acceptance-rejection criterion is based on a probabilistic process while in tabu-search it is based on a deterministic process.

Three techniques are quite popular for local search. These include simulated annealing, tabu search, and genetic algorithms. These Local search techniques start with a schedule and try to look for a better solution in the *neighborhood*. The method is performed through iterations. Many *neighbor* solutions are generated by making changes in the current solution at each iteration and, compared with current solution. Local search continues through iterations till termination criteria are reached.

8.2 SIMULATED ANNEALING

It is a search process that was originated in materials science. It was first developed for simulating the physical annealing process of solids. In scheduling theory, it is applied as a local search tool to improve the starting schedule. The following algorithm specifies the steps for creating a schedule for single machine problem.

Algorithm

Let's term,

S_C = Candidate schedule
S_O = Best schedule found so far
S_K = Schedule constructed at k^{th} iteration (k = iteration counter)
$G(S_O)$ = Aspiration criterion (value of best schedule)
$G(S_K)$ = Value of schedule constructed at K^{th} iteration.
$G(S_C)$ = Value of candidate schedule
$P(S_K , S_C)$ = Probability of moving from S_k schedule to S_C schedule at k^{th} iteration.

$$P(S_k, S_C) = \exp\left(\frac{G(S_k) - G(S_C)}{\beta_k}\right)$$

Where,

β_k is called cooling parameter in annealing terminology
Usually, $\beta_k = a^k$ where a = [0, 1]

Step 1: Initialize

Set k = 1, set β_1 equal to given value.
Form starting sequence by any heuristic; call it S_1
Let $S_O = S_1$, then $G(S_O) = G(S_1)$

Step 2:

Select S_C from S_K

IF $G(S_O) < G(S_C) < G(S_K)$,
THEN $S_{K+1} = S_C$
GOTO Step (3)

IF $G(S_C) < G(S_O)$,
THEN $S_O = S_{K+1} = S_C$
GOTO Step (3)

IF $G(S_O) > G(S_K)$,
THEN generate a random number $U_k \sim [0, 1]$
IF $U_K \leq P(S_K , S_C)$,
THEN $S_{K+1} = S_C$

ELSE $S_{K+1} = S_K$
GOTO STEP (3)

Step 3:

Set $\beta_{K+1} \leq \beta_K$. Set k = k + 1.
IF k <= N
THEN GOTO Step (2)
ELSE Stop

Example 8.1

For the $1 \parallel \sum w_j T_j$ instance, solve the following problem using Simulated Annealing method. Apply the technique for FIVE iterations (K <= 5).

Job	1	2	3	4
P_j	3	6	2	5
D_j	4	7	13	9
w_j	3	1	2	4

Use the following U values = {0.8, 0.01, 0.52, 0.43} in the problem. Take initial value of β equal to 0.9. Start with first sequence as $\{j_1, j_2, j_3, j_4\}$

Solution:

Step 1: Initialize;

Given:

β_1=0.9, k=1 , $S_1 = \{j_1, j_2, j_3, j_4\}$
Set $S_O = S_1$

Find $\sum w_j T_j$

Job	j_1	j_2	j_3	j_4
P_j	3	6	2	5
D_j	4	7	13	9
C_j	3	9	11	16
T_j	0	2	0	7
w_j	3	1	2	4
$w_j T_j$	0	2	0	28
$\sum w_j T_j$ =	30			

Table 8.1 Calculation of $G(S_1)$

So, $G(S_O) = 30$, $G(S_1) = 30$,

Step 2: Iteration #1

Select a candidate sequence S_C from neighborhood of S_1

Let $S_C = \{j_2, j_1, j_3, j_4\}$

Find $\sum w_j T_j$ for sequence S_C

Job	j_2	j_1	j_3	j_4
P_j	6	3	2	5
D_j	7	4	13	9
C_j	6	9	11	16
T_j	0	5	0	7
w_j	1	3	2	4
$w_j T_j$	0	15	0	28
$\sum w_j T_j =$	43			

Table 8.2 Calculation of $G(S_C)$

Hence, $G(S_C) = 43$,

Now test the conditions;

IF $G(S_O) < G(S_C) < G(S_1)$,THEN $S_2 = S_C$ GOTO Step (3)

IF $G(S_C) < G(S_O)$, THEN $S_O = S_C$ & $S_2 = S_C$ GOTO Step (3)

} These conditions are False

IF $G(S_C) > G(S_1)$, ⟵This condition is true

THEN

Generate U_1 between [0, 1]. Given $U_1 = 0.8$

Find $P(S_1 , S_C) = \exp\left\{\frac{G(S_1) - G(S_C)}{\beta_1}\right\} = e^{\left\{\frac{30-43}{0.9}\right\}} = e^{-14.4} = 5.574 \times 10^{-7}$

IF $U_1 <= P(S_1 , S_C)$ → No

So, Set $S_2 = S_1$, $S_2 = \{j_1, j_2, j_3, j_4\}$

GOTO Step (3)

Step 3:

Select β_2 $\beta_2 = (\beta_1)^2 = 0.81$

k = 2, N=5

IF k <= N.

GOTO Step (2), otherwise STOP

.....

Step 2: Iteration #2

Select a candidate sequence S_C from neighborhood of S_2 as follows

$S_2 = \{j_1, j_2, j_3, j_4\}$

$S_C = \{j_1, j_3, j_2, j_4\}$

Find $\sum w_j T_j$ for sequence S_C

Job	j_1	j_3	j_2	j_4
P_j	3	2	6	5
D_j	4	13	7	9
C_j	3	5	11	16
T_j	0	0	4	7
w_j	3	2	1	4
w_jT_j	0	0	4	28
$\sum w_jT_j =$	32			

Table 8.3 Calculation of $G(S_C)$

Hence, $G(S_C) = 32$

Now test the conditions;

IF $G(S_O) < G(S_C) < G(S_2)$,THEN $S_3 = S_C$ GOTO Step (3)

IF $G(S_C) < G(S_O)$, THEN $S_O = S_C$ & $S_3 = S_C$ GOTO Step (3)

} These conditions are False

IF $G(S_C) > G(S_2)$, ⟵ This condition is true

THEN

Generate U_2 between [0, 1]. Given $U_2 = 0.01$

Find $P(S_2 , S_C) = \exp\left\{\frac{G(S_2) - G(S_C)}{\beta_2}\right\} = e^{\left\{\frac{30-32}{0.81}\right\}} = 0.085$

IF $U_2 <= P(S_2 , S_C)$ → Yes

So, Set $S_3 = S_C$, $S_3 = \{j_1, j_3, j_2, j_4\}$

GOTO Step (3)

Step 3:

Select β_3 $\beta_3 = (\beta_1)^3 = 0.729$

k = 3, N = 5

IF k <= N. GOTO Step (2), otherwise STOP

Step 2: Iteration #3

Select a candidate sequence S_C from neighborhood of S_3 as follows

$S_3 = \{j_1, j_3, j_2, j_4\}$

$S_C = \{j_3, j_1, j_2, j_4\}$

Find $\sum w_j T_j$ for sequence S_C

Job	j_3	j_1	j_2	j_4
P_j	2	3	6	5
D_j	13	4	7	9
C_j	2	5	11	16
T_j	0	1	4	7
w_j	2	3	1	4
$w_j T_j$	0	3	4	28
$\sum w_j T_j =$	35			

Table 8.4 Calculation of $G(S_C)$

Hence, $G(S_C) = 35$

Now, test the conditions

IF $G(S_0) < G(S_C) < G(S_3)$,THEN S3 = SC GOTO Step (3)

IF $G(S_C) < G(S_0)$, THEN $S_0 = S_C$ & $S_3 = S_C$ GOTO Step (3)

} These conditions are False

IF $G(S_C) > G(S_3)$, ⟵ This condition is true

THEN

Generate U_3 between [0, 1]. Given $U_3 = 0.52$

Find $P(S_3, S_C) = \exp\left\{\frac{G(S_3) - G(S_C)}{\beta_3}\right\} = e^{\left\{\frac{30-35}{0.729}\right\}} = e^{\frac{-5}{0.729}} = 0.00105$

IF U_3 <= $P(S_3, S_C)$ → No

S_O, Set $S_4 = S_3$, $S_4 = \{j_1, j_3, j_2, j_4\}$

GOTO Step (3)

Step 3:

Select β_4 $\beta_3 = (\beta_1)^4 = 0.6561$

k = 4, N = 5

IF k <= N. GOTO Step (2), otherwise STOP

Step 2: Iteration #3

Select a candidate sequence S_C from neighborhood of S_4 as follows

$S_4 = \{j_1, j_3, j_2, j_4\}$

$S_C = \{j_1, j_3, j_4, j_2\}$

Find $\sum w_j T_j$ for sequence S_C

Job	j_1	j_3	j_4	j_2
P_j	3	2	5	6
D_j	4	13	9	7
C_j	3	5	10	16
T_j	0	0	1	9
w_j	3	2	4	1
$w_j T_j$	0	0	4	9
$\sum w_j T_j =$	13			

Table 8.5 Calculation of $G(S_C)$

Hence, $G(S_C) = 13$

Now, test the conditions

IF $G(S_C) < G(S_O)$, ⟵ This condition is true

THEN

New values of S_O and S_5 are;

$S_O = S_C = \{j_1, j_3, j_4, j_2\}$, $G(S_O) = 13$

$S_5 = S_C = \{j_1, j_3, j_4, j_2\}$, $G(S_5) = 13$

GOTO Step (3)

Step 3:

Select β_5 $\beta_5 = (\beta_1)^5 = 0.6561$

k = 5, N=5

IF k <= N. GOTO Step (2), otherwise STOP

Step 2: Iteration 5

Select a candidate sequence S_C from neighborhood of S_5 as follows

$S_5 = \{j_1, j_3, j_4, j_2\}$

$S_C = \{j_1, j_4, j_3, j_2\}$

Find $\sum w_j T_j$ for sequence S_C

Job	j_1	j_4	j_3	j_2
P_j	3	5	2	6
D_j	4	9	13	7
C_j	3	8	10	16
T_j	0	0	0	9
w_j	3	4	2	1
w_jT_j	0	0	0	9
$\sum w_jT_j$ =	9			

Table 8.6 Calculation of $G(S_C)$

Hence, $G(S_C) = 9$
Now, test the conditions
IF $G(S_C) < G(S_O)$, ⟵This condition is true

THEN
New values of S_O and S_5 are;
$S_O = S_C = \{j_1, j_4, j_3, j_2\}$, $G(S_O) = 13$
$S_6 = S_C = \{j_1, j_4, j_3, j_2\}$, $G(S_6) = 13$
GOTO Step (3)

Step 3:

Select β_6 $\beta_6 = (\beta_1)^6 = 0.6561$
k = 6, N=5
IF k <= N. GOTO Step (2), otherwise STOP

We stop at the end of Iteration #5. Best Sequence = $S_0 = \{ j_1, j_4, j_3, j_2 \}$

Minimum weighted Tardiness = $\sum w_jT_j = 9$

8.3 TABU SEARCH

It is local search procedure like simulated annealing. However, the selection of neighborhood schedule is deterministically decided as opposed to simulated annealing where probabilistic approach is followed. A record of Tabu moves is kept in Tabu list to avoid duplication of job swaps in the list.

Tabu List

The main components of a tabu-search algorithm are memory structures, in order to have a trace of the evolution of the search, and strategies for using the memory information in the best possible way. The fundamental memory structure is a so-called tabu list, which stores attributes characterizing solutions that should not be considered again for a certain length of time. Usually a first-in-first-out (FIFO) strategy is applied to the list. Old attributes are deleted as new attributes are inserted.

Algorithm

Step 1: Initialize

Set k = 1
Form starting sequence by any heuristic; call it S_1
Let $S_O = S_1$,
Then $G(S_O) = G(S_1)$

Step 2:

Select S_C from neighborhood of S_K
IF move from S_K to S_C is not allowed in the Tabu List
THEN $S_{K+1} = S_K$, GOTO step (3)
IF $G(S_C) < G(S_0)$,
THEN $S_0 = S_C$
Delete the oldest Tabu move in the Tabu List
Add fresh Tabu move at head of the list.
GOTO Step (3)

Step 3:

Set k = k + 1.
IF k <= N
THEN GOTO Step (2)
ELSE Stop

Example 8.2

Solve the problem in Example 8.1 using Tabu Search. Apply the technique for FIVE iterations. Make the length of the Tabu List equal to two. (i.e. the pair of jobs that were interchanged during last moves can not be interchanged again).

The data from Example 8.1 is shown below.

Job	j_1	j_2	j_3	j_4
P_j	3	6	2	5
D_j	4	7	13	9
w_j	3	1	2	4

Solution:

Iteration 1

Initially, Tabu List = { },
Let, starting sequence, $S_1 = \{j_1, j_2, j_3, j_4\}$
Let, S_0 = Overall best sequence,
Set, $S_0 = S_1$.
So, $S_0 = \{j_1, j_2, j_3, j_4\}$
Then, $G(S_0) = 30$, $G(S_1) = 30$.

The neighborhood of a schedule is defined as all schedules that can be obtained through adjacent pair wise interchanges.

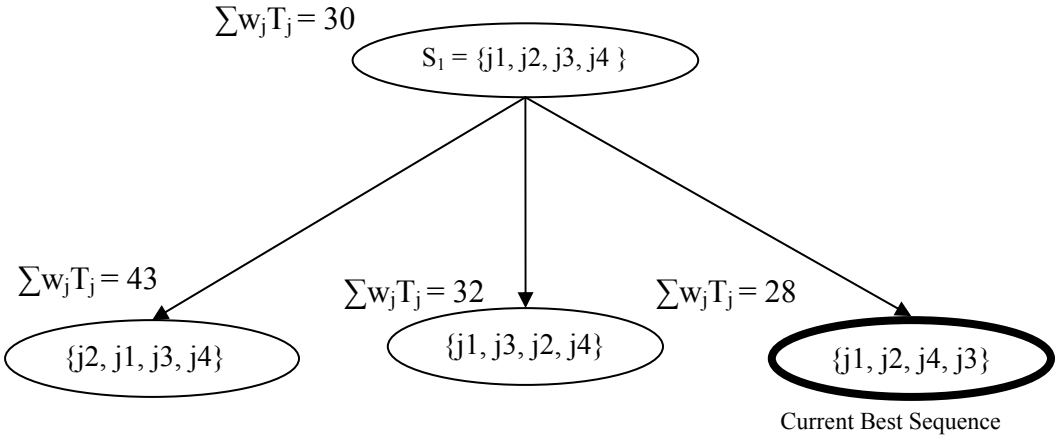


Figure 8.1 Sequences of S_1 by adjacent pair wise interchange

The computations of $\sum w_j T_j$ for three sequences are shown in Tables 8.7 below.

