NEHRU COLLEGE OF ENGINEERING AND RESEARCH CENTRE
(NAAC Accredited)
(Approved by AICTE, Affiliated to APJ Abdul Kalam Technological University, Kerala)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE MATERIALS

CS 365 OPTIMIZATION TECHNIQUES

VISION OF THE INSTITUTION

To mould true citizens who are millennium leaders and catalysts of change through excellence in education.

MISSION OF THE INSTITUTION

NCERC is committed to transform itself into a center of excellence in Learning and Research in Engineering and Frontier Technology and to impart quality education to mould technically competent citizens with moral integrity, social commitment and ethical values.

We intend to facilitate our students to assimilate the latest technological know-how and to imbibe discipline, culture and spiritually, and to mould them in to technological giants, dedicated research scientists and intellectual leaders of the country who can spread the beams of light and happiness among the poor and the underprivileged.

ABOUT DEPARTMENT

- Established in: 2002
- Course offered : B.Tech in Computer Science and Engineering

  M.Tech in Computer Science and Engineering

  M.Tech in Cyber Security
- Approved by AICTE New Delhi and Accredited by NAAC
- Affiliated to the University of A P J Abdul Kalam Technological University.

DEPARTMENT VISION

Producing Highly Competent, Innovative and Ethical Computer Science and Engineering Professionals to facilitate continuous technological advancement.

DEPARTMENT MISSION

1. To Impart Quality Education by creative Teaching Learning Process
2. To Promote cutting-edge Research and Development Process to solve real world problems with emerging technologies.
3. To Inculcate Entrepreneurship Skills among Students.
4. To cultivate Moral and Ethical Values in their Profession.

PROGRAMME EDUCATIONAL OBJECTIVES

PEO1: Graduates will be able to Work and Contribute in the domains of Computer Science and Engineering through lifelong learning.

PEO2: Graduates will be able to Analyse, design and development of novel Software Packages, Web Services, System Tools and Components as per needs and specifications.

PEO3: Graduates will be able to demonstrate their ability to adapt to a rapidly changing environment by learning and applying new technologies.

PEO4: Graduates will be able to adopt ethical attitudes, exhibit effective communication skills, Teamworkand leadership qualities.

PROGRAM OUTCOMES (POS)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSO)

PSO1: Ability to Formulate and Simulate Innovative Ideas to provide software solutions for Real-time Problems and to investigate for its future scope.

PSO2: Ability to learn and apply various methodologies for facilitating development of high quality System Software Tools and Efficient Web Design Models with a focus on performance optimization.

PSO3: Ability to inculcate the Knowledge for developing Codes and integrating hardware/software products in the domains of Big Data Analytics, Web Applications and Mobile Apps to create innovative career path and for the socially relevant issues.

COURSE OUTCOMES

CO1	Organize and make decision for optimization problem
CO2	Understand and apply various functions of optimization Functions
CO3	Analyze and apply unconstrained functions and Linear Programming
CO4	Learn the various tests for optimality and apply
CO5	Analyze the Network by linear programming and shortest route
CO6	Apply GA for optimized solution in various problems

MAPPING OF COURSE OUTCOMES WITH PROGRAM OUTCOMES

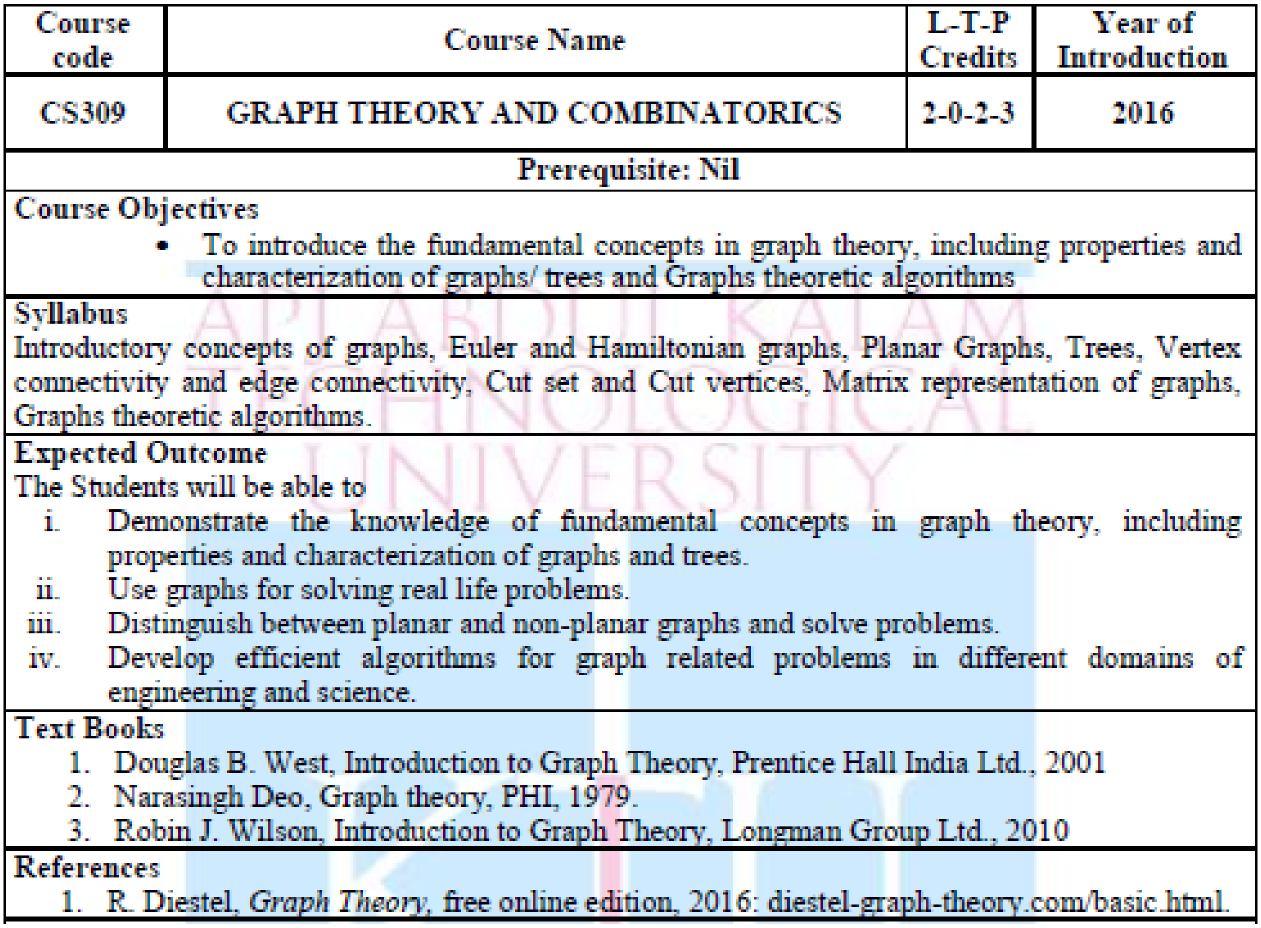
	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**
CO1	3	3	3	3	3	-	-	-	-	-	-	-
CO2	3	3	3	3	3	-	-	-	-	-	-	-
CO3	3	3	3	3	3	-	-	-	-	-	-	-
CO4	3	3	3	3	3	-	-	-	-	-	-	-
CO5	3	3	3	3	3	-	-	-	-	-	-	-
CO6	3	3	3	3	3	-	-	-	-	-	-	-