Job(j)	j_2	j_1	j_3	j_4	j_1	j_3	j_2	j_4	j_1	j_2	j_4	j_3
P_j	6	3	2	5	3	2	6	5	**3**	6	5	2
D_j	7	4	13	9	4	13	7	9	4	7	9	13
C_j	6	9	11	16	3	5	11	16	3	9	14	16
T_j	0	5	0	7	0	0	4	7	0	2	5	3
W_j	1	3	2	4	3	2	1	4	3	1	4	2
w_jT_j	0	15	0	28	0	0	4	28	0	2	20	6
$\sum w_jT_j$	43				32				28			

Table 8.7 Computations of $\sum w_jT_j$ for three sequences

Let S_C = Current best sequence
Then, from three sequences from S_1, the current best sequence is $\{j_1, j_2, j_4, j_3\}$,
So, $S_C = \{j_1, j_2, j_4, j_3\}$.
The value of S_C is then; $G(S_C) = 28$,

Since, $G(S_C) < G(S_0)$,
$S_0 = S_C$, and, New value of $G(S_0) = 28$,

Since, the current best sequence S_C has been obtained by interchanging jobs 3 and 4 in sequence S_1; Tabu Move = (3, 4), and, Tabu List = {(3, 4)}
Set, $S_2 = S_C$
And, go to next iteration.

Iteration 2:

Now, generate all sequences of S_2 by interchanging adjacent pairs as shown below

Sequence S_2	Pair to be interchanged	New Sequence
$\{j_1, j_2, j_4, j_3\}$	{1, 2}	$\{j_2, j_1, j_4, j_3\}$
$\{j_1, j_2, j_4, j_3\}$	{2, 4}	$\{j_1, j_4, j_2, j_3\}$
$\{j_1, j_2, j_4, j_3\}$	{4, 3}	$\{j_1, j_2, j_3, j_4\}$

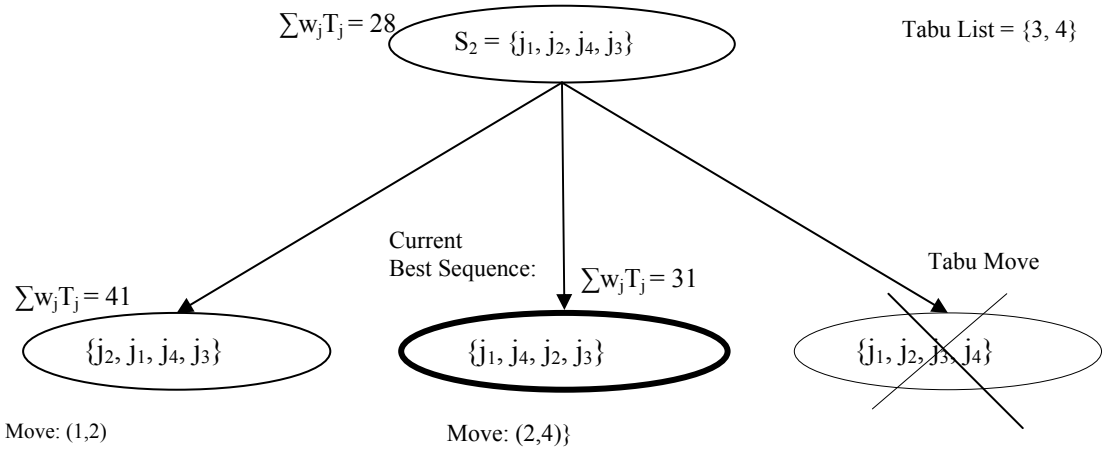


Figure 8.2 Sequences of S_2 by adjacent pair wise interchange

The computations of $\sum w_j T_j$ for sequences $\{j_2, j_1, j_4, j_3\}$ & $\{j_1, j_4, j_2, j_3\}$ are shown in the Table 8.8 below.

Job(j)	j_2	j_1	j_4	j_3	j_1	j_4	j_2	j_3
P_j	6	3	5	2	3	5	6	2
D_j	7	4	9	13	4	9	7	13
C_j	6	9	14	16	3	8	14	16
T_j	0	5	5	3	0	0	7	3
W_j	1	3	4	2	3	4	1	2
$w_j T_j$	0	15	20	6	0	0	7	6
$\sum w_j T_j$	41				13			

Table 8.8 Computations of $\sum w_j T_j$ for two sequences

Current best sequence by adjacent pair wise interchanges in Sequence S_2 is $\{j_1, j_4, j_2, j_3\}$,
So, $S_C = \{j_1, j_4, j_2, j_3\}$. The value of S_C is; G (S_C) = 13,

Since, $G(S_C) < G(S_0)$,
$S_0 = S_C$, and, New value of G (S_0) = 13,

Since, the current best sequence S_C has been obtained by interchanging jobs 2 and 4 in sequence S_2;

Tabu Move = (2, 4), and, Tabu List = {(3, 4), (2, 4)}

Set, $S_3 = S_C = \{j_1, j_4, j_2, j_3\}$, $G(S_0) = 13$
GOTO, next iteration.

Iteration 3:

Now, generate all sequences of S_2 by interchanging adjacent pairs as shown below in Figure 8.3

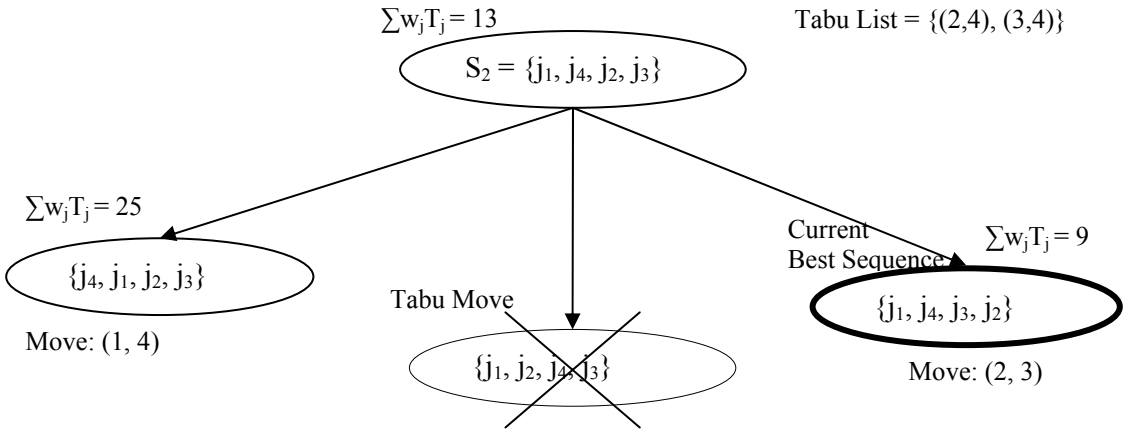


Figure 8.3 Sequences of S_3 by adjacent pair wise interchange

The computations of $\sum w_j T_j$ for sequences $\{j_4, j_1, j_2, j_3\}$ & $\{j_1, j_4, j_3, j_2\}$ are shown in Table 8.9;

Job(j)	j_4	j_1	j_2	j_3	j_1	j_4	j_3	j_2
P_j	5	3	6	2	3	5	2	6
D_j	9	4	7	13	4	9	13	7
C_j	5	8	14	16	3	8	10	16
T_j	0	4	7	3	0	0	0	9
W_j	4	3	1	2	3	4	2	1
w_jT_j	0	12	7	6	0	0	0	9
$\sum w_jT_j$	25				9			

Table 8.9 Computations of $\sum w_j T_j$ for two sequences

Current best sequence by adjacent pair wise interchanges in Sequence S_3 is $\{j_1, j_4, j_3, j_2\}$,
So, $S_C = \{j_1, j_4, j_3, j_2\}$. The value of S_C is; $G(S_C) = 9$,
Since, $G(S_C) < G(S_0)$,
$S_0 = S_C$, and, New value of $G(S_0) = 9$,

Since, the current best sequence S_C has been obtained by interchanging jobs 2 and 3 in sequence S_3;
Tabu Move = (2, 3). Since length of Tabu List is 2, we update the list. Remove (2, 4) from List and add (2, 3). New Tabu List = {(3, 4), (2, 3)}.

Set, $S_4 = S_C = \{j_1, j_4, j_3, j_2\}$. $G(S_0) = 9$

Iteration 4:

Now, generate sequences of S_4;

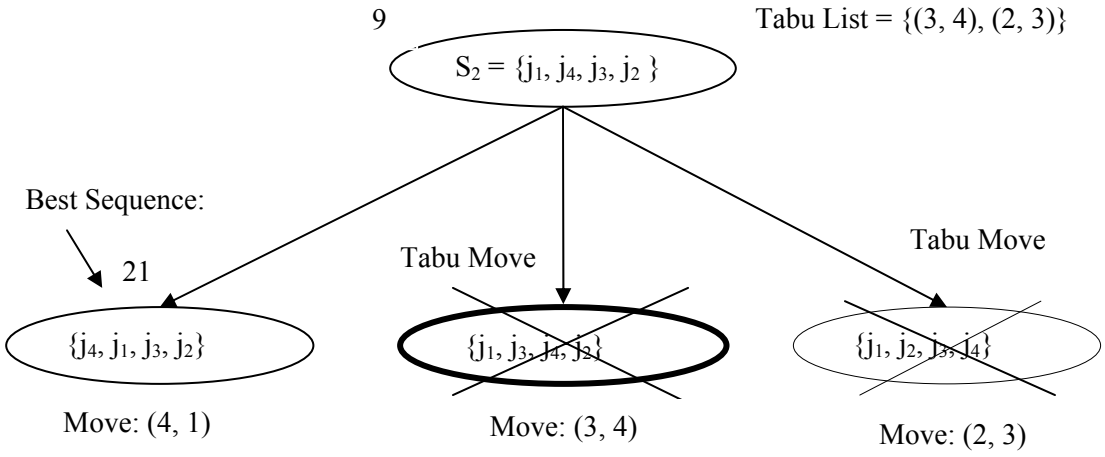


Figure 8.4 Sequences of S_4 by adjacent pair wise interchange

The computation for sequence $\{j_4, j_1, j_3, j_2\}$ is;

Job(j)	j_4	j_1	j_3	j_2
P_j	5	3	2	6
D_j	9	4	13	7
C_j	5	8	10	16
T_j	0	4	0	9
W_j	4	3	2	1
w_jT_j	0	12	0	9

$\sum w_j T_j$	21

Table 8.10 Computations of $\sum w_j T_j$ for two sequences

Only possible move is (1,4) with $G(S_C) = 21$.
Since, $G(S_C) > G(S_0)$, $G(S_0)$ remains at value 9.

With Tabu Move = (4, 1) in Sequence S_4, update Tabu List. Remove (3, 4) from List and add (4, 1). New Tabu List = {(2, 3), (4, 1)}

Set, $S_5 = S_C = \{j4, j_1, j_3, j_2\}$. $G(S_0) = 9$
GOTO, next iteration.

Iteration 5:

Now, generate sequences of S_5 as follows:

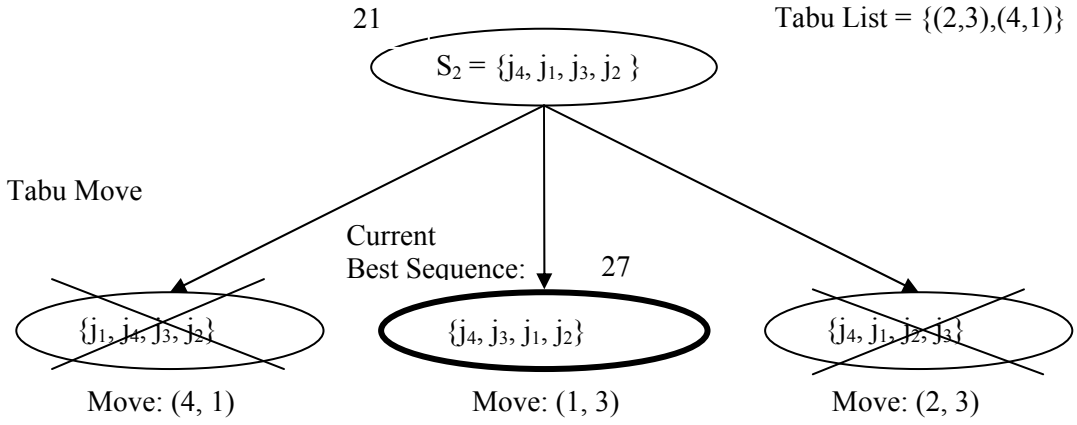


Figure 8.5 Sequences of S_4 by adjacent pair wise interchange

The computation for sequence $\{j_4, j_3, j_1, j_2\}$ is;

Job(j)	j_4	j_3	j_1	j_2
P_j	5	2	3	6
D_j	9	13	4	7
C_j	5	7	10	16
T_j	0	0	6	9
W_j	4	2	3	1
$w_j T_j$	0	0	18	9

$\sum w_j T_j$	27

Table 8.11 Computations of $\sum w_j T_j$ for two sequences

Only possible move is (1, 3) with $G(S_C) = 27$.
Since, $G(S_C) > G(S_0)$, $G(S_0)$ remains at value 9.
With Tabu Move = (1, 3) in Sequence S_5, update Tabu List. Remove (2, 3) from List and add (1, 3). New Tabu List = {(4, 1), (1, 3)}
Set, $S_6 = S_C = \{j_4, j_3, j_1, j_2\}$. $G(S_0) = 9$

At the end of Iteration #5, $G(S_0) = 9$, and best overall sequence = $\{j_1, j_4, j_3, j_2\}$.

EXERCISES

8.1 Consider $1 \mid\mid \sum T_j$ problem for the data given in the following Table

jobs	1	2	3	4
P_j	10	8	12	6
D_j	7	5	8	3

Solve the problem by using Simulated Annealing method. Take the initial parameters as follows:
$\beta_1 = 0.9$, Initial Sequence = { j_4, j_3, j_2, j_1}.Take U values in the following order

	1	2	3
U values	0.17	0.92	0.67

Perform THREE iterations and find the best sequence.

8.2 Consider $1 \mid\mid \sum T_j$ problem for the data given in the following Table

jobs	1	2	3	4
p_j	10	8	12	6
d_j	7	5	8	3

Solve the problem by using tabu-search method. Take Initial Sequence = {j_4, j_3, j_2, j_1}. Keep length of tabu list equal to 2. Perform THREE iterations and find the best sequence.

8.3 Consider the instance of $P_2 \mid\mid \sum w_j T_j$ with following 6 jobs

jobs	1	2	3	4	5	6
p_j	13	9	13	10	8	11
d_j	6	18	10	11	13	18
w_j	2	4	2	5	4	3

Solve the problem using Simulated Annealing method. Using the initial parameters $\beta_1 = 0.8$, Initial Sequence = {$j_5, j_4, j_2, j_1, j_6, j_3$}.Take U values in the following order $U_1 = 0.23$, $U_2 = 0.54$, $U_3 = 0.93$. Perform THREE iterations and find the best sequence.

8.4 Consider the instance of $F_3 \mid p_{ij} = p_i \mid \sum w_j T_j$ for following problem

jobs	1	2	3	4
p_j	9	9	12	3
d_j	10	8	5	28
w_j	14	12	1	12

Apply tabu-search consider as the neighborhood again all schedules that can be obtained through adjacent pair wise interchanges. Start out with sequence 3, 1, 4, 2 and apply the technique for four iterations. Keep the length of the tabu-list equal to 2. Determine whether the optimal sequence is reached.