MAPPING OF COURSE OUTCOMES WITH PROGRAM SPECIFIC OUTCOMES

	PSO1	PSO2	PSO3
CO1	3	3	-
CO2	3	3	-
CO3	3	3	-
CO4	3	3	-
CO5	3	3	-
CO6	3	3	-

Note: H-Highly correlated=3, M-Medium correlated=2, L-Less correlated=1

SYLLABUS



Course code	Course Name	L-T-P Credits	Year of Introduction
CS309	GRAPH THEORY AND COMBINATORICS	2-0-2-3	2016

Prerequisite: Nil

Course Objectives

- To introduce the fundamental concepts in graph theory, including properties and characterization of graphs/ trees and Graphs theoretic algorithms

Syllabus

Introductory concepts of graphs, Euler and Hamiltonian graphs, Planar Graphs, Trees, Vertex connectivity and edge connectivity, Cut set and Cut vertices, Matrix representation of graphs, Graphs theoretic algorithms.

Expected Outcome

The Students will be able to

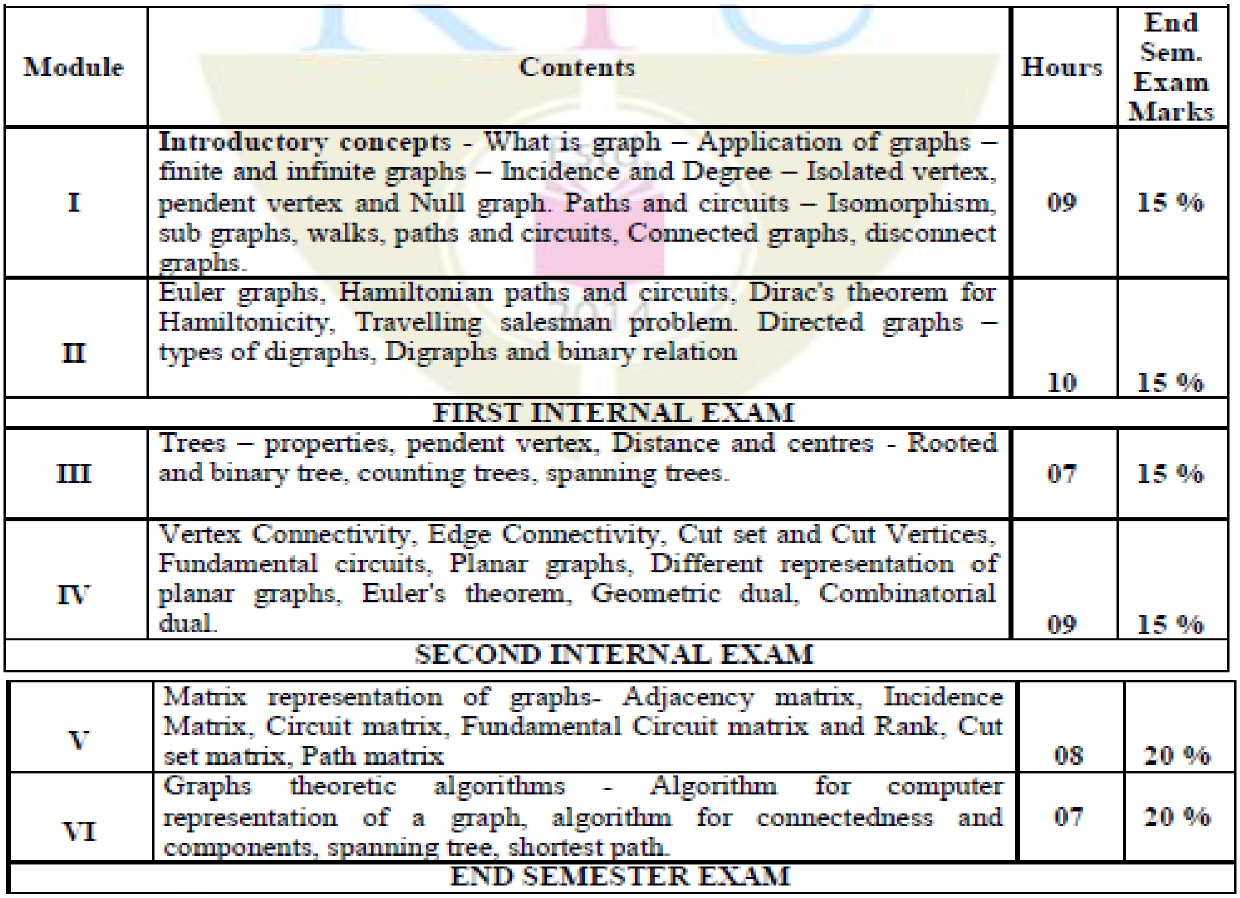
i. Demonstrate the knowledge of fundamental concepts in graph theory, including properties and characterization of graphs and trees.
ii. Use graphs for solving real life problems.
iii. Distinguish between planar and non-planar graphs and solve problems.
iv. Develop efficient algorithms for graph related problems in different domains of engineering and science.