8.5 Consider the same instance as in problem 8.4. Now apply simulated annealing to this instance. Adopt the same neighborhood structure and select neighbors within the neighborhood at random. Choose $\beta_k = (0.9)k$. Start with 3, 2, 1, 4 as the initial sequence. Terminate the procedure after two iterations and compare the result with the result obtained in the previous exercise. Use the following numbers as uniform random numbers: $U_1 = 0.91$, $U_2 = 0.27$, $U_3 = 0.83$, $U_4 = 0.17$.

9

Genetic Algorithm

CHAPTER CONTENTS

9.1 INTRODUCTION

Genetic algorithms were developed by Holland in 1975. Since genetic algorithms (GAs) are adaptive and flexible, the GAs were shown to be successfully applied to several optimization problems. For example, they have been applied to routing, scheduling, adaptive control, game playing, cognitive modeling, transportation problems, traveling salesman problems, optimal control problems, database query optimization, etc.

The GAs are stochastic search techniques whose search algorithms simulate natural phenomena (biological evolution). The basic idea of the GAs is that the strong tend to adapt and survive while the weak tend to die. One of the strengths of GAs is that they use past information to direct their search with the assumption of improved performance. The formal description of the GA which was provided by Grefenstette is as follows:

> ...A genetic algorithm is an iterative procedure maintaining a population of structures that are candidate solutions to specific domain challenges. During each temporal increment (called a generation), the structures in the current population are rated for their effectiveness as domain solutions, and on the basis of these evaluations, a new population of candidate solutions is formed using specific genetic operators such as reproduction, crossover, and mutation. (Grefenstette 1985)

9.2 METHODOLOGY

The general procedures of the GA are as follows:

1. Initialize a population of binary or non-binary chromosomes.
2. Evaluate each chromosome in the population using the fitness function.
3. Select chromosomes to mate (reproduction).
4. Apply genetic operators (crossover and mutation) on chromosome selected.
5. Put chromosomes produced in a temporary population.
6. If the temporary population is full, then go to step 7. Otherwise, go to step 3.
7. Replace the current population with the temporary population.
8. If termination criterion is satisfied, then quit with the best chromosome as the solution for the problem. Otherwise, go to step 2.

In the above steps, the first element is the size of the population and how to generate the initial population. The initial population of chromosomes can be

generated randomly or by using some heuristics that are suitable for the problem considered. The determination of the population size is a crucial element in the GAs. Selecting a very small population size increases the risk of prematurely converging to a local optimal. Large population sizes increase the probability of converging to a global optimal, but it will take more time to converge. In most of the GA applications, the population size was maintained at a constant.

The second element of the GAs is the fitness function, which is very important to the GAs process of evolution. The GA without a fitness function is blind because, as mentioned earlier, the GA directs its search using historical data which are the fitness values of each chromosome. The GA will use the fitness value of each chromosome to determine if the chromosome can survive and produce offspring, or die.

The selection of chromosomes to reproduce is the third element of the GA. This is a very important element in the GA because it plays an important role in the convergence of the GA. If the selection process is always biased to only accept the best chromosome, the algorithm will quickly have a population of almost the same chromosomes which will cause the GA to converge to a local optimum. Several selection methods have been employed by several researchers to select among the best performers. Some of these methods are: the proportional selection scheme; the roulette wheel selection; deterministic selection; ranking selection; tournament selection, etc.

In step four, two genetic operators were used. The first operator is crossover, which combines the features of two fittest chromosomes and carries these features to the next generation by forming two offspring. The SGA performs the crossover by selecting two chromosomes and a random crossover position (single-position crossover method), then the corresponding parental segments are swapped to form two new children. Several crossover methods have been developed and applied to binary representation. One of them is the two-position crossover method, which is performed by selecting two crossover positions in two chromosomes and then swapping segments between the two chromosomes. The multi-position crossover method is a natural extension of the two-position crossover. A version of the multi-position crossover method is the segmented crossover method, which varies the number of segments during the implementation of the GAs while the multi-position crossover uses a fixed number of segments. Shuffle crossover was proposed as a crossover method which first shuffles the crossover positions in the two selected chromosomes. Then it exchanges the segments between the crossover positions and finally un-shuffles the chromosomes. The final crossover method proposed is the uniform crossover, a generalization of the one-position and multi-position crossover methods. The uniform crossover method produces two new children by exchanging genes in two chromosomes according to a crossover probability and a random value

given to the same gene in both chromosomes. The random value assigned to each gene is uniformly distributed between 0 and 1 and denoted by X_i, $i = 1,...,n$ where n is the number of genes. The uniform crossover is performed as follows: Let P_1 and P_2 be two parents in which each has n genes so that $P_1 = \{P_{11}, P_{12}, P_{13},..., P_{1n}\}$ and $P_2 = \{P_{21}, P_{22}, P_{23},..., P_{2n}\}$. These two parents will produce two children which are denoted by C_1 and C_2. Hence, if the crossover probability is P_c, then the uniform crossover is performed as follows:

If $X_i < P_c$ then $C_{1i} = P_{1i}$ and $C_{2i} = P_{2i}$ and If $X_i \geq P_c$ then $C_{1i} = P_{2i}$ and $C_{2i} = P_{1i}$

To demonstrate how the uniform crossover method works, assume that there are two chromosomes and each gene is assigned a random value as shown below:

P_1: 0110000111, and P_2: 0001011111

Assume X_i = 0.79, 0.83, 0.44, 0.88, 0.11, 0.89, 0.59, 0.7, 0.45, and 0.14,
for i = 1,...,10.

Assume that the P_c is 0.5. The implementation of the uniform crossover method will result in the following children:

C_1: 0011011111, and C_2: 0100000111

The second operator is mutation, which alters one or more of the chromosome genes randomly to ensure search diversification, which hopefully will lead the population out of the local optimum. In the SGA approach, the mutation is performed by first selecting a mutation position. Then, if the gene value is 0, it is flipped to 1. If the gene value is 1, then it is changed to 0.

Finally, the last element in the GA procedures is the stopping criterion. Several criteria have been suggested. One of them is that the GA will stop if the maximum number of generations has been reached, or if the population has converged. The convergence of the population has been interpreted by researchers through several measures. One of them is that the GA converges after a chromosome with a certain high fitness value is located. Another one is that the GA converges after all chromosomes have attained a certain degree of homogeneity (that is, all of them have almost the same fitness value).

7.3 SEQUENCING AND SCHEDULING PROBLEMS

A binary representation of a population was not suitable for all applications. One of the applications that the binary representation was not suitable for, but can be applied to, is the combinatorial optimization problems. Some of these combinatorial optimization problems are the traveling salesman problem (TSP), the bin packing problem, the job scheduling problem (JSP), the plant layout, etc. Several representations of population have evolved from the applications of the genetic

algorithms (GAs) to the TSP. Because of the similarities between TSP and JSP, these representations have been used in JSP. In the following paragraphs, population representations and the associated genetic operators that have been applied to JSP will be discussed.

Ordinal representation was developed by Grefenstette et al. (1985). It was developed to represent a population in a GA approach that solved a TSP. In the ordinal representation method, all classical crossover methods that were explained earlier can be applied to the ordinal representation method. However, the classical mutation method cannot be applied because there is no gene that can be flipped to either 0 or 1. Therefore, several mutation methods have been developed to handle such population representations and other representations. One of these mutation methods is the simple inversion, which is performed by first selecting two mutation positions in a chromosome. The segment between these two positions is reversed. The second mutation method, called insertion, is where a gene is selected and inserted in a random place. Displacement is the third method, which is performed by selecting a string of genes which is inserted in a random position. Order-based mutation (OBM) is the fourth method, which selects two genes randomly and swaps them. A version of the order-based mutation is position-based mutation (PBM), which selects two genes randomly and then inserts the second gene before the first. Scramble sub-sequence mutation (SSM) is another mutation method, which selects a sub-sequence in a chromosome, and scrambles the genes in the sub-sequence to produce a new chromosome.

The second representation method is an order-based representation (also called permutation ordering representation, path representation, natural representation, or direct representation) where a chromosome is represented by a sequence of jobs. In the order-based representation method, a chromosome is formed as a sequence of jobs, such as: 4-6-9-7-5-3-1-2-8. This chromosome is interpreted as follows: job 4 is sequenced first, job 6 is sequenced second, and likewise until job 2 is sequenced second to last, and then job 8 is sequenced last. Clearly this representation is simple and has a meaningful interpretation. All mutation methods that are applied to the ordinal representation method can be applied to the order based representation method. However, infeasible chromosomes will be generated when the classical crossover method that was explained in the previous section are performed. The infeasible chromosomes produced by the classical crossover can be demonstrated by the following example. Assume that in the initial population there are two parents which are:

Parent 1: 4-6-9-7-5-3-1-2-8 and Parent 2: 8-2-4-6-9-1-3-5-7

A single-position crossover method is performed on the two parents, where the single-position crossover is denoted by '|' as shown below.

Parent 1: 4-6-9-7-5-3-|1-2-8 and Parent 2: 8-2-4-6-9-1-|3-5-7

The result of the crossover is shown below:

Child 1: 4-6-9-7-5-3-3-5-7 and Child 2: 8-2-4-6-9-1-1-2-8

It is obvious that both of the children represent infeasible sequences because both of them have only six jobs out of the nine jobs, and each has three duplicated jobs. Therefore, to solve this infeasibility problem, several crossover methods that produce feasible chromosomes were proposed by several researchers:

1. Order Crossover (OX) by Davis (1985).
2. Partially Mapped Crossover (PMX) by Goldberg and Lingle (1985).
3. Sub-sequence-Swap crossover (SSX) and Sub-sequence-Chunk Crossover (SCX) by Grefenstette et al. (1985).
4. Cycle Crossover (CX) by Oliver, Smith, and Holland (1987).
5. Edge Recombination Crossover (ERX) by Whitley, Starkweather, and Fuguay (1989).
6. Linear order Crossover (LOX) by Falkenauer and Bouffouix (1991).
7. Order-based Crossover (OBX) and Position-based Crossover (PBX) by Syswerda (1991).
8. Enhanced edge recombination crossover (EERX) by Starkweather et al. (1991).

The PMX was developed by Goldberg and Lingle (1985) to handle the infeasibility problem in a GA approach that was applied to TSP. Given two parents, the PMX first randomly selects two positions which are the same in both parents. Then segments between these two positions are exchanged. The exchanging of the segments will define a series of mappings between genes. The defined mappings will be used to replace genes that are causing infeasibility in the new chromosomes. The following example will show how the PMX works assuming that the following parents are given:

Parent 1: 4-6-9-7-5-3-1-2-8 and Parent 2: 8-2-4-6-9-1-3-5-7

The two cutting positions on the two parents are selected where the two positions are denoted by '|' as shown below:

Parent 1: 4-6-|9-7-5-3|-1-2-8 and Parent 2: 8-2-|4-6-9-1|-3-5-7

The result of the segment swapping is shown below:

Child 1: x-x-|4-6-9-1|-x-x-x and Child 2: x-x-|9-7-5-3|-x-x-x

From the segments swapped, the defined mappings are as follows: 4⇔9, 6⇔7, 9⇔5, and 1⇔3. Therefore, the defined mappings will be used to correct infeasibility. In

parent 1, job 4 is mapped as follows: 4⇔9⇔5 (job 4 is replaced with job 5). Job 6 is replaced with job 7. Job 1 is replaced with job 3. Both jobs 2 and 8 are not causing infeasibility, hence, they are not involved. In parent 2, job 3 is replaced with job 1. Job 5 is replaced with job 4, because of the mapping, 5⇔9⇔4. Job 7 is replaced with 6. The result of the PMX is two feasible children given below:

Child 1: 5-7-|4-6-9-1|-3-2-8 and Child 2: 8-2-|9-7-5-3|-1-4-6

9.4 GENETIC ALGORITHM WITH PMX OPERATOR

Step #1:

Set k=1,

From the given data, create population of fixed size.

Step #2

Evaluate the sequences, by finding the values of given criteria. Arrange the sequences in ascending order, from best sequence to the worst.

Step #3

Select the top two sequences as parents. Call them parent 1 and parent 2 respectively.

Step #4

Apply cross-over operator PMX on the parents to generate two child sequences.

Step #5

Apply mutation on the generated children.

Step #6

Find values for the newly created (child) sequences.

Step #7

Replace these child sequences with poor sequences in the existing population, if it is feasible.

Step #8

If (k < N)

Go to Step #2. And k = k+1,

Else

Go to step #9,

Step #9

Stop.

The best sequence is lowest value in the population (at the top).

Example 9.1

The following problem represents an instance of 1 || L_{max}. Apply genetic Algorithm and find best sequence.

Job(j)	1	2	3	4	5
$\mathbf{p_j}$	3	4	6	10	2
$\mathbf{d_j}$	3	10	17	18	15

Take a population size of six with the following sequences as initial population.

1. $\{j_3, j_1, j_2, j_5, j_4\}$
2. $\{j_1, j_4, j_2, j_5, j_3\}$
3. $\{j_2, j_1, j_5, j_4, j_3\}$
4. $\{j_5, j_4, j_1, j_2, j_3\}$
5. $\{j_5, j_3, j_2, j_1, j_4\}$
6. $\{j_5, j_3, j_4, j_1, j_2\}$

For PMX cross over operator, take the 3rd and 4th sequence positions as cut-off positions.

For mutation, interchange the positions of the jobs within the cut-off positions.

Apply the genetic algorithm methodology for THREE iterations.

Solution:

Iteration #1

Find the L_{max} values of sequences in initial population.

Job(j)	3	1	2	5	4	
$\mathbf{p_j}$	6	3	4	2	10	
$\mathbf{d_j}$	17	3	10	15	18	$L_{max} = 7$
$\mathbf{C_j}$	6	9	13	15	25	
$\mathbf{L_j}$	-11	6	3	0	7	
Job(j)	5	3	4	1	2	
$\mathbf{p_j}$	2	6	10	3	4	
$\mathbf{d_j}$	15	17	18	3	10	$L_{max} = 18$
$\mathbf{C_j}$	2	8	18	21	25	
$\mathbf{L_j}$	-13	-9	0	18	15	

Job(j)	1	4	2	5	3	
$\mathbf{p_j}$	3	10	4	2	6	
$\mathbf{d_j}$	3	18	10	15	17	$L_{max} = 8$
$\mathbf{C_j}$	3	13	17	19	25	
$\mathbf{L_j}$	0	-5	7	4	8	
Job(j)	5	4	1	2	3	
$\mathbf{p_j}$	2	10	3	4	6	
$\mathbf{d_j}$	15	18	3	10	17	$L_{max} = 12$
$\mathbf{C_j}$	2	12	15	19	25	
$\mathbf{L_j}$	-13	-6	12	9	8	
Job(j)	2	1	5	4	3	
$\mathbf{p_j}$	4	3	2	10	6	
$\mathbf{d_j}$	10	3	15	18	17	$L_{max} = 8$
$\mathbf{C_j}$	4	7	9	19	25	
$\mathbf{L_j}$	-6	4	-6	1	8	
Job(j)	5	3	2	1	4	
$\mathbf{p_j}$	2	6	4	3	10	
$\mathbf{d_j}$	15	17	10	3	18	$L_{max} = 12$
$\mathbf{C_j}$	2	8	12	15	25	
$\mathbf{L_j}$	-13	-9	2	12	7	

Table 9.1 Calculation of L_{max} for all population

The values of L_{max} for the six sequences in ascending order are;

Sequence						$\mathbf{L_{max}}$
1	3	1	2	5	4	**7**
2	1	4	2	5	3	**8**
3	2	1	5	4	3	**8**
4	5	4	1	2	3	**12**
5	5	3	2	1	4	**12**
6	5	3	4	1	2	**18**

Table 9.2 Arranging the sequences in ascending order of L_{max}

Select the two least L_{max} value sequences as parents from Table 9.2. Apply the cross-over operator (PMX) to yield the two children as under:

Parent 1: 3 - 1 - | 2 - 5 | - 4
Parent 2: 2 - 1 - | 5 - 4 | - 3

Child 1: x - x - | 2 - 5 | - x
From the mapping, $2 \Leftrightarrow 5,\ 5 \Leftrightarrow 4,\ 2 \Leftrightarrow 4$
Hence, Child 1: 4 - 1 - | 2 - 5 | - 3

Similarly, find child 2 from cross over of Parent2 and Parent 1 as under;

Parent 2: 2 - 1 - | 5 - 4 | - 3
Parent 1: 3 - 1 - | 2 - 5 | - 4

Child 2: x - x - | 5 - 4 | - x
From the mapping, $2 \Leftrightarrow 5,\ 5 \Leftrightarrow 4,\ 2 \Leftrightarrow 4$
Hence, Child 2: 3 - 1 - | 5 - 4 | - 2

Now apply Mutation on child 1 and child 2; i.e., change the adjacent jobs within cut-off position.