Text Books

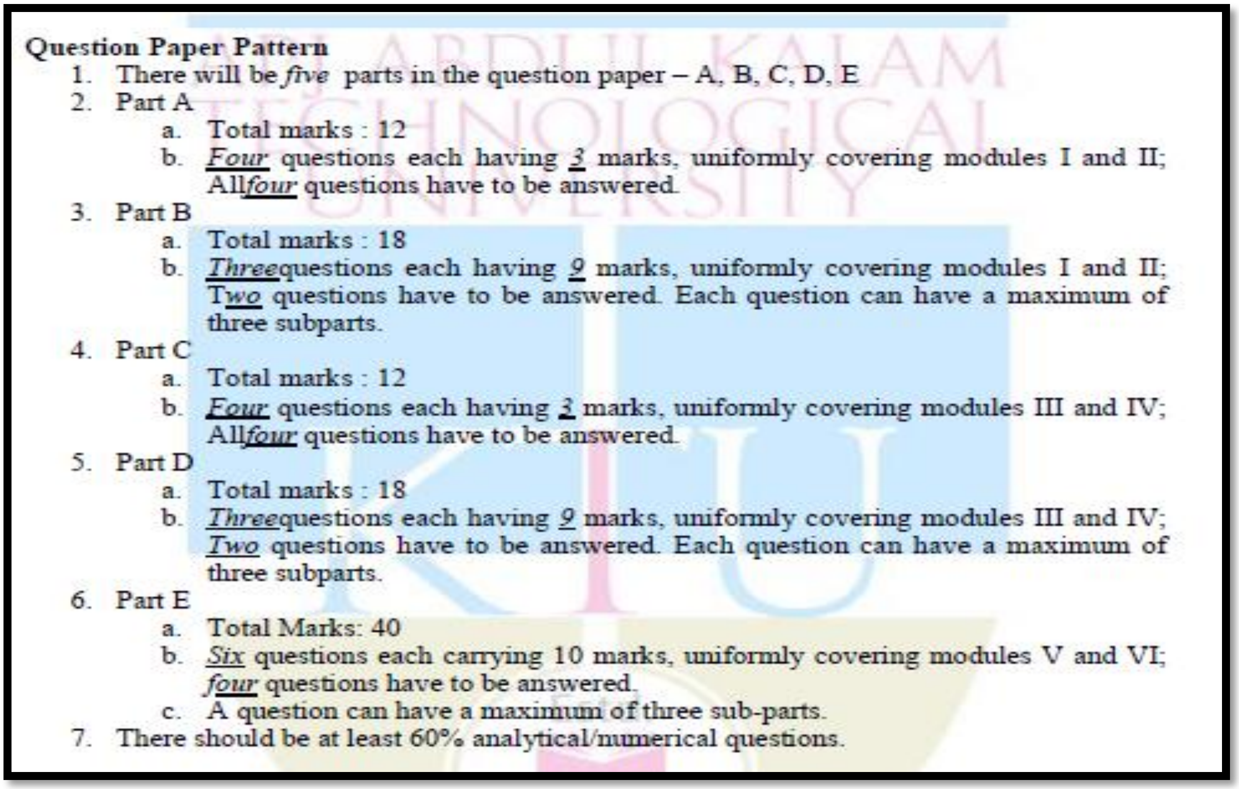
1. Douglas B. West, Introduction to Graph Theory, Prentice Hall India Ltd., 2001
2. Narasingh Deo, Graph theory, PHI, 1979.
3. Robin J. Wilson, Introduction to Graph Theory, Longman Group Ltd., 2010

References

1. R. Diestel, *Graph Theory,* free online edition, 2016: diestel-graph-theory.com/basic.html.



Module	Contents	Hours	End Sem. Exam Marks
I	**Introductory concepts** - What is graph – Application of graphs – finite and infinite graphs – Incidence and Degree – Isolated vertex, pendent vertex and Null graph. Paths and circuits – Isomorphism, sub graphs, walks, paths and circuits, Connected graphs, disconnect graphs.	09	15 %
II	Euler graphs, Hamiltonian paths and circuits, Dirac's theorem for Hamiltonicity, Travelling salesman problem. Directed graphs – types of digraphs, Digraphs and binary relation	10	15 %
	FIRST INTERNAL EXAM		
III	Trees – properties, pendent vertex, Distance and centres - Rooted and binary tree, counting trees, spanning trees.	07	15 %
IV	Vertex Connectivity, Edge Connectivity, Cut set and Cut Vertices, Fundamental circuits, Planar graphs, Different representation of planar graphs, Euler's theorem, Geometric dual, Combinatorial dual.	09	15 %
	SECOND INTERNAL EXAM		
V	Matrix representation of graphs- Adjacency matrix, Incidence Matrix, Circuit matrix, Fundamental Circuit matrix and Rank, Cut set matrix, Path matrix	08	20 %
VI	Graphs theoretic algorithms - Algorithm for computer representation of a graph, algorithm for connectedness and components, spanning tree, shortest path.	07	20 %
	END SEMESTER EXAM		



Question Paper Pattern

1. There will be *five* parts in the question paper – A, B, C, D, E
2. Part A
   a. Total marks : 12
   b. *Four* questions each having *3* marks, uniformly covering modules I and II; All*four* questions have to be answered.
3. Part B
   a. Total marks : 18
   b. *Three*questions each having *9* marks, uniformly covering modules I and II; T*wo* questions have to be answered. Each question can have a maximum of three subparts.
4. Part C
   a. Total marks : 12
   b. *Four* questions each having *3* marks, uniformly covering modules III and IV; All*four* questions have to be answered.
5. Part D
   a. Total marks : 18
   b. *Three*questions each having *9* marks, uniformly covering modules III and IV; *Two* questions have to be answered. Each question can have a maximum of three subparts.
6. Part E
   a. Total Marks: 40
   b. *Six* questions each carrying 10 marks, uniformly covering modules V and VI; *four* questions have to be answered.
   c. A question can have a maximum of three sub-parts.
7. There should be at least 60% analytical/numerical questions.