	Before Mutation	After mutation
Child 1:	4 - 1 - | 2 - 5 | - 3	4 - 1 - | 5 - 2 | - 3
Child 2:	3 - 1 - | 5 - 4 | - 2	3 - 1 - | 4 - 5 | - 2

Calculations of L_{max} for sequences (4, 1, 5, 2, 3) and (3, 1, 4, 5, 2) are as under;

Job(j)	4	1	5	2	3	
$\mathbf{p_j}$	10	3	2	4	6	
$\mathbf{d_j}$	18	3	15	10	17	$L_{max} = 10$
$\mathbf{C_j}$	10	13	15	19	25	
$\mathbf{L_j}$	-8	10	0	9	8	
Job(j)	3	1	4	5	2	
$\mathbf{p_j}$	6	3	10	2	4	
$\mathbf{d_j}$	17	3	18	15	10	$L_{max} = 15$
$\mathbf{C_j}$	6	9	19	21	25	
$\mathbf{L_j}$	-11	6	1	6	15	

Table 9.3 Computation of L_{max} value for child sequences

Comparing L_{max} values of the two children with sequences in Table 9.2, it is found that these sequences may be replaced with worst valued sequences from Table 9.2 as, Sequences 5 and 6 will leave the population to make room for the two children sequences as follows:

Sequence						**L_{max}**
1	3	1	2	5	4	**7**
2	1	4	2	5	3	**8**
3	2	1	5	4	3	**8**
4	5	4	1	2	3	**12**
*****	4	1	5	2	3	**10**
*****	3	1	4	5	2	**15**

Arranging values of L_{max} for the sequences in ascending order are;

Sequence						**L_{max}**
1	3	1	2	5	4	**7**
2	2	1	5	4	3	**8**
3	1	4	2	5	3	**8**
4	4	1	5	2	3	**10**
5	5	4	1	2	3	**12**
6	3	1	4	5	2	**15**

Table 9.4 Arranging the sequences in ascending order of L_{max}

Iteration #2

Since, sequence 1 and 2 did not produce children better than parents, choose now sequence 1 and 3 for cross over and apply cross-over operator (PMX) as follows:

Parent 1: 3 - 1 - | 2 - 5 | - 4
Parent 2: 1 - 4 - | 2 - 5 | - 3

Child 1 : x - x - | 2 - 5 | - x
From the mapping, $2 \Leftrightarrow 5$
Hence, Child 1: 1 - 4 - | 2 - 5 | - 3

Similarly, find child 2 from cross over of Parent2 and Parent 1 as under;

Parent 2: 1 - 4 - | 2 - 5 | - 3
Parent 1: 3 - 1 - | 2 - 5 | - 4

Child 2: x - x - | 2 - 5 | - x
From the mapping, $2 \Leftrightarrow 5$
Hence, Child 2: 3 - 1 - | 2 - 5 | - 4

Now apply Mutation on child 1 and child 2; i.e., change the adjacent jobs within cut-off position.

	Before Mutation	After mutation
Child 1	1 - 4 - | 2 - 5 | - 3	1 - 4 - | 5 - 2 | - 3
Child 2	3 - 1 - | 2 - 5 | - 4	3 - 1 - | 5 - 2 | - 4

Job(j)	1	4	5	2	3	L_{max}=9
Job(j)	3	1	5	2	4	L_{max}=15

Table 9.5 Computation of L_{max} value for child sequences

The sequence (1, 4, 5, 2, 3) has better value of L_{max} than sequence 6 in Table 9.4. Hence, this sequence will replace sequence 6. New population is as under;

Sequence						**L_{max}**
1	3	1	2	5	4	**7**
2	2	1	5	4	3	**8**
3	1	4	2	5	3	**8**
4	1	4	5	2	3	**9**
5	4	1	5	2	3	**10**
6	5	4	1	2	3	**12**

Table 9.6 Arranging the sequences in ascending order of L_{max}

Iteration #3

Since, sequence 1 and 3 did not produce children better than parents, choose now sequence 1 and 4 for cross over and apply cross-over operator (PMX) as follows:

Parent 1: 3 - 1 - | 2 - 5 | - 4
Parent 2: 1 - 4 - | 5 - 2 | - 3

Child 1: x - x - | 2 - 5 | - x
From the mapping, $2 \Leftrightarrow 5$
Hence, Child 1: 1 - 4 - | 2 - 5 | - 3

Similarly, find child 2 from cross over of Parent 2 and Parent 1 as under;

Parent 2: 1 - 4 - | 5 - 2 | - 3
Parent 1: 3 - 1 - | 2 - 5 | - 4

Child 2: x - x - | 2 - 5 | - x
From the mapping, $2 \Leftrightarrow 5$
Hence, Child 2: 3 - 1 - | 5 - 2 | - 4

Now apply Mutation on child 1 and child 2; i.e., change the adjacent jobs within cut-off position.

	Before Mutation	After mutation
Child 1	1 - 4 - | 2 - 5 | - 3	1 - 4 - | 5 - 2 | - 3
Child 2	3 - 1 - | 5 - 2 | - 4	3 - 1 - | 2 - 5 | - 4

These two sequences have already been evaluated. Since, three iterations are complete, best sequence is (3, 1, 2, 5, 4) with $L_{max} = 7$

The order-based representation can be easily interpreted and applied to single machine and flow shop problems because both the single machine and the flow shop problems are permutation scheduling problems. However, a job shop problem is not a permutation scheduling problem and hence the order-based representation is not easily interpreted and applied to job shop problems. As a result of this difficulty, several variations of the order-based representation have been developed to handle the interpretation problem faced in the job shop implementations. These variations will be discussed in the next section.

As mentioned earlier, the population representations can be represented by various representations such as integer values. The integer value representation of population was suggested by Dorndorf and Pesch (1995). They proposed two GA applications to use this type of representation. In the first, the chromosomes were formed of genes which represented an integer value which corresponded to a

dispatching rule number from a given list of dispatching rules. The integer values in the second application depict a machine number. This means a chromosome was formed of genes each of which represented a machine number from a list of machines that were in the shop. In this representation, all classical crossovers will always produce feasible chromosomes. Also, all mutation methods that are applied to the ordinal representation method can be applied to this representation.

9.5 CONSTRAINED GENETIC ALGORITHM

In this section, an introduction will be given to the constrained genetic algorithm (CGA) which was developed by Al-Harkan and Foote (1994, 1996). The CGA was developed to address the single machine total weighted tardiness (TWT) problem which is strongly NP-hard. The proposed CGA approach obtained close to optimal solutions with much less deviation from optimal and much less computational effort than the conventional or unconstrained GA (UGA), which does not exploit the problem structure. This superior performance was achieved by combining sequencing and scheduling theory with the genetic algorithms methodology. Their approach can be called a hybrid GA, since it incorporates local search features in its procedures. However, they offered an additional feature that of constraining the order of certain elements of the chromosomes according to precedence relationships established theoretically. Hence, they called this approach a constrained GA.

This study was motivated by several scheduling problems that are classified as NP-hard problems which can be solved by using implicit enumerative methods which are branch and bound (B&B) and dynamic algorithm (DA). One of these problems is the total weighted tardiness. For large-sized problems, B&B and DA will take a long time to find the optimal solution; also, the time required by the B&B is unpredictable. Hence, these implicit enumeration methods are only efficient when time is not considered a factor. When faced with this reality, a search for a substitution method that is efficient and gives good results was the next alternative. Several methods have been found to solve such NP-hard problems: one of them is the GA approach. Researchers claim that GAs give fairly good and close to optimal solutions faster than the implicit enumeration methods.

Wainwright expanded on that where he stated: The GAs are a robust search technique that will produce “close” to optimal results in a “reasonable” amount of time.... The GAs should be used when a good fitness function is available; when it is feasible to evaluate each potential solution; when a near-optimal, but not optimal solution is acceptable; and when the state-space is too large for other methods (Wainwright 1993, 12-13).

Also, Koulamas, Antony, and Jaen elaborated on the robustness of these search techniques: OR researchers are increasingly turning towards new solution techniques such as neural networks, genetic algorithms, simulated annealing, and tabu search to solve management science problems. These techniques can be used as heuristics for finding near optimal solutions to a problem, and serve as alternatives to problem specific heuristics.... Typically, these techniques have been successfully applied to NP-hard problems. (Koulamas, Antony, and Jaen 1994, 41)

Unconstrained Genetic Algorithm

The Unconstrained Genetic Algorithm (UGA) parameters were selected according to pilot runs that were done previously. These parameters are: the population size; the number of generations; the generation of the initial population; the selection methods; the reproduction methods (crossover and mutation), and termination criterion. The population size and the number of generations are determined as a function of the problem size (i.e., the number of jobs). The initial population for the UGA was randomly generated. Two selection methods were used in this study. The first method was the elitist method, which enforces preserving the best chromosomes in the reproduction process. Thus, at each generation the elitist method will be used to move a fraction of the population to the next generation. The second was a variant of the binary tournament that was suggested by Norman and Bean (1994). The variant method is performed by first randomly selecting two chromosomes from the population. Then the genetic operators are applied to these two chromosomes. Next, the best of the two produced chromosomes will be selected and allowed to enter the pool of the potential chromosomes for the next generation. The tournament procedures will be repeated until a new generation of chromosomes is produced. The linear order crossover (LOX) and order-based mutation (OBM) were used as the genetic operators. The UGA terminated its procedures when the maximum number of generations had been reached.

Constrained Genetic Algorithm

The section will give a discussion of the proposed constrained genetic algorithm (CGA). In the UGA, a random population of feasible sequences was generated to be used as an initial population. This starting initial population will affect the quality of solutions and the time taken to obtain the solution. This claim was the conclusion of a sensitivity study that will be discussed later. Hence, this step can be improved by using one of the heuristics that solve for the TWT. Three heuristics were used to generate three of the initial sequences. These three heuristics are the SPT, the EDD, and the ATC. Thus, when the CGA was implemented, three chromosomes were generated according to the SPT, the EDD, and the ATC

heuristics. The rest of the population was randomly generated to avoid the bias that might be caused by the three heuristics.

As mentioned earlier, the OBM procedures were to select two jobs at random and swap them; however, swapping these two jobs could fail to satisfy standard dominance conditions of the TWT problem. Hence, dominance rules can be used to avoid dominated swapping of jobs, and so better objective values can be obtained. Two theorems can be used as dominance rules for the TWT problem. These are:

Rule 1: For two jobs j and k, if

$$P_j \le P_k,\ d_j \le d_k, \text{ and } W_j \ge W_k,$$

then there exists an optimal sequence in which job j appears before job k.

Rule 2: If there exists a job k that satisfies

$$d_k \ge \sum_{j=1}^{n} P_j$$

then there exists an optimal sequence in which job k is assigned the last position in the sequence.

The dominance rules were implemented on the children produced by the LOX operator by ordering the set of jobs located in the segment between the crossover positions according to a precedence constraint based on the dominance rule. The motivation behind only ordering the jobs in the crossover block was to avoid the bias that might be caused if the whole chromosome was sorted, which would tend to create a whole set of chromosomes that were similar, tending to localize the search. Further, sorting the whole chromosome is time-consuming. These two conjectures are the conclusions of a sensitivity test study that will be discussed in the following section. The UGA approach was modified to adopt all mentioned improvements, which resulted in the CGA approach. For detailed explanations for both the UGA and the CGA, the reader can refer to Al-Harkan and Foote (1994, 1996).

9.6 APPLICATIONS OF GENETIC ALGORITHMS TO SCHEDULING PROBLEMS

In this section, a listing of most of the genetic algorithm (GA) studies that have been applied to all sequencing and scheduling problems will be given. However, since the focus is on the job shop problem, the GAs that has been applied to job shop problems will be discussed in more depth.

The GAs were applied to single machine problems by Liepins et al. (1987), Gupta, Gupta, and Kumar (1993), Lee and Choi (1995), Lee and Kim (1995), and Rubin and Ragatz (1995). Liepins et al. (1987) applied a GA approach to minimize lateness. In their study, they compared the performance of three crossover methods

(PMX, greedy weak crossover heuristics, and greedy powerful crossover heuristic). They concluded that PMX dominated both crossover methods. Gupta, Gupta, and Kumar (1993) tried to minimize flow time variance using a GA approach. In their study, they tested the effect of the GA parameters, which were population size, number of generations, problems size, crossover rate, and mutation rate. They found that most of these parameters have significant effects on the GA approach--especially the population size and the number of generations. Only the crossover rate had an insignificant effect. Lee and Choi (1995) applied a GA approach to solve a single machine problem where the total earliness and tardiness penalties was minimized. Lee and Kim (1995) developed a parallel GA to solve a single machine, using a common due date where the weighted sum of the total earliness and total tardiness was minimized. A GA approach to handle sequence dependent set-up time has been applied by Rubin and Ragatz (1995) where the total tardiness was minimized.

Cleveland and Smith (1989) used a GA approach to solve a flow shop problem where the total flow time was minimized. Neppalli (1993) tested the effect of the genetic parameters on the GA approach, using both total flow time and the makespan as performance measures. Neppalli concluded that the application of GAs are problem dependent, and the non-random initial population has a significant effect on the GA convergence. A GA approach was used to minimize the C_{max} in flow shop problems by Stöpller and Bierwirth (1992), Vempati, Chen, and Bullington (1993), Sridhar and Rajendran (1994), Chen, Vempati, and Aljaber (1995), and Reeves (1995). Stöpller and Bierwirth (1992) developed a parallel GA to the solve the flow shop problem. Reeves (1995) compared GA and simulated annealing, and found that when the problem is small, the two are comparable, but as the problem gets bigger, the GA performs better.

Davis (1985) was the first to apply GAs to job shop problems. However, he was not the only one. Several researchers have been attempting to solve the job shop problem using GAs. These attempts were made by Bagchi et al. (1991), Falkenauer and Bouffouix (1991), Nakano and Yamada (1991), Fang, Ross, and Corne (1993), Gen, Tsumjimura, and Kubota (1994), Norman and Bean (1994), Bierwirth (1995), Bierwirth, Kopfer, Mattfel, and Rixen (1995), Kobayashi, Ono, and Yamamura (1995), Croce, Tadei, and Volta (1995), Dorndorf and Pesch (1995), and Mattfeld (1996).

Davis (1985) presented a conceptual and instructional study to show how the GA can be applied to job shop. Davis attempted to solve a job shop problem, using an indirect representation of the population which allows the use of Holland's crossover operator. Davis represented a chromosome as a preference list of operations where the chromosome is time dependent and machine controlling. Each machine has a list of these chromosomes, which are activated sequentially as time passes. Davis's representation of each chromosome has four elements. The first

element is the activation time of the chromosome. The second element is a preference list of operations, and the third and fourth elements are keys to control the machine, which are 'wait' and 'idle'. However, for reasons that have been reported by several researchers, Davis's work can be summarized by the following statements:

> ...The performance of the Davis-style approach in initial runs on Problem 1 was not particularly notable. Some improvement was observed over time, but the final solution obtained was not as good as that obtained by the standard-GA. (Cleveland and Smith 1989, 167)
>
> ...Davis (1985) uses an intermediate representation which is guaranteed to produce legal schedule when operated upon by genetic recombination operators. However, the example used is not very complicated, and there are no significant results. (Bagchi et al. 1991, 11)
>
> ...Davis (1985) discusses a more indirect encoding that permits the use of the traditional crossover operator. For this encoding, a chromosome consists of a sequence of job preferences combined with times at which these job preferences become active. However, this encoding suffers from inflexibility due to the need to determine an appropriate time scale and appropriate machine idle and waiting time periods. (Norman and Bean 1994, 6)
>
> ...Davis (1985) presented an application of genetic search to a simple job shop scheduling problem. The focus of the paper was on developing a workable representation of the problem. Only a single example problem was presented, with very limited computational experience. (Rubin and Ragatz 1995, 87)

Bagchi et al. (1991) developed and implemented a GA approach to solve a job shop problem. They designed a hypothetical job shop that had three machines and could process three products. The eleven orders produced by the job shop were orders for one of three products with a specific batch size. Each of the three products had several alternative process plans, including three process plans for product one, and two process plans for products two and three. All the process plans had three operations except one. All operations could be processed by two alternative machines except two of them were processed by only one alternative machine.

In their study, Bagchi et al. used three representations of the population which are variants of the order-based representation. The first representation is a simple order-based representation, but the second and third representations are known as problem-specific-based representation. In the first representation, each gene in a chromosome represented the order priority. A chromosome in the second representation was formed by genes that had two elements. The first element of a gene was the order priority, and the second was the process plan assigned to the

order. The third representation was the same as the second representation; however, the third representation was more specific than the second representation. In the second element of the gene, the third representation not only assigns a process plan to an order, but also specifies the machines to perform the operations in the process plan assigned. Bagchi et al. compared the three representations using machine utilization as the performance measure and found that the third representation was superior. The major conclusion of their study was:

> ...To enhance the performance of the algorithm and to expand the search space, a chromosome representation which stores problem-specific information is devised. (Bagchi et al. 1991, 10)

Falkenauer and Bouffouix (1991) solved a job shop problem using a GA approach where jobs had different release times. Falkenauer and Bouffouix used an order-based representation version which is known as preference-list-based representation. In this representation a chromosome is formed by several sub-chromosomes. These sub-chromosomes contain genes which represent the preference list for a specific machine. Each gene in the sub-chromosome represents an operation to be performed on that machine. For example, if there are three machines in the job shop, then there will be three sub-chromosomes in a chromosome. Also, if each machine performs five operations, there will be five genes in each sub-chromosome. In their implementation, each chromosome was evaluated, using a simulation model for the problem under consideration. The LOX and PMX were used as the crossover operators and inversion was the mutation operator. Each of these crossover methods was implemented on two chromosomes by crossing the first sub-chromosome of one parent with the first sub-chromosome of the other parent, the second with the second, and likewise until the last with the last.

Falkenauer and Bouffouix performed their experiment using three job shop models which they called small, big, and giant. The small model had 24 operations, the big had 60 operations, and the giant had 250 operations. In their GA approach they maximized the difference of the summation of weighted earliness and the summation of squared tardiness where the earliness was given a weight between 0 and 1. Falkenauer and Bouffouix used a pilot study to determine the GA parameters. From the pilot study, they fixed the following parameters: crossover rate was 0.6; mutation rate was 0.033; the population size was 30, and the number of generations was 100. To evaluate the performance of the GA, they used the following dispatching rules: SPT and JST. Falkenauer and Bouffouix performed ten replicates for each model mentioned above. From their results, they concluded the following: the GA is superior when compared to the dispatching rules, and LOX performed better than PMX.

Nakano and Yamada (1991), as mentioned in the previous section, developed a GA approach to solve job shop problems using binary representation of the

population. The classical crossover and mutation operators were applied as they were by Holland. They evaluated their chromosomes using semi-active schedules. In their experiment they solved three well-known problems designed by Fisher and Thompson (1963). From their results, it was clear that their GA approach obtained results comparable to the results obtained by other approaches.

Fang, Ross, and Corne (1993) and Gen, Tsumjimura, and Kubota (1994) implemented GA approaches that utilized a variant of an order-based representation known as operation-based representation. In this representation a chromosome is formed by genes which represent an integer value which corresponds to a job number. In each chromosome, a job's number will be repeated according to its number of operations. Therefore, a chromosome becomes a sequence of operations for all jobs. For example, if there are three machines and three jobs in the job shop and all jobs go through all machines, then there will be 9 genes in a chromosome as follows: 3-1-1-3-2-3-2-1-2, where the first 3 stands for operation 1 of job 3, the first 1 stands for operation 1 of job 1, the second 1 stands for operation 2 of job 1, and likewise until the third 2 stands for operation 3 of job 2. In the chromosome given, each job was repeated three times because each of them had three operations. The given chromosome can be interpreted when the process plan of each job is given. Hence, assume that the process plans for jobs 1, 2, and 3 are as follows: 1-2-3, 1-3-2, and 2-1-3 respectively (where numbers in the process plans indict the machine number). Then, the chromosome above can be interpreted as follows: job 3 is processed first at machine 2, job 1 is processed first at machines 1 and 2, job 3 is processed second at machine 1, job 2 is processed third at machine 1, job 3 is processed first at machine 3, job 2 is processed second at machine 3, job 1 is processed third at machine 3, and job 2 is processed third at machine 2.

Gen, Tsumjimura, and Kubota (1994) implemented their GA approach to solve a job shop problem where the makespan was minimized. In their implementation, each chromosome was evaluated using deterministic Gantt charting. Specifically, for each chromosome, they constructed a semi-active schedule. Gen, Tsumjimura, and Kubota developed their own crossover operator which they named partial schedule exchange crossover (for detailed explanations for the developed crossover operator, the reader can refer to Gen and Cheng 1997). They developed their own crossover method because all the other crossover methods that can be applied to the order-based representation cannot be applied to operation-based representation. The OBM was used as the mutation operator and the elitist method was used as the production method. Dynamic population size was utilized where at the end of each generation the population size was increased by a percent of the summation of mutation and crossover rates. Then, the population size was reduced to the original size, where only the best individuals were selected from the inflated population size. Gen, Tsumjimura, and Kubota solved three well-known benchmarks

from Fisher and Thompson (1963). In their experiment, they used the following parameters: crossover rate was 0.4; mutation rate was 0.3; the population size was 60, and the number of generations was 5000. They compared their results to branch and bound approaches and other GAs. From their results, it is clear that they performed better than the other GAs but not better than branch and bound approaches.

Norman and Bean (1994) performed a study in which they developed and implemented a GA approach to solve a job shop problem using a random key representation method. They designed the GA approach to solve a job shop with m machines and n jobs where these jobs arrive at the job shop separately. Also, setup times were sequence dependent, and machine down time and scarce tools were incorporated. The GA approach was applied to the described job shop model to minimize the total tardiness. In the GA implementation, the elitist method, which enforces preserving the best chromosomes, was used in the reproduction process. A variant of the binary tournament was used to select two chromosomes to reproduce. Uniform crossover and immigration mutation were the two genetic operators used. In every generation, the immigration mutation method inserted a new random chromosome. By using the immigration mutation, the study tried to eliminate the effect of the elitist reproduction, which causes premature convergence. In this study, the GA approach terminates if the best solution found has not changed for 15 generations.

Norman and Bean incorporated problem specific data to enhance the performance of the GA approach by using ready times and due dates to prioritize jobs. They stated:

The scheduling application incorporates problems specific into the random keys encoding to improve the rate of convergence. Recall that for the general random keys encoding the random keys for all the genes are uniform (0,1) variates. The scheduling application contains problem specific data which can be used to bias the random key values of the jobs. If the problem objective is to minimize total tardiness then it is likely that jobs that have early ready and due times will be found early in the optimal sequence. Likewise, jobs with late ready and due times will probably be found late in the optimal sequence. (Norman and Bean 1994, 13)

The enhancement incorporated in their model was performed when the chromosomes were generated. That is, if job 5 has to be before job 2 in the optimal sequence, the uniform random number assigned to job 2 will be biased to be large (for example, the random number for job 2 will be uniformly distributed between 0.8 and 1 instead of being uniformly distributed between 0 and 1). By doing so, job 2 will usually be located in later positions in the sequence. On the other hand, job 5 will be assigned a smaller random number which will often locate it in earlier positions. The example given by Norman and Bean was not a good example to demonstrate the data specific enhancement. In addition, they did not give any explanations of how to

handle difficult situations. This enhancement does not incorporate job processing times, which does not make it robust enough. The reason for not being robust enough is that their objective function, total tardiness, is a function of ready times, due dates, and processing times. Also, this enhancement is performed only on the initial population and not during the evolution process. This implies that this enhancement is predictive and not reactive.

Norman and Bean performed an elementary testing by solving three types of data sets. The first consisted of a single machine and 16 jobs. The second set had seven problems which each contained two machines and 350 jobs. Five problems were in the third data set, with each problem having ten machines and 250 jobs. For the first data set, ten replications were performed and the GA approach was able to obtain the optimal solution provided by Kanet and Sridharan (1991). They concluded that the results of all the data sets were encouraging, and claimed that the GA approach was good in solving the job shop problem.

Bierwirth (1995) developed a GA approach (GP-GA) to solve a job shop problem using an operation-based representation where the makespan was minimized. In the GP-GA, each chromosome was evaluated according to an active schedule. As mentioned earlier, all the crossover methods that can be applied to an order-based representation cannot be applied to operation-based representation. Therefore, Bierwirth developed a crossover method which is a generalization of OX (GOX). In the conducted experiment, the following parameters were used: the population size was 100, and two levels of the number of generations were 100 and 150. Ranking selection method was used to select chromosomes to reproduce. Bierwirth solved twelve standard problems which were designed by Fisher and Thompson (1963) and Lawrence (1984). Bierwirth performed a total of 100 replicates for the two problems that were designed by Fisher and Thompson and 25 replicates for the other ten problems that were designed Lawrence (1984). From the results obtained, Bierwirth reported that the average solutions for all problems were within a percentage of deviation of errors that ranged between 0.7% and 7%. Also, Bierwirth concluded that the GP-GA was a promising approach. Bierwirth, Kopfer, Mattfel, and Rixen (1995) performed a preliminary study in which they extended the GP-GA to solve dynamic job shop problem where jobs had different release times.

Croce, Tadei, and Volta (1995) developed a GA approach to solve a job shop problem using a preference-list-based representation that was developed by Falkenauer and Bouffouix (1991). In their implementation, each chromosome was evaluated using a simulation model for the problem considered. Croce, Tadei, and Volta claimed that schedules produced by the simulation model were only non-delay schedules. Hence, they constructed schedules with look-ahead function to introduce delay. The look-ahead function used by Croce, Tadei, and Volta violated the definition of non-delay schedule to a certain extent so that some of the delay

schedules could be incorporated in the final solution. The look-ahead function was accomplished as follows: when a machine finishes processing and becomes available to process the operations waiting for it, an operation with the highest priority will be scheduled to be processed. However, before scheduling this operation, the look-ahead function will first determine the processing time and the completion time of the candidate operation. Then, the look-ahead function will check to see if there is an operation which will arrive before the candidate operation finishes and has higher priority than the candidate operation. If there is an operation that satisfies both conditions, then the machine will stay idle until the new operation arrives. Otherwise, the candidate will be scheduled.

The LOX was the crossover method used by Croce, Tadei, and Volta. The OBM was applied by swapping genes within a sub-chromosome. The steady-state reproduction was the reproduction method used, where at each generation a number of new chromosomes were inserted. Croce, Tadei, and Volta performed a pilot study to determine the GA parameters. From the pilot study, they fixed the following parameters: crossover rate was 1; mutation rate was 0.03; the population size was 300, and ten new chromosomes were inserted at each generation for the reproduction method. Croce, Tadei, and Volta applied the GA approach developed to minimize the makespan using eleven standard problems by performing five runs for each of them. Three of these problems were designed by Fisher and Thompson (1963), and the other eight were designed by Lawrence (1984). The optimal solutions for these problems were provided by Fisher and Thompson (1963), and Lawrence (1984). Croce, Tadei, and Volta obtained the results for the eleven problems and compared the best obtained result for each problem with the best obtained results of three other studies which had solved the same eleven problems. One of these studies which solved the eleven problems by the simulated annealing (SA) algorithm was performed by Laarhoven, Aarts, and Lenstra (1992). The second study was performed by Dell'Amico and Trubian (1993) who solved the eleven problems using the tabu search (TS) approach. The Shifting Bottleneck (SB) algorithm (Adams, Balas, and Zawack 1988) was the third heuristic that also was used to solve the eleven problems. From the results of this study and the other three studies, it is clear that the tabu search approach was superior. Out of the eleven problems, the TS converges to the optimal solution in ten problems. The SA approach found the optimal solution to 8 problems. The SB and GA found the optimal solutions to 7 and 6 problems respectively.

As mentioned earlier, Dorndorf and Pesch (1995) proposed a GA approach that used an integer value representation of population which was used to solve a job shop problem where the makespan was minimized. Recall from the previous section that they proposed two GAs, which they named P-GA and SB-GA. In the P-GA, each chromosome consisted of n-1 genes where n-1 is the number of operations in the

problem under consideration. Each gene was represented by an integer value which corresponded to a dispatching rule number from a list of twelve dispatching rules (SPT, LPT, LRPT, SRPT, RANDOM, FCFS, TWORK, TLPT, MWR, LWR, longest operation successor, and longest operation reaming processing time). This implies that each gene can have an integer value between 1 and 12. In the P-GA, the schedules were constructed using an active schedule algorithm which was developed by Giffler and Thompson (1960). At each iteration of Giffler and Thompson's algorithm a conflict set of operations is formed which can have one or more operations. From the conflicting set of operations, an operation is selected randomly or by using a single dispatching rule. Hence, this selection problem motivated Dorndorf and Pesch to developed their P-GA approach, which was used to solve the conflict in selecting an operation. The selection of an operation was performed by referring to the gene that was associated with this operation, and this gene would prioritize this operation according to the relevant dispatching rule.