QUESTION BANK

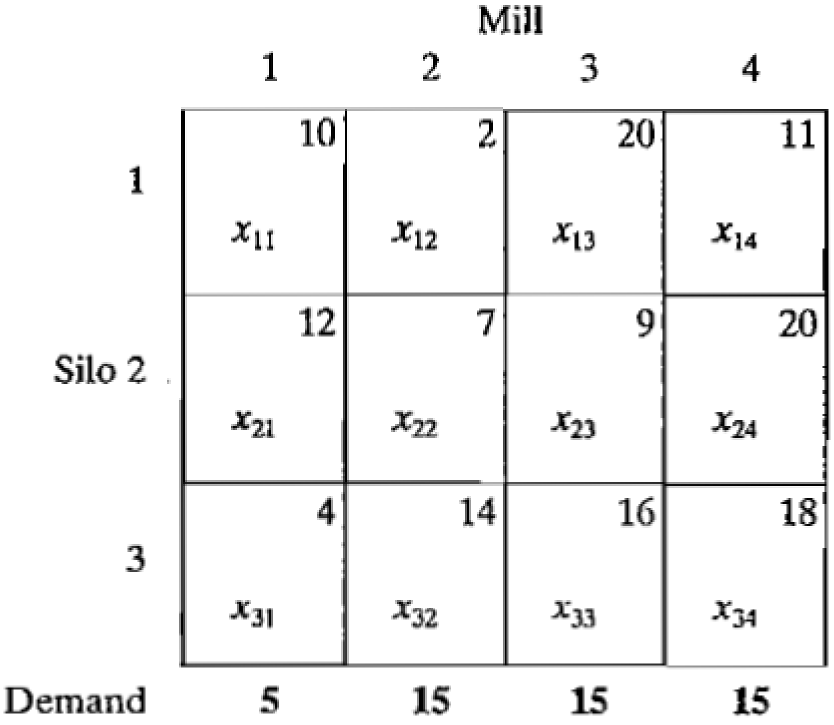
MODULE I

Q:NO:	QUESTIONS	CO	KL	PAGE NO:
1	Write the steps involved in Monte Carlo Simulation.	CO1	K3	24
2	Explain decision making under certainty and under uncertainty.	CO1	K5	14
3	Write about the elements of a queuing model.	CO1	K3	22
4	Give any three applications of optimization problem.	CO1	K2	29
5	Write classification based on existence of constraints and based on the physical structure of the problem.	CO1	K3	25
6	Write about the hierarchy of optimization.	CO1	K3	28
7	Explain different form of Queue disciplines.	CO1	K5	23
8	Write about the classification based on the nature of the equations involved and based on the permissible values of the decision variables.	CO1	K3	25
9	Write down any seven typical applications of optimization.	CO1	K3	29

MODULE II

Q:NO:	QUESTIONS	CO	KL	PAGE NO:
1	Determine whether the following functions are convex or concave. (i) $f(x) = e^x$ (ii) $f(x)= -8x^2$	CO2	K5	34
2	Explain the statement of an optimization problem.	CO2	K5	31
3	Draw the constraint surfaces in a hypothetical two-dimensional design space.	CO2	K1	32

4	Determine whether the following functions are convex or concave. 1. $f(x) = e^x$ 2. $f(x_1, x_2) = 3x_1^3 - 6x_2^2$	CO2	K5	34
5	Explain in detail about Convex and Concave functions	CO2	K5	34
6	Write about investment costs and operating costs in objective function.	CO2	K3	37
7	Write about the Hessian matrix.	CO2	K3	35
8	Explain about the various objective functions.	CO2	K5	33
9	Explain in detail about Optimizing profitability constraints.	CO2	K5	39
MODULE III				
1	Write about improved approximation in the Newton-Raphson Method.	CO3	K3	47
2	Write about the necessary and sufficient conditions for optimum of unconstrained functions.	CO3	K3	46
3	Use Newton-Raphson Method to solve the following problem. $f(x) = 4x^4 - x^2 + 5$	CO3	K5	47
4	Write down the One-Dimensional Minimization Methods.	CO3	K3	50
5	Determine the extreme points of the following: $f(x_1, x_2, x_3) = x_1 + 2x_3 + x_2x_3 - x_1^2 - x_2^2 - x_3^2$	CO3	K5	47
6	Write down the steps for the numerical method of optimization.	CO3	K3	49
7	Determine the extreme points of the following:	CO3	K5	47

<table>
<tr><td></td><td>$f(\mathbf{X}) = x_1^3 + x_2^3 - 3x_1x_2$</td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>Write down the algorithm for search with fixed step size.</td><td>CO3</td><td>K3</td><td>50</td></tr>
<tr><td colspan="5"><b>MODULE IV</b></td></tr>
<tr><td>1</td><td>Write down the transportation algorithm.</td><td>CO4</td><td>K3</td><td>77</td></tr>
<tr><td>2</td><td>Describe assignment problem.</td><td>CO4</td><td>K2</td><td>96</td></tr>
<tr><td>3</td><td>SunRay Transport Company ships truckloads of grain from three silos to four mills. The supply(in truckloads) and the demand (also in truckloads) together with the unit transportation costs per truckload on the different routes are summarized in the transportation modal in the following Table. The unit transportation costs, $c_{ij,}$ (shown in the northeast corner of each box) are in hundreds of dollars. The model seeks the minimum-cost shipping schedule x_{ij} between silo i and mill j (i=1,2,3; j=1,2,3,4).
Determine the starting solution using Northwest-corner ,
Least-cost method

<table>
<tr><td></td><td colspan="4">Mill</td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td></tr>
<tr><td>1</td><td>10
x_{11}</td><td>2
x_{12}</td><td>20
x_{13}</td><td>11
x_{14}</td></tr>
<tr><td>Silo 2</td><td>12
x_{21}</td><td>7
x_{22}</td><td>9
x_{23}</td><td>20
x_{24}</td></tr>
<tr><td>3</td><td>4
x_{31}</td><td>14
x_{32}</td><td>16
x_{33}</td><td>18
x_{34}</td></tr>
<tr><td>Demand</td><td>5</td><td>15</td><td>15</td><td>15</td></tr>
</table></td><td>CO4</td><td>K5</td><td>77</td></tr>
</table>

4	Describe about the determination of closed loop in iterative computations of the transportation algorithm.	CO4	K2	88
5	Give the transportation model and optimal solution for the following MG problem. Minimize $z=80x_{11}+215x_{12}+100x_{21}+108x_{22}+102x_{31}+68x_{32}$ Subject to $x_{11}+ x_{12}=1000$(Los Angeles) $x_{21}+ x_{22}=1500$(Detroit) $x_{31}+ x_{32}=1200$(New Oreleans) $x_{11}+ x_{21}+ x_{31}=2300$(Denver) $x_{12}+ x_{22}+ x_{32}=1400$(Miami)	CO4	K2	74
6	List down and differentiate among the starting solution determination methods.	CO4	K1	77
7	Draw the representation of the transportation model with nodes and arcs.	CO4	K1	74
8	Write about the mathematical formulation of transportation problem.	CO4	K3	76
9	Define the transportation modal.	CO4	K1	74
10	Describe the solution for assignment problem using the Hungarian method.	CO4	K2	96
MODULE V				
1	Write down the Dijkstra’s algorithm for shortest route problem.	CO5	K3	106
2	Write down the Floyd’s algorithm for all pair shortest path problem .	CO5	K3	108
3	List down the requirement for non-traditional optimization techniques.	CO5	K1	115
4	Explain in detail about np-hard and np-complete	CO5	K5	118

	problems.			
5	Write down the Tabu search algorithm.	CO5	K3	121
6	Explain in detail about short term and long term memory in Tabu search.	CO5	K5	125
MODULE VI				
1	Explain the basic concepts of genetic algorithms.	CO6	K5	128
2	Write down the genetic algorithm.	CO6	K3	128
3	Write down the simulated annealing algorithm.	CO6	K3	143
4	Explain acceptance probability and cooling of simulated annealing.	CO6	K5	144
5	Write down the application of genetic algorithm in TSP.	CO6	K3	146
6	Write down the application of simulated annealing in sequencing.	CO6	K3	147

APPENDIX 1

CONTENT BEYOND THE SYLLABUS

SL.NO:	TOPIC	PAGE NO:
1	Why do we need better optimization Algorithms?	154
2	Stochastic Gradient Descent with Momentum.	156
3	AdaGrad	157
4	RMSProp	158
5	Adam Optimizer	160
6	What is the best Optimization Algorithm for Deep Learning?	162

MODULE NOTES

Module I

The action of making the best or most effective use of a **situation** or **resource.**

Objectives

1. Understand the need and origin of the optimization methods.

2. Get a broad picture of the various applications of optimization methods used in engineering.

Introduction

1. Optimization : The act of obtaining the best result under the given circumstances.
2. Design, construction and maintenance of engineering systems involve decision making both at the managerial and the technological level.
3. Goals of such decisions :

   to minimize the effort required or

   to maximize the desired benefit

Optimization : Defined as the process of finding the conditions that give the minimum or maximum value of a function, where the function represents the effort required or the desired benefit.

Engineering applications of optimization

1. Design of structural units in construction, machinery, and in space vehicles.
2. Maximizing benefit/minimizing product costs in various manufacturing and construction processes.
3. Optimal path finding in road networks/freight handling processes.
4. Optimal production planning, controlling and scheduling.
5. Optimal Allocation of resources or services among several activities to maximize the benefit

Problem Definition

Problem definition and formulation, steps involved:

1. identification of the decision variables;
2. formulation of the model objective(s);
3. the formulation of the model constraints.

In performing these steps one must consider the following.

1. Identify the important elements that the problem consists of.
2. Determine the number of independent variables, the number of equations required to describe the system, and the number of unknown parameters.
3. Evaluate the structure and complexity of the model
4. Select the degree of accuracy required of the model

Decision-making procedure under certainty and under uncertainty

Decision problems

1. Decision problems involving a finite number of alternatives arise frequently in practice.
2. The tools used to solve these problems depend largely on the type of data available.

    1. deterministic,

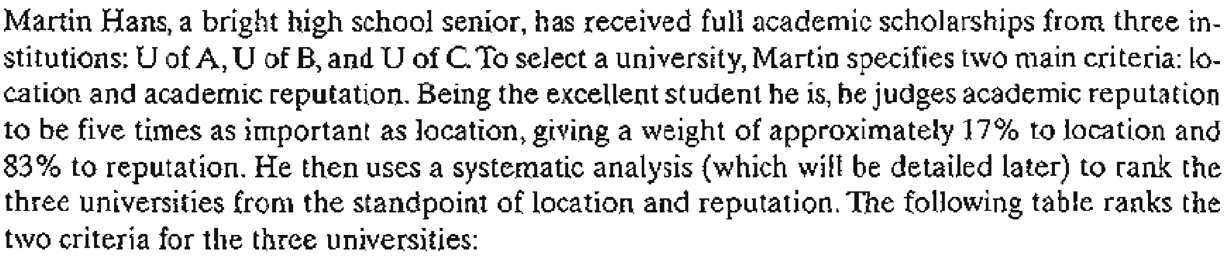
    2. probabilistic,

    3. uncertain

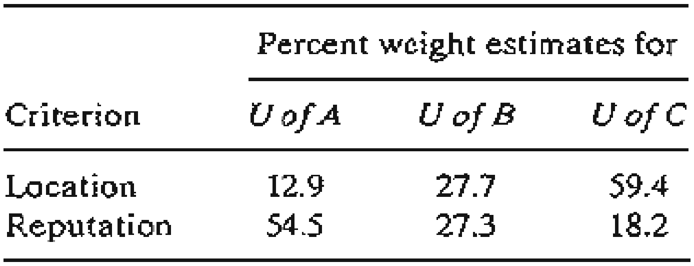
DECISION MAKING UNDER CERTAINTY

1. The **analytic hierarchy process** (AHP) is a prominent tool for dealing with decisions under certainty.
2. All the functions are well defined.
3. AHP is designed for situations in which ideas, feelings, and emotions affecting the decision process are quantified to provide a numeric scale for prioritizing the alternatives.