In the second application, the developed GA approach (SB-GA) was part of the shifting bottleneck (SB) algorithm. Recall that the SB algorithm sequences machines sequentially, one at a time until all machines are sequenced. It should be clear that the sequence of machine selection affect the quality of solutions obtained. Again, the selection problem motivated Dorndorf and Pesch to develop the SB-GA approach which controlled the machine selection at the first step of the SB algorithm. Each chromosome in the SB-GA approach consisted of m genes where m is the number of machines in the job shop. Each gene represented a machine number which could have any value between 1 and m.

Dorndorf and Pesch used three well-known benchmarks by Fisher and Thompson (1963) to tune their parameters. They used the elitist method in both GA approaches. For the P-GA, they used the following parameters: crossover rate was 0.65; mutation rate was 0.001; inversion rate was 0.7, and the population size was 200. In the SB-GA, mutation and inversion were not implemented, the crossover rate was 0.75, and the population size was 40. Dorndorf and Pesch randomly generated and solved 105 problems by the P-GA and the SB-GA, and then compared the results obtained to the results of four other heuristics. These were: a random selection; dispatching rules, and two versions of the SB algorithm. Also, they solved 40 problems that were designed by Lawrence (1984). Then they concluded that with respect to the makespan, the SB-GA performed better than the SB and the other heuristics. However, in terms of CPU time, the SB performed better than all heuristics. On the other hand, the SB algorithm dominated the P-GA approach in both time and objective function. The improvement gained by using the SB-GA over the SB algorithm was on the average very small. Also, the CPU time needed by SB-GA was increased by a huge percentage in both small and large problems.

Kobayashi, Ono, and Yamamura (1995) implemented a GA approach to solve a job shop problem where chromosomes were represented using the preference-list-based representation. In their implementation, each chromosome was evaluated using an active schedule. The OX and sub-sequence exchange crossover (SXX) were used as the crossover methods and mutation was not applied. Kobayashi, Ono, and Yamamura tuned their GA with two well-known benchmarks which were designed by Fisher and Thompson (1963). From the pilot study, they fixed the following parameters: crossover rate was 1.0, and the population size was 600. Random selection without replacement was used to select chromosomes. In their final experiment, they performed a total of 100 replicates for Fisher and Thompson's problems and they concluded that SXX performed better than OX, and the GA approach developed was promising.

Mattfeld (1996) developed three GA approaches to solve the job shop problem using operation-based representation. In all the GAs developed (GA1, GA2, and GA3), each chromosome was evaluated using a semi-active schedule, then the resultant schedule was re-optimized using a hill climbing algorithm. Also, a proportional selection method was used. Using GA1, they compared three mutation operators, PBM, OBM, and SBM, and concluded that PBM was the best. Also, using GA1, they compared two crossover operators, GOX and a developed version of PBX (called GPX). The conclusion of the second experiment was that the GOX was superior. Also, Mattfeld performed an experiment where the GA1 was compared with pure GA. The pure GA used neither semi-active schedules nor hill climbing algorithm. Then he concluded that GA1 achieved better results than the pure GA in fewer generations. The parameters used in the GA1 implementation were as follows: crossover rate was 0.6; mutation rate was 0.03; the population size was 100; the number of generations was 100, and the number of neighbors was 100. Using the parameters mentioned, Mattfeld solved twelve benchmarks to evaluate the performance of the GA1. Two of these problems were designed by Fisher and Thompson (1963), three of them were designed by Adams, Balas, and Zawack (1988), and the other seven were designed by Lawrence (1984). From the results obtained, Mattfeld reported that the average percentage error of deviation ranged between 1.3% and 4.8%. In the GA2, Mattfeld (1996) introduced structured population GA. Using the same parameters except for the crossover rate was 1, and the number of neighbors was 4, using a population structure of 10x10. In the GA2, Mattfeld used an acceptance criterion to either accept or reject the replacement of a parent by its offspring. The same twelve problems were solved by the GA2 and Mattfeld reported that the average percentage of errors ranged between 0.4% and 1.1%. The GA3 used the same parameters used by GA2, except the crossover and mutation rates were auto-adaptive. When the same twelve problems were solved by the GA3, the percentage of errors ranged between 0.3% and 1%.

EXERCISES

9.1 Consider $1 \| \sum T_j$ problem for the data given in the following Table

jobs	1	2	3	4
p_j	10	8	12	6
d_j	7	5	8	3

Solve the problem by using Genetic Algorithm method. Start with a population of the three sequences (3, 4, 1, 2), (4, 3, 1, 2) and (3, 2, 1, 4) perform three iterations. Apply cross-over operator PMX on the parents to generate two child sequences

9.2 Consider the instance of $P_2 \| \sum w_j T_j$ with following 6 jobs

jobs	1	2	3	4	5	6
p_j	13	9	13	10	8	11
d_j	6	18	10	11	13	18
w_j	2	4	2	5	4	3

Solve the problem using Genetic Algorithm. Using the initial population generated by SPT, LPT, EDD and WSPT. Perform THREE iterations and find the best sequence.

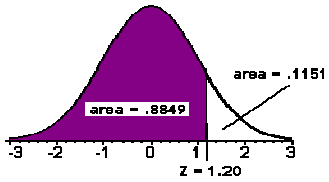
9.3 Consider the instance of $F_3 \mid p_{ij} = p_i \mid \sum w_j T_j$ for following problem

jobs	1	2	3	4
p_j	9	9	12	3
d_j	10	8	5	28
w_j	14	12	1	12

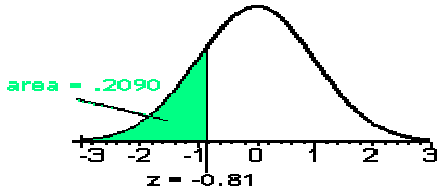
Apply the Genetic Algorithm to the instance. Start with a population of the three sequences (2, 4, 1, 3), (1, 3, 4, 2) and (1, 2, 3, 4) perform three iterations.

Appedix A

Normal Distribution Table



Z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
2.6	0.9953	0.9955	0.9956	0.9957	0.9959	0.9960	0.9961	0.9962	0.9963	0.9964
2.7	0.9965	0.9966	0.9967	0.9968	0.9969	0.9970	0.9971	0.9972	0.9973	0.9974
2.8	0.9974	0.9975	0.9976	0.9977	0.9977	0.9978	0.9979	0.9979	0.9980	0.9981
2.9	0.9981	0.9982	0.9982	0.9983	0.9984	0.9984	0.9985	0.9985	0.9986	0.9986
3.0	0.9987	0.9987	0.9987	0.9988	0.9988	0.9989	0.9989	0.9989	0.9990	0.9990
3.1	0.9990	0.9991	0.9991	0.9991	0.9992	0.9992	0.9992	0.9992	0.9993	0.9993
3.2	0.9993	0.9993	0.9994	0.9994	0.9994	0.9994	0.9994	0.9995	0.9995	0.9995
3.3	0.9995	0.9995	0.9995	0.9996	0.9996	0.9996	0.9996	0.9996	0.9996	0.9997
3.4	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9997	0.9998



z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
-3.6	0.00016	0.00015	0.00015	0.00014	0.00014	0.00013	0.00013	0.00012	0.00012	0.00011
-3.5	0.00023	0.00022	0.00022	0.00021	0.00020	0.00019	0.00019	0.00018	0.00017	0.00017
-3.4	0.00034	0.00032	0.00031	0.00030	0.00029	0.00028	0.00027	0.00026	0.00025	0.00024
-3.3	0.00048	0.00047	0.00045	0.00043	0.00042	0.00040	0.00039	0.00038	0.00036	0.00035
-3.2	0.00069	0.00066	0.00064	0.00062	0.00060	0.00058	0.00056	0.00054	0.00052	0.00050
-3.1	0.00097	0.00094	0.00090	0.00087	0.00084	0.00082	0.00079	0.00076	0.00074	0.00071
-3.0	0.00135	0.00131	0.00126	0.00122	0.00118	0.00114	0.00111	0.00107	0.00103	0.00100
-2.9	0.00187	0.00181	0.00175	0.00169	0.00164	0.00159	0.00154	0.00149	0.00144	0.00139
-2.8	0.00256	0.00248	0.00240	0.00233	0.00226	0.00219	0.00212	0.00205	0.00199	0.00193
-2.7	0.00347	0.00336	0.00326	0.00317	0.00307	0.00298	0.00289	0.00280	0.00272	0.00264
-2.6	0.00466	0.00453	0.00440	0.00427	0.00415	0.00402	0.00391	0.00379	0.00368	0.00357
-2.5	0.00621	0.00604	0.00587	0.00570	0.00554	0.00539	0.00523	0.00508	0.00494	0.00480
-2.4	0.00820	0.00798	0.00776	0.00755	0.00734	0.00714	0.00695	0.00676	0.00657	0.00639
-2.3	0.01072	0.01044	0.01017	0.00990	0.00964	0.00939	0.00914	0.00889	0.00866	0.00842
-2.2	0.01390	0.01355	0.01321	0.01287	0.01255	0.01222	0.01191	0.01160	0.01130	0.01101
-2.1	0.01786	0.01743	0.01700	0.01659	0.01618	0.01578	0.01539	0.01500	0.01463	0.01426
-2.0	0.02275	0.02222	0.02169	0.02118	0.02067	0.02018	0.01970	0.01923	0.01876	0.01831
-1.9	0.02872	0.02807	0.02743	0.02680	0.02619	0.02559	0.02500	0.02442	0.02385	0.02330
-1.8	0.03593	0.03515	0.03438	0.03362	0.03288	0.03216	0.03144	0.03074	0.03005	0.02938
-1.7	0.04456	0.04363	0.04272	0.04181	0.04093	0.04006	0.03920	0.03836	0.03754	0.03673
-1.6	0.05480	0.05370	0.05262	0.05155	0.05050	0.04947	0.04846	0.04746	0.04648	0.04551
-1.5	0.06681	0.06552	0.06425	0.06301	0.06178	0.06057	0.05938	0.05821	0.05705	0.05592
-1.4	0.08076	0.07927	0.07780	0.07636	0.07493	0.07353	0.07214	0.07078	0.06944	0.06811
-1.3	0.09680	0.09510	0.09342	0.09176	0.09012	0.08851	0.08691	0.08534	0.08379	0.08226
-1.2	0.11507	0.11314	0.11123	0.10935	0.10749	0.10565	0.10383	0.10204	0.10027	0.09852
-1.1	0.13566	0.13350	0.13136	0.12924	0.12714	0.12507	0.12302	0.12100	0.11900	0.11702
-1.0	0.15865	0.15625	0.15386	0.15150	0.14917	0.14686	0.14457	0.14231	0.14007	0.13786
-0.9	0.18406	0.18141	0.17878	0.17618	0.17361	0.17105	0.16853	0.16602	0.16354	0.16109
-0.8	0.21185	0.20897	0.20611	0.20327	0.20045	0.19766	0.19489	0.19215	0.18943	0.18673
-0.7	0.24196	0.23885	0.23576	0.23269	0.22965	0.22663	0.22363	0.22065	0.21769	0.21476
-0.6	0.27425	0.27093	0.26763	0.26434	0.26108	0.25784	0.25462	0.25143	0.24825	0.24509
-0.5	0.30853	0.30502	0.30153	0.29805	0.29460	0.29116	0.28774	0.28434	0.28095	0.27759
-0.4	0.34457	0.34090	0.33724	0.33359	0.32997	0.32635	0.32276	0.31917	0.31561	0.31206
-0.3	0.38209	0.37828	0.37448	0.37070	0.36692	0.36317	0.35942	0.35569	0.35197	0.34826
-0.2	0.42074	0.41683	0.41293	0.40904	0.40516	0.40129	0.39743	0.39358	0.38974	0.38590
-0.1	0.46017	0.45620	0.45224	0.44828	0.44433	0.44038	0.43644	0.43250	0.42857	0.42465
-0.0	0.50000	0.49601	0.49202	0.48803	0.48404	0.48006	0.47607	0.47209	0.46811	0.46414

REFERENCES

- Adams, J., Balas, E., and Zawack, D. 1988. "The Shifting Bottleneck Procedure for Job Shop Scheduling." Management Science 34(3), pp. 391-401.
- Al-Harkan, I. and Foote, B. L. "Genetic Algorithms and Simulated Annealing Algorithm: Unobserved Potential." Working Paper, School of Industrial Engineering, University of Oklahoma, Norman, May-1994.
- Al-Harkan, I. and Badiru, A., "Knowledge-Based Approach to machine sequencing", Engineering Design and automation 1(1), 1995, pp. 43-58.
- Al-Harkan, I. and Foote, B. L. "On Merging Optimality Conditions with Genetic Algorithms: An Experimental Investigation." Working Paper, School of Industrial Engineering, University of Oklahoma, Norman, May-1996.
- Al-Harkan, I. and Foote, B. L. "The Constrained and the Unconstrained Genetic Algorithms Computer Codes for Job Shop Scheduling Problems." Technical Report, School of Industrial Engineering, University of Oklahoma, Norman, May-1997.
- Al-Harkan, I. M., "On Merging Sequencing and Scheduling Theory with Genetic Algorithms to Solve Stochastic Job Shops", Dissertation of doctor of philosophy, University of Oklahoma, 1997.
- Al-Harkan, I. and Foote, B., "Flow Shop group technology manufacturing cell scheduling with fixed cycle time and zero buffer", Engineering Design and Automation 4(4), 1998, 233-247.
- Al-Harkan, I. "On Merging optimality conditions with genetic Algorithm Operators to Solve Single Machine Total Weighted Tardiness Problem", Journal of Systems Science and Systems Engineering, Volume 14, number 2, June 2005, pp. 187 – 205.
- Al-Harkan, I. "New Heuristics to Minimize the Mean Flow Time for Static Permutation Flowshop Problems" Journal of King Saud University (Engineering Science), Volume 18, No 1. (2006/1426)
- Anderson, E, J, and Nyirenda, J. C. 1990. "Two New Rules to Minimize Tardiness in a Job Shop." International Journal of Production Research 28(12), pp. 2277-2292.
- Arumugam, V. and Ramani, S. 1980. "Evaluation of Value Time Sequencing Rules in a Real World Job-Shop." Journal of Operational Research Society 31, pp. 895-904.
- Ashour, S. and Vaswani, S. D. 1972. "A GASP Simulation Study of Job-Shop Scheduling." Simulation 18(1), pp. 1-10.
- Bagchi, S., Uckun, S., Miyabe, Y., and Kawamura, K. 1991. "Exploring Problem-Specific Recombination Operators for Job Shop Scheduling." In Belew, R. and Booker, L. Editors Proceedings of the Fourth International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 10-17.