AHP(A**nalytic Hierarchy Process)**
Example

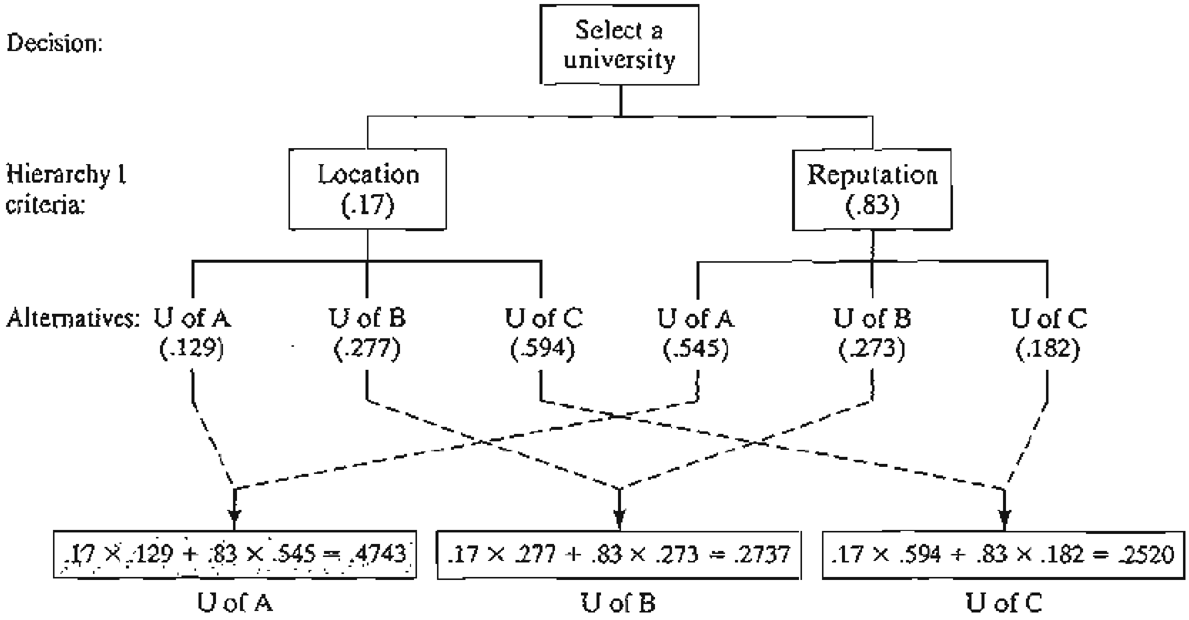


Martin Hans, a bright high school senior, has received full academic scholarships from three institutions: U of A, U of B, and U of C. To select a university, Martin specifies two main criteria: location and academic reputation. Being the excellent student he is, he judges academic reputation to be five times as important as location, giving a weight of approximately 17% to location and 83% to reputation. He then uses a systematic analysis (which will be detailed later) to rank the three universities from the standpoint of location and reputation. The following table ranks the two criteria for the three universities:



	Percent weight estimates for		
Criterion	*U of A*	*U of B*	*U of C*
Location	12.9	27.7	59.4
Reputation	54.5	27.3	18.2

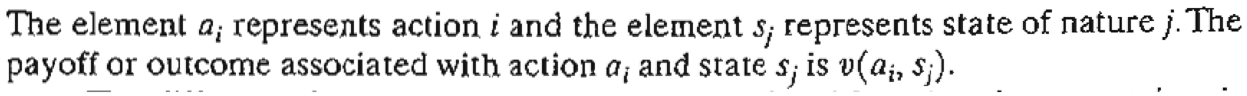
The structure of the decision problem is summarized



DECISION UNDER UNCERTAINTY

1. involves alternative actions whose payoffs depend on the (random) states of nature.
2. the payoff matrix of a decision problem with m alternative actions and n states of nature can be represented as

$$\begin{array}{c|cccc|}
 & s_1 & s_2 & \cdots & s_n \\
\hline
a_1 & v(a_1, s_1) & v(a_1, s_2) & \cdots & v(a_1, s_n) \\
a_2 & v(a_2, s_1) & v(a_2, s_2) & \cdots & v(a_2, s_n) \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
a_m & v(a_m, s_1) & v(a_m, s_2) & \cdots & v(a_m, s_n) \\
\hline
\end{array}$$



The element a_i represents action i and the element s_j represents state of nature j. The payoff or outcome associated with action a_i and state s_j is $v(a_i, s_j)$.

in the case of uncertainty, the probability distribution associated with the states *Sj*, *j* = 1, 2, ... , *n*, is either unknown or cannot be determined.

This lack of information has led to the development of the following criteria for analyzing the decision problem:

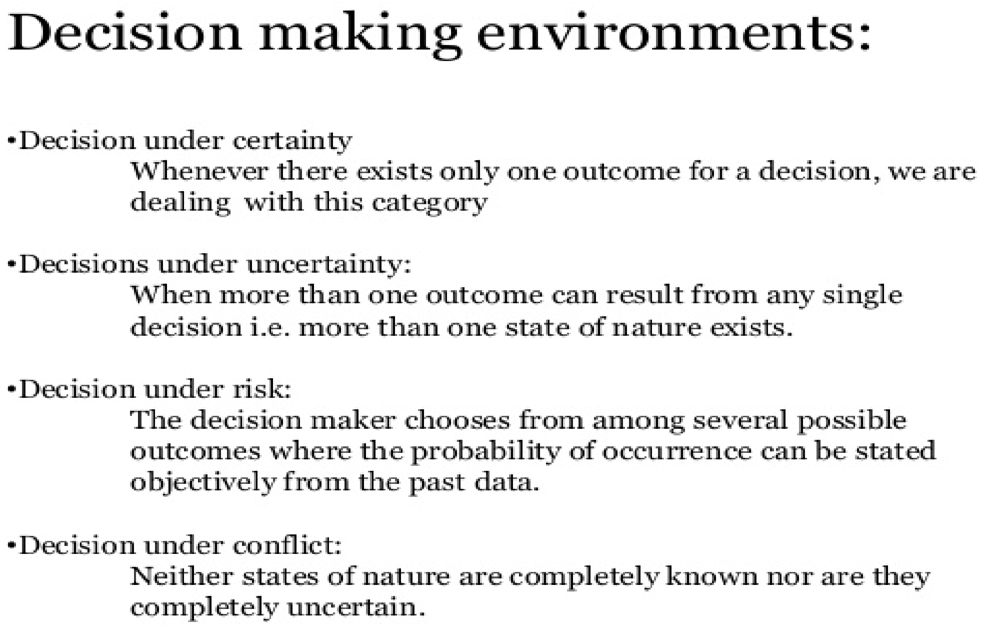
1. Laplace

2. Minimax

3. Savage

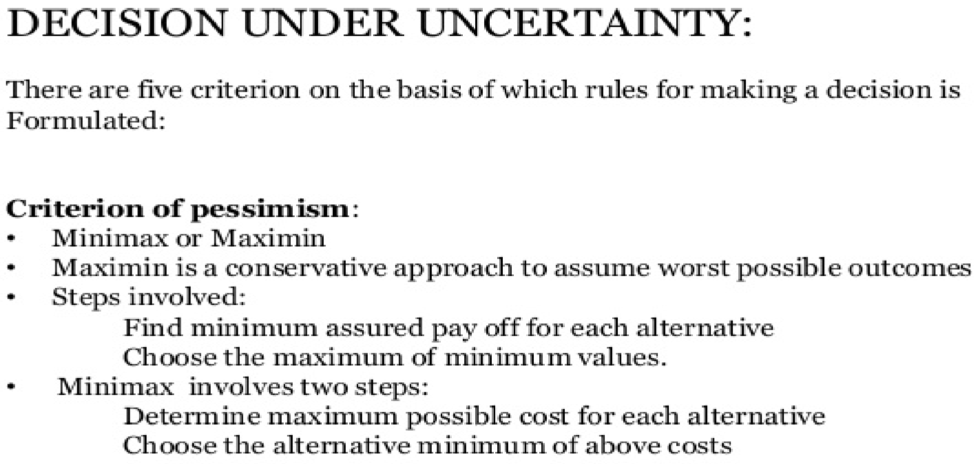
4. Hurwicz

These criteria differ in how conservative the decision maker is in the face of uncertainty.



Decision making environments:

- Decision under certainty
  Whenever there exists only one outcome for a decision, we are dealing with this category
- Decisions under uncertainty:
  When more than one outcome can result from any single decision i.e. more than one state of nature exists.
- Decision under risk:
  The decision maker chooses from among several possible outcomes where the probability of occurrence can be stated objectively from the past data.
- Decision under conflict:
  Neither states of nature are completely known nor are they completely uncertain.

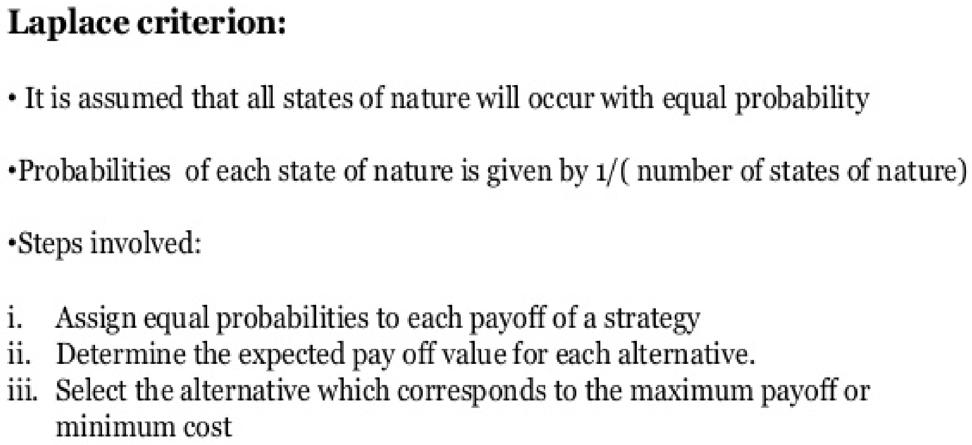


DECISION UNDER UNCERTAINTY:

There are five criterion on the basis of which rules for making a decision is Formulated:

Criterion of pessimism:

- Minimax or Maximin
- Maximin is a conservative approach to assume worst possible outcomes
- Steps involved:
  Find minimum assured pay off for each alternative
  Choose the maximum of minimum values.
- Minimax involves two steps:
  Determine maximum possible cost for each alternative
  Choose the alternative minimum of above costs



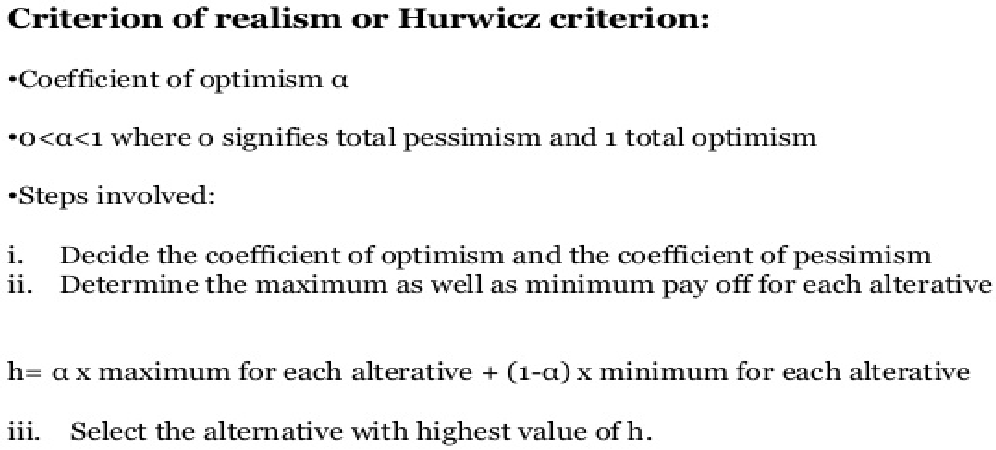
Laplace criterion:

• It is assumed that all states of nature will occur with equal probability

•Probabilities of each state of nature is given by 1/(number of states of nature)

•Steps involved:

i. Assign equal probabilities to each payoff of a strategy
ii. Determine the expected pay off value for each alternative.
iii. Select the alternative which corresponds to the maximum payoff or minimum cost



Criterion of realism or Hurwicz criterion:

•Coefficient of optimism α

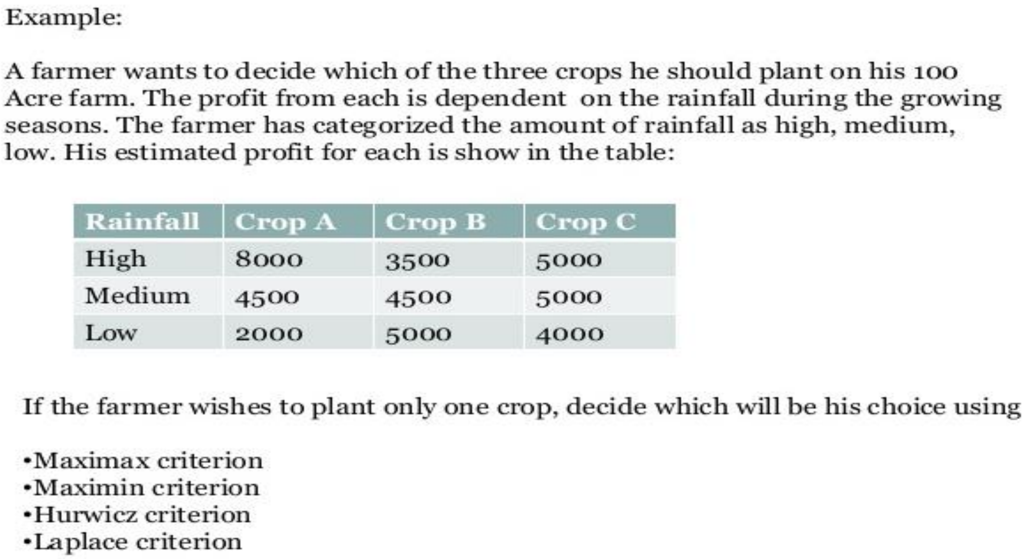
•0<α<1 where 0 signifies total pessimism and 1 total optimism

•Steps involved:

i. Decide the coefficient of optimism and the coefficient of pessimism
ii. Determine the maximum as well as minimum pay off for each alterative

h= α x maximum for each alterative + (1-α) x minimum for each alterative

iii. Select the alternative with highest value of h.



Example:

A farmer wants to decide which of the three crops he should plant on his 100 Acre farm. The profit from each is dependent on the rainfall during the growing seasons. The farmer has categorized the amount of rainfall as high, medium, low. His estimated profit for each is show in the table:

Rainfall	Crop A	Crop B	Crop C
High	8000	3500	5000
Medium	4500	4500	5000
Low	2000	5000	4000

If the farmer wishes to plant only one crop, decide which will be his choice using

- Maximax criterion
- Maximin criterion
- Hurwicz criterion
- Laplace criterion

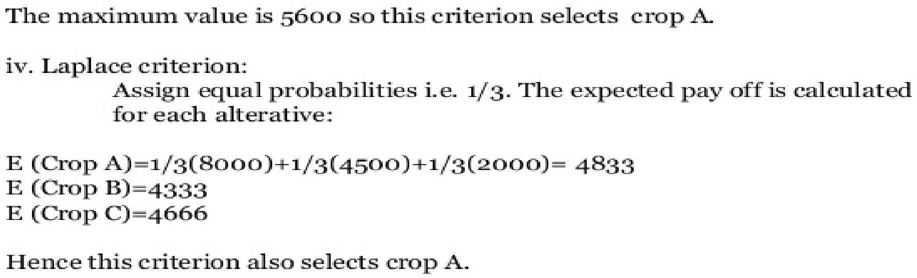
Rainfall	Crop A	Crop B	Crop C
High	8000	3500	5000
Medium	4500	4500	5000
Low	2000	5000	4000

i. Maximax criterion:
From table we observe that maximum pay off for each alternative are 8000, 5000 ad 5000 respectively. Maximum among these is 8000 corresponding to crop A. So this strategy chooses crop A .

ii. Maximin criterion selects crop C

iii. Hurwicz criterion:
Assuming degree of optimism $\alpha = 0.6$ ad therefore $1-\alpha = 0.4$, the value of h is calculated in the table:

Alternative	Maximum pay off	Minimum pay off	h
Crop A	8000	2000	5600
Crop B	5000	3500	4400
Crop C	5000	4000	4600



The maximum value is 5600 so this criterion selects crop A.

iv. Laplace criterion:

Assign equal probabilities i.e. 1/3. The expected pay off is calculated for each alterative:

E (Crop A)=1/3(8000)+1/3(4500)+1/3(2000)= 4833
E (Crop B)=4333
E (Crop C)=4666

Hence this criterion also selects crop A.

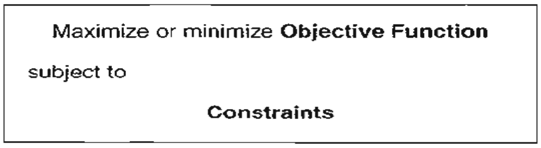
Basic terminology of operations research

1. mathematical modeling,
2. feasible solutions,
3. Optimization, and
4. Iterative computations

Defining the problem correctly is the most important (and most difficult) phase of practicing OR.

We can look at the situation as a decision-making problem whose solution requires answering three questions:

1. What are the decision alternatives?
2. Under what restrictions is the decision made?
3. What is an appropriate objective criterion for evaluating the alternatives?
4. While mathematical modeling is a cornerstone of OR, intangible (unquantifiable) factors (such as human behavior) must be accounted for in the final decision.
5. The general OR model can be organized in the following general format:



Maximize or minimize **Objective Function**

subject to

Constraints

1. Feasible solution

- if it satisfies all the constraints

2. Optimal solution

-in addition to being feasible, it yields the best (maximum or minimum) value of the objective

function

1. OR models are designed to "optimize" a specific objective criterion subject to a set of constraints.
2. The quality of the resulting solution depends on the completeness of the model in representing the real system.

SOLVING THE OR MODEL

I. In OR, we do not have a single general technique to solve all mathematical models that can arise in practice.

II. OR techniques

1. linear programming,

   linear objective and constraint functions

2. Integer programming

   the variables assume integer values,

3. dynamic programming

   the original model can be decomposed into more manageable sub problems

4. network programming

   the problem can be modeled as a network

5. nonlinear programming

   functions of the model are nonlinear

Probability and decision- making

1. How probability can be used to aid the decision–making process.

Eg:

suppose we're considering launching a new product on the market.

2. We conduct a pre–launch questionnaire and 86 out of the 100 questionnaire respondents say that they would buy our product if it was on the market.

3. probability statement:

$$P(\text{product successful}) = \frac{86}{100} = 0.86,$$

which is quite good, and so surely we should launch the product?

It looks promising! But . . .

we should also consider the financial outcome of our situation.

- For example,

1. if the product is successful,

we might make a reasonable profit, but

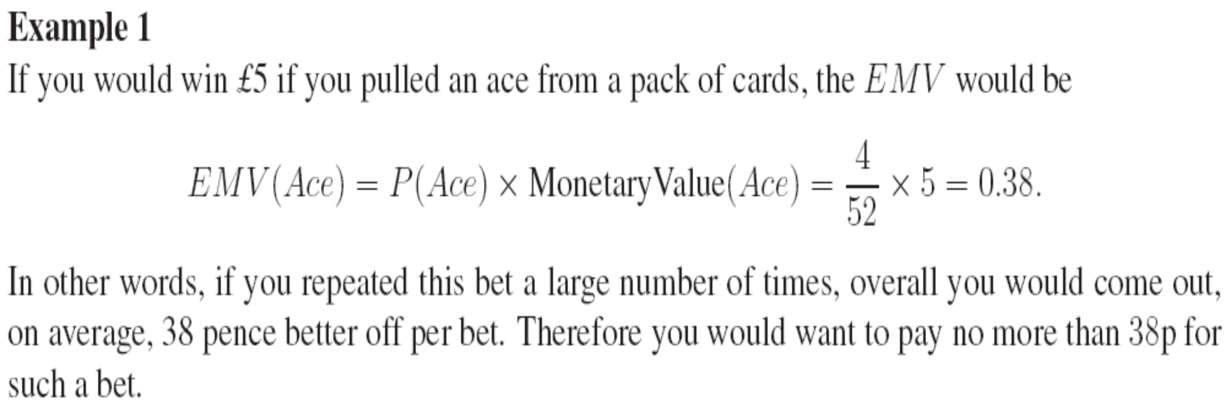
2. if the product is not successful,

we could stand to lose a lot more than we would gain under success (set–up costs, advertising, production costs etc), and

such financial considerations could outweigh the high probability of success alone.

So, in real–life scenarios, not only do we use probability to aid the decision–making process, but we also take into account the financial implications of our decisions.

This is achieved by weighting the probability of different outcomes by their value Expected Monetary Value (EMV), which is often financial.



Example 1

If you would win £5 if you pulled an ace from a pack of cards, the EMV would be

$$EMV(