- Bahouth, S. B. 1991. Experimental Investigation On The Johnson Rule for Sequencing Jobs in a Two-Bottleneck Job Shop. Ph.D. Dissertation, The University of Oklahoma.
- Bahouth, S. B. and Foote, B. L. 1994. "Managing a Two-Bottleneck Job Shop with a Two-Machine Flow Shop Algorithm." International Journal of Production Research 32(10), pp. 2463-2477.
- Baker, C. T. and Dzielinski, B. P. 1960. "Simulation of a Simplified Job Shop." Management Science 6(3), pp. 311-323.
- Baker, K. R. 1974. Introduction to Sequencing and Scheduling. New York, John Wiley and Sons.
- Baker, K. R. 1984. "Sequencing Rules and Due-Date Assignments in a Job Shop." Management Science 30(9), pp. 1093-1104.
- Baker, K. R. and Bertrand, J. W. M. 1981. "An Investigation of Due-Date Assignment Rules with Constrained Tightness." Journal of Operations Management 1(3), pp. 109-120.
- Baker, K. R. and Bertrand, J. W. M. 1982. "A Dynamic Priority Rule for Sequencing Against Due-Dates." Journal of Operations Management 3(1), pp. 37-42.
- Baker, K. R. and Kanet, J. J. 1983. "Job Shop Scheduling with Modified Due Dates." Journal of Operations Management 4(1), pp. 11-22.
- Balas, E., Lenstra, J. K., and Vazacopoulos, A. 1995. "The One-Machine Problem With Delayed Precedence Constraints and Its Use in Job Shop Scheduling." Management Science 41(1), pp. 94-109.
- Bean, J. C. 1994. "Genetic Algorithms and Random Keys for Sequencing and Optimization." ORSA Journal on Computing 6(2), pp. 154-160.
- Belew, R. and Booker, L. Editors, 1991. Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc.
- Bellman, R., Esogbue, A. O., and Nabeshima, I. 1982. Mathematical Aspects of Scheduling and Applications. New York, Pergamon Press.
- Berry, W. L. and Finlay, R. A. 1976. "Critical Ratio Scheduling with Queue Waiting Time Information: An Experimental Analysis." AIIE Transactions 8(2), pp. 161-168.
- Bhaskaran, K. and Pinedo, M. 1992. "Dispatching." In Salvendy, G. ed. Handbook of Industrial Engineering New York, John Wiley & Sons, Incorporated, pp. 2182-2198.
- Bierwirth, C. 1995. "A Generalized Permutation Approach to Job Shop Scheduling With Genetic Algorithms." OR Spektrum 17, pp. 87-92.
- Bierwirth, C., Kopfer, H., Mattfel, D., and Rixen, I. 1995. "Genetic Algorithm Based Scheduling in a Dynamic Manufacturing Environment." In DeSilva, C.

Editor. Proceedings of the Second IEEE Conference on Evolutionary Computation. Perth, IEEE Press, pp. 67-73.

- Bierwirth, C., Kopfer, H., Mattfel, D., and Rixen, I. 1995. "Genetic Algorithm Based Scheduling in a Dynamic Manufacturing Environment." In DeSilva, C. Editor. Proceedings of the Second IEEE Conference on Evolutionary Computation. Perth, IEEE Press, pp. 67-73.
- Brown, A. P. and Lomnicki, Z. A. 1966. "Some Application of the "Branch-and-Bound" Algorithm to Machine Scheduling Problem." Operational Research Quarterly 17(2), pp. 173-186.
- Browne, J., Harhen, J. and Shivnan, J. 1988. Production Management Systems. New York, Addison-Wesley Publishing Company.
- Buffa, E. S. and Miller, J. G. 1968. Production-Inventory Systems: Planning and Control. Third Edition, Homewood, Illinois, Richard D. IRWIN, Incorporated
- Buxey, G. 1989. "Production Scheduling: Practice and Theory." European Journal of Operational Research 39, pp. 17-31.
- Campbell, H. G., Dudek, R. A., and Smith, M. L. 1970. "A Heuristic Algorithm for the n Job, m Machine Sequencing Problem." Management Science 16(10), pp. B630-B637.
- Carroll, D. C. 1965. Heuristic Sequencing of Single and Multiple Component Jobs. Ph.D. Dissertation, Sloan School of Management, MIT.
- Chambers, L. Editor, 1995a. Practical Handbook of Genetic Algorithms Applications. Volume I, New York, CRC Press.
- Chambers, L. Editor, 1995b. Practical Handbook of Genetic Algorithms Applications. Volume II, New York, CRC Press.
- Chang, Feng-Chang R. 1994. "A Study of Factor Affecting Due-Date Predictability in a Simulation Dynamic Job Shop." Journal of Manufacturing Systems 13(6), pp. 393-400.
- Chang, Y., Sueyoshi, T., and Sullivan, R. 1996. "Ranking Dispatching Rules by Data Envelopment Analysis in a Job Shop Environment." IIE Transactions 28, pp. 631-642
- Chen, Chuen-Lung, Vempati, V. S., and Aljaber, N. 1995. "An Application of Genetic Algorithms for Flow Shop Problems." European Journal of Operational Research 80, pp. 389-396.
- Chen, H. and Flann, N. 1994. "Parallel Simulated Annealing and Genetic Algorithms: A Space of Hybrid Methods." in Davidor, Y., Schwefel, H., and Männer, R. Editors Parallel Problem Solving from Nature-PPSN III. Berlin Heidelberg, Springer-Verlag, pp. 428-438.

- Cheng, T. C. E. and Gupta, M. C. 1989. "Survey of Scheduling Research Involving Due Date Determination Decisions." European Journal of Operational Research 38, pp. 156-166.
- Cheng, T. C. E. and Sin, C. C. S. 1990. "A State-of-the-art Review of Parallel-Machine Scheduling Research." European Journal of Operational Research 47, pp. 271-292.
- Cleveland, G. and Smith, S. F. 1989. "Using Genetic Algorithms to Schedule Flow Shop Releases." In Schaffer, J. Editor Proceedings of the Third International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 160-169.
- Conway, R. W. 1965a. "Priority Dispatching and Work-In-Process Inventory in a Job Shop." Journal of Industrial Engineering 16(2), pp. 123-130.
- Conway, R. W. 1965b. "Priority Dispatching and Job Lateness In a Job Shop." Journal of Industrial Engineering 16(4), pp. 228-237.
- Conway, R. W., Johnson, B. M., and Maxwell, W. L. 1960. "An Experimental Investigation of Priority Dispatching." Journal of Industrial Engineering 11(3), pp. 221-229.
- Conway, R. W., Maxwell, W. L., and Miller, L. W. 1967. Theory of Scheduling. Massachusetts, Addison-Wesley Publishing Company.
- Croce, F. D., Tadei, R., and Volta, G. 1995. "A Genetic Algorithm for the Job Shop Problem." Computers and Operations Research 22(1), pp. 15-24.
- Dannenbring, D. G. 1977. "An Evaluation of Flow Shop Sequencing Heuristics." Management Science 23(11), pp. 1174-1182.
- Dar-El, E. M. and Wysk, R. A. 1982. "Job Shop Scheduling-- A Systematic Approach." Journal of Manufacturing Systems 1(1), pp. 77-88.
- Dauzere-Peres, S. and Lasserre, J. B. 1993. "A Modified Shifting Bottleneck Procedure for Job-Shop Scheduling." International Journal of Production Research 31(4), pp. 923-932.
- Davis, L. 1985. "Job Shop Scheduling With Genetic Algorithms." In Grefenstette, J. J. Editor, Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 136-140.
- Davis, L. ed. 1991. Handbook of Genetic Algorithms. New York, Van Nostrand Reinhold.
- Day, J. E. and Hottenstein, M. P. 1970. "Review of Sequencing Research." Naval Research Logistic Quarterly 17(11), pp. 11-39.
- Dayhoff, J. E. and Atherton, R. W. 1986. "Signature Analysis of Dispatching Schemes in Wafer Fabrication." IEEE Transactions on Components, Hybrid and Manufacturing Technology 9, pp. 498-507.

- Dell'Amico, M. and Trubian, M. 1993. "Applying Tabu Search to the Job Shop Scheduling Problem." Annals of Operations Research 41, pp. 231-252.
- Dorndorf, U. and Pesch, E. 1995. "Evolution Based Learning in a Job Shop Scheduling Environment." Computers and Operations Research 22(1), pp. 25-40.
- Dudek, R. A., Panwalkar, S. S., and Smith, M. L. 1992. "The Lessons of Flowshop Scheduling Research." Operations Research 40(1), pp. 7-13.
- Eilon, S. and Chowdhury, I. G. 1976. "Due Dates in Job Shop Scheduling." International Journal of Production Research 14(2), pp. 223-237.
- Eilon, S., Chowdhury, I. G., Serghiou, S. S. 1975. "Experiments With the SI^X rule in Job-Shop Scheduling." Simulation 24(2), pp. 45-48.
- Elmaghraby, Salah E. 1968. "The Machine Sequencing Problem-Review and Extensions." Naval Research Logistics Quarterly 15(2), pp. 205-232.
- Elvers, D. A. 1973. "Job Shop Dispatching Rules Using Various Delivery Date Setting Criteria." Production Inventory Management 14(4), pp. 62-70.
- Elvers, D. A. 1974. "The Sensitivity of the Relative Effectiveness of Job Shop Dispatching Rules with Respect to Various Arrival Distributions" AIIE Transactions 6(1), pp. 41-49.
- Elvers, D. A. and Taube, L. R. 1983. "Time Completion for Various Dispatching Rules in Job Shops." OMEGA: The International Journal of Management Science 11(1), pp. 81-89.
- Elvers, D. A. and Treleven, M. D. 1985. "Job-shop vs. Hybrid Flow-Shop Routing in a Dual Resource Constrained System." Decision Sciences 16, pp. 213-222.
- Emmons, H. 1975. "One Machine Sequencing to Minimize Mean Flow Time with Minimum Number Tardy." Naval Research Logistic Quarterly 22(3), pp. 585-592.
- Eshelman, L. J. Editor, 1995. Proceedings of the Sixth International Conference on Genetic Algorithms. San Francisco, CA, Morgan Kaufmann Publishers.
- Falkenauer, E. and Bouffouix, S. 1991. "A Genetic Algorithm for Job Shop." In Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp. 824-829.
- Fang, H., Ross, P., and Corne, D. 1993. "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems." In Forrest, S. Editor, Proceedings of the Fifth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, pp. 375-382.
- Farn, C. K. 1979. The Dynamic Aspects of Some Scheduling Problems. Master Thesis Manchester University.

- Fisher, H. and Thompson, G. L. 1963. "Probabilistic Learning Combination of Local Job-Shop Scheduling Rules." In Muth, J. F. and Thompson, G. L. Editors, Industrial Scheduling. New Jersey, Prentice-Hall, Inc, pp. 225-251.
- Foote, B. L. Fall-1993. "Lot Size Selection." Research Notes, School of Industrial Engineering, University of Oklahoma, Norman.
- Forrest, S. Editor, 1993. Proceedings of the Fifth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers.
- Forst, F. G. 1984. "A review of the Static, Stochastic Job Sequencing Literature." Opsearch 21(3), pp. 127-144.
- French, S. 1982. Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. New York, John Wiley and Sons.
- Gen, M. and Cheng, R. 1997. Genetic Algorithms and Engineering Design. New York, John Wiley & Sons, Inc.
- Gen, M., Tsumjimura, Y., and Kubota, E. 1994."Solving Job-Shop Scheduling Problems By Genetic Algorithm." In Gen, M. and Kobayashi, T. Editors. Proceedings of the 16th International Conference on Computers and Industrial Engineering. Ashikaga, Japan, pp. 1577-1582.
- Gere, W. S. 1966. "Heuristic in Job Shop Scheduling." Management Science 13(3), pp. 167-190.
- Giffler, B. and Thompson, G. L. 1960. "Algorithm for Solving Production Scheduling Problems." Operations Research 8(4), pp. 487-503.
- Goldberg, D. E. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Reading, Mass., Addison-Wesley.
- Goldberg, D. E. and Lingle R. 1985. "Alleles, Loci, and the Traveling Salesman Problem." In Grefenstette, J. J. Editor, Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 154-159.
- Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. 1979. "Optimisation and Approximation in Determining Sequencing and Scheduling: A Survey." Annals of Discrete Mathematics 5, pp. 287-326.
- Graves, S. C. 1981. "A Review of Production Scheduling." Operations Research 29(4), pp. 646-675.
- Grefenstette, J. Editor, 1987. Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates.
- Grefenstette, J. J. Editor, 1985. Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates.
- Grefenstette, J. J., Gopal, R., Rosmaita, B., and Gucht, D. V. 1985. "Genetic Algorithms for the Traveling Salesman Problem." In Grefenstette, J. J. Editor,

Proceedings of the First International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 160-168.

- Gupta, M., Gupta, Y., and Kumar, A. 1993. "Minimizing Flow Time Variance in a Single Machine System Using Genetic Algorithms." European Journal of Operational Research 70, pp. 289-303.
- Hall, N. and Sriskandarajah, C. 1995. "A Survey of Machine Scheduling Problems With Blocking and No-Wait in Process." Operations Research 43, pp. 510-525.
- Haupt, R. 1989. "A Survey of Priority Rule-Based Scheduling." OR Spektrum 11, pp. 3-16.
- Hershauer, J. C. and Ebert, R. J. 1975. "Search and Simulation Selection of a Job-Shop Sequencing Rule." Management Science 21(7), pp. 833-843.
- Holland, J. H. 1992. Adaptation in Natural and Artificial Systems. MIT Press.
- Holloway, C. A. and Nelson, R. T. 1974. "Job-Shop Scheduling With Due-Dates and Variable Processing Times." Management Science 20(9), pp. 1264-1275.
- Hottenstein, M. P. 1970. "Expedition in Job-Order Control Systems: A Simulation Study." AIIE Transactions 2(1), pp. 46-54.
- Hurrion, R. D. 1978. "An Investigation of Visual Interactive Simulation Methods Using the Job-Shop Scheduling Problem." Journal of Operational Research Society 29(11), pp. 1085-1093.
- Husband, P. and Mill, F. 1991. "Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling." In Belew, R. and Booker, L. Editors, Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc, pp. 264-270.
- Irastorza, J. C. and Deane, R. H. 1974. "A Loading and Balancing Methodology for Job Shop Control." AIIE Transactions 6(4), pp. 302-307.
- Jackson, J. R. 1957a. "Networks of Waiting Lines" Operations Research 5, pp. 518-521.
- Jackson, J. R. 1957b. "Simulation Research On Job Shop Production." Naval Research Logistic Quarterly 4(4), pp. 287-295.
- Jackson, J. R. 1963. "Jobshop-Like Queueing Systems" Management Science 10(1), pp. 131-142.
- Johnson, S. M. 1954. "Optimal Two- and Three-Stage Production Schedules With Setup Times Included." Naval Research Logistic Quarterly 1(1), pp. 61-68.
- Jones, C. H. 1973. "An Economic Evaluation of Job Shop Dispatching Rules." Management Science 20(3), pp. 293-307.
- Kamath, M. 1994. "Recent Development in Modeling and Performance Analysis Tools for Manufacturing Systems." In Joshi, S. B. and Smith, J. S.,

eds. Computer Control of Flexible Manufacturing Systems: Research and Development. New York, Chapman and Hall, pp. 231-263.

- Kanet, J. J. and Christy, D. P. 1989. “Manufacturing Systems with Forbidden Early Shipments: Implications for Setting Manufacturing Lead Times.” International Journal of Production Research 27(5), pp. 783-792.
- Kanet, J. J. and Sridharan, V. 1991. “PROGENITOR: A Genetic Algorithm for Production Scheduling.” Wirtschafts Informatik, 33, pp. 332-336.
- Kanet, J. J. and Zhou, Z. 1993. “A Decision Theory Approach to Priority Dispatching for Job Shop Scheduling.” Production and Operations Management 2(1), pp. 2-14.
- Karsiti, M. N., and Cruz, J. B., and Mulligan, J. H. 1992. “Simulation Studies of Multilevel Dynamic Job Shop Scheduling Heuristic Dispatching Rules.” Journal of Manufacturing Systems 11(5), pp. 346-358.
- Kobayashi, S., Ono, I., and Yamamura, M. 1995. “An Efficient Genetic Algorithm for Job Shop Scheduling Problems.” In Eshelman, L. J. Editor, Proceedings of the Sixth International Conference on Genetic Algorithms. San Francisco, CA, Morgan Kaufmann Publishers, pp. 506-511.
- Koulamas, C., Antony, S., and Jaen, R. 1994. “A Survey of Simulated Annealing Applications to Operations: Research Problems.” OMEGA: The International Journal of Management Science 22(1), pp. 41-56.
- Kovalev, M. Y., Shafransku, Y. M., Strusfvich, V. A., Tanafv, V. S., and Tuzikov, A. V. 1989. “Approximation Scheduling Algorithms: A Survey.” Optimization 20, pp. 859-878.
- Laarhoven, P. J. M., Aarts, E. H. L., and Lenstra, J. K. 1992. "Job Shop Scheduling By Simulated Annealing." Operations Research 40, pp. 113-125.
- Law, A. and Kelton, W. 1991. Simulation Modeling and Analysis. New York, McGraw-Hill, Inc.
- Lawrence, S. 1984. Supplement to “Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques.” GSIA, Carnegie-Mellon University.
- Lee, C. and Choi, J. 1995. “A Genetic Algorithm For Job Sequencing Problems With Distinct Due Dates and General Early-Tardy Penalty Weights.” Computers and Operations Research 22(8), pp. 857-869.
- Lee, C. and Kim, S. 1995. “Parallel Genetic Algorithms for the Earliness-Tardiness Job Scheduling Problem with General Penalty Weights.” Computer and Industrial Engineering 28(2), pp. 231-243.
- Lee, I., Sikora, R., and Shaw, M. 1993. “Joint Lot Sizing and Sequencing with Genetic Algorithms for Scheduling: Evolving the Chromosome Structure.” In Forrest, S. Editor, Proceedings of the Fifth International Conference on

Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, pp. 383-389.

- Lemoine, A. 1977. "State-of-the-art: Networks of Queues- A Survey of Equilibrium Analysis." Management Science 24(4), pp. 464-481.
- Liang, X. 1996. A Methodology for Quick Estimates of Total Tardiness and Number of Tardy Jobs for Stochastic Job Shops. Master Thesis, The University of Oklahoma.
- Liepins, G. E. and Hilliard, M. R. 1989. "Genetic Algorithms: Foundations and Applications." Annals of Operations Research 21, pp. 31-58.
- Liepins, G. E., Hilliard, M. R., Palmer, M., Morrow, M. 1987. "Greedy Genetics." In Grefenstette, J. J. Editor, Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 90-99.
- Lin, Jin-Ling 1993. An Analysis of Genetic Algorithm Behavior for Combinatorial Optimization Problems. Ph.D. Dissertation, The University of Oklahoma.
- Maccarthy, B. L. and Liu, J. 1993. "Addressing the Gap in Scheduling Research: A review of Optimization and Heuristic Methods in Production Scheduling." International Journal of Production Research 31(1), pp. 59-79.
- Mahfoud, S. W. and Goldberg, D. E. April 1992. "Parallel Recombinative Simulated Annealing: a Genetic Algorithm." Illinois Genetic Algorithms Laboratory (IlliGAL) Report 92002, Department of General Engineering, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 South Mathews Avenue, Urbana, IL 61801.
- Mattfeld, D. 1996. Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling. New York, Springer-Verlag.
- Michalewicz, Z. 1994. Genetic Algorithms + Data Structure = Evolution Programs. 2d ed. New York, Springer-Verlag.
- Miyazaki, S. 1981. "Combined Scheduling System for Reducing Job Tardiness in a Job Shop." International Journal of Production Research 19(2), pp. 201-211.
- Moore, J. M. and Wilson, R. C. 1967. "A Review of Simulation Research in Job Shop Scheduling." Production and Inventory Management 8(1), pp. 1-10.
- Morton, T. E. and Pentico, D. W. 1993. Heuristic Scheduling Systems. New York, John Wiley and Sons.
- Muchnik, M. 1992. Complete Guide to Plant Operations Management. New Jersey, Prentice Hall.
- Muhlemann, A. P., Lockett, A. O., Farn, C. K. 1982. "Job Shop Scheduling Heuristics and Frequency of Scheduling." International Journal of Production Research 20(2), pp. 227-241.

- Muth, J. F. and Thompson, G. L. 1963. Industrial Scheduling. New Jersey, Prentice-Hall, Inc.
- Nakano, R. and Yamada, T. 1991. "Conventional Genetic Algorithm for Job Shop Problems." In Belew, R. and Booker, L. Editors. Proceedings of the Fourth International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 474-479.
- Nanot, Y. R. 1963. An Experimental Investigation and Comparative Evaluation of Priority Disciplines in Job Shop-Like Queueing Networks. Ph.D. Dissertation, University of California Los Angeles.
- Nelson, R. T. and Jackson, J. R. 1957. "SWAC Computations For Some m X n Scheduling Problems." Journal of Association for Computing Machinery 4(4), pp. 438-441.
- Nelson, R. T., Holloway, C. A., and Wong, R. M. L. 1977. "Centralized Scheduling and Priority Implementation Heuristics for a Dynamic Job Shop Model." AIIE Transactions 9(1), pp. 95-102.
- Neppalli, V. R. 1994. Optimized Genetic Algorithms Approach to Solve Flow Shop Scheduling Problem. Master Thesis, Mississippi State University.
- Nof, S. Y., Rajan, V. N., and Frederick. S. W. 1990. "Knowledge-Based, Dynamic, Real-Time Scheduling and Rescheduling: A Review and Some Annotated References." Research Memorandum No. 89-16, School of Industrial Engineering, Purdue University, West Lafayette.
- Norman, B. A. and Bean, J. C. 1994. "Random Keys Genetic Algorithm for Job Shop Scheduling." Technical Report 94-5, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor.
- Noronha, S. J., and Sarma, V. V. S. 1991. "Knowledge-Based Approaches for Scheduling Problems: A Survey." IEEE Transactions on Knowledge and Data Engineering 3(2), pp. 160-171.
- Nowicki, E. and Smutnicki, C. 1996. "A Fast Taboo Search Algorithm for the Job Shop Problem." Management Science 42(6), pp. 797-813.
- Oliver, I. M., Smith. D. J., and Holland, J. R. C. 1987. "A Study of Permutation Crossover Operators on the Traveling Salesman Problem." In Grefenstette, J. J. Editor, Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 224-230.
- Osman, Ibrahim and Kelly, J. Editors, 1996. Meta-Heuristics: Theory and Applications. Boston, Kluwer Academic Publishers.
- Panwalkar, S. S. and Iskander, W. 1977. "A Survey of Scheduling Rules." Operations Research 25(1), pp. 45-61.

- Park, Y. B., Pegden, C. D., and Enscore, E. E. 1984. "A Survey and Evaluation of Static Flowshop Scheduling Heuristics." International Journal of Production Research 22(1), pp. 127-141.
- Pinedo, M. 1995. Scheduling: Theory, Algorithms, and Systems. New York, Prentice Hall.
- Putnam, A. O., Everdall, R. Dorman, G. H., Cronan, R. R., and Lindgren, L. H. 1971. "Updating Critical Ratio and Slack Time Priority Scheduling Rules." Production And Inventory Management 12(4), pp. 51-72.
- Rachamadugu, R. V., Raman, N., and Talbot, F. B. 1986. "Real-Time Scheduling of an Automated Manufacturing Center." National Bureau Standards Special Publication 724, pp. 293-315.
- Ragatz, G. L. and Mabert, V. A. 1984. "A Simulation Analysis of Due Date Assignment Rules." Journal of Operations Management 5(1), pp. 27-39.
- Raghavachari, M. 1988. "Scheduling Problems with Non-Regular Penalty Functions--A Review." Opsearch 25(3), pp. 144-164.
- Raghu, T. S. and Rajendran, C. 1993. "An Efficient Dynamic Dispatching Rule for Scheduling in a Job Shop." International Journal of Production Economics 32, pp. 301-313.
- Ramasesh, R. 1990. "Dynamic Job Shop Scheduling: A Survey of Simulation Research." OMEGA: The International Journal of Management Science 18(1), pp. 43-57.
- Rawlins, G. J. E. Editor, 1991. Foundations of Genetic Algorithms. San Mateo, California, Morgan Kaufmann Publishers.
- Reeves, C. 1995. "A Genetic Algorithm for Flowshop Sequencing." Computer and Operations Research 22(1), pp. 5-13.
- Rinnooy Kan, A. H. G. 1976. Machine Scheduling Problems: Classification, Complexity, and Computations. Martinus Nijhoff, The Hague.
- Rodammer, F. A. and White, K. P. 1988. "A Recent Survey of Production Scheduling." IEEE Transactions on Systems, Man and Cybernetics 18(6), pp. 841-851.
- Rohleder, T. R. and Scudder, G. 1993a. "Comparing Performance Measure in Dynamic Job Shops: Economics vs. Time." International Journal of Production Economics 32, pp. 169-183.
- Rohleder, T. R. and Scudder, G. 1993b. "A Comparison of Order-Release and Dispatch Rules for the Dynamic Weighted Early/Tardy Problem." Production and Operations Management 2(3), pp. 221-238.
- Rowe, A. J. 1958. Sequential Decision Rules in Production Scheduling. Ph.D. Dissertation, University of California Los Angeles.

- Rubin, P. and Ragatz, G. L. 1995. "Scheduling in a Sequence Dependent Setup Environment With Genetic Search." Computers and Operations Research 22(1), pp. 85-99.
- Russell, Dar-El, and Taylor 1987."A Comparative Analysis of the COVERT Job Sequencing Rule Using Various Shop Performance Measures." International Journal of Production Research 25(10), pp. 1523-1540.
- Sawaqed, N. M. 1987. Experimental Investigations In a Hybrid Assembly Job Shop. Ph.D. Dissertation, The University of Oklahoma.
- Schaffer, J. Editor 1989. Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc.
- Schultz, C. R., 1989. "An Expediting Heuristic for the Shortest Processing Time Dispatching Rule." International Journal of Production Research 27(1), pp. 31-41.
- Sen, T. and Gupta, S. K. 1984. "A State-of-Art Survey of Static Scheduling Research Involving Due Dates." OMEGA: The International Journal of Management Science 12(1), pp. 63-76.
- Sridhar, H. and Rajendran, C. 1994. "A Genetic Algorithm for Family and Job Scheduling in A Flowline-Based Manufacturing Cell." Computers and Industrial Engineering 27(1-4), pp. 469-472.
- Srinivas, M. and Patnaik, L. M. 1994. "Genetic Algorithms: A Survey." Computer 27(1), pp. 17-26.
- Starkweather, T., McDaniel, S. Mathias, K., Whitley, C., and Whitley, D. 1991. "A Comparison of Genetic Sequencing Operators." In Belew, R. and Booker, L. Editors, Proceedings of the Fourth International Conference on Genetic Algorithms. San Mateo, CA, Morgan Kaufmann Publishers, Inc. pp. 69-76.
- Stöpller, S. and Bierwirth, C. 1992. "The Application of a Parallel Genetic Algorithm to the n/m/P/Cmax Flowshop Problem." In Fandel, G., Gulledge, T., and Jones, A. Editors. New Directions For Operations Research in Manufacturing. Berlin Heidelberg, Springer Verlag, pp. 161-175.
- Syswerda, G. 1991. "Scheduling Optimization Using Genetic Algorithms." In Davis, L. Editor, Handbook of Genetic Algorithms. New York, Van Nostrand Reinhold, pp. 332-349.
- Szelke, E. and Kerr, R. M. 1994. "Knowledge-Based Reactive Scheduling." Production Planning and Control 5(2), pp. 124-145.
- Udo, Godwin 1993. "An Investigation of Due-Dates Assignment Using Workload Information of a Dynamic Shop." International Journal of Production Economics 29, pp. 89-101.

- Vempati, V. S., Chen, C., and Bullington, S. 1993. “An Effective Heuristic for Flow Shop Problems with Total Flow Time as Criterion.” Computer and Industrial Engineering 25(1-4), pp. 219-222.
- Vepsalainen, A. P. J. and Morton, T. E. 1988. “Improving Local Priority Rules With Global Leadtime Estimates.” Journal of Manufacturing and Operations Management 1, pp. 102-118.
- Vepsalainen, A. P. J. and Morton, T. E. 1989. “Priority Rules for Job Shops with Weighted Tardiness Cost.” Management Science 33(8), pp. 1035-1047.
- Vig, M. M. and Dooley, K. J. 1993. “Mixing Static and Dynamic Flowtime Estimates for Due-Date Assignment.” Journal of Operations Management 11, pp. 67-79.
- Wagner, H. M. 1959. "An Integer Programming Model for Machine Scheduling." Naval Research Logistic Quarterly 6, pp. 131-140.
- Wainwright, R. L. 1993. “Introduction to Genetic Algorithms: Theory and Applications.” Tutorial Session, In The Seventh Oklahoma Symposium on Artificial Intelligence. Oklahoma State University, Stillwater, November 18-19.
- Wayson, R. D. 1965. “The Effects of Alternative Machines on Two Priority Dispatching Disciplines in the General Job Shops.” Master Thesis, Cornell University
- Weeks, J. K. 1979. “A Simulation Study of Predictable Due-Dates.” Management Science 25(4), pp. 363-373.
- Whitley, D., Starkweather, T., and Fuguay D. 1989. “Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator.” In Schaffer, J. Editor Proceedings of the Third International Conference on Genetic Algorithms. Los Altos, CA, Morgan Kaufmann Publishers, pp. 133-140.
- Yamada, T. and Nakano, R. 1996. “Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search.” In Osman, Ibrahim and Kelly, J. Editors, Meta-Heuristics: Theory and Applications. Boston, Kluwer Academic Publishers, pp. 237-248.
- Yamada, T., Rosen, E., and Nakano, R. 1994. “A Simulated Annealing Approach to Job Shop Scheduling Using Critical Block Transition Operators.” In The 1994 IEEE International Conference on Neural Networks. Piscataway, IEEE, Inc., pp. 4687-4692.
- Yen, P. C. and Pinedo, M. L. 1994. “Scheduling Systems: A Survey.” Technical Report, Department of Industrial Engineering and Operations Research, Columbia University, New York.

Add the following refernce to the book as we talked last time and we need to meet this week to finalize and wrap up the issues: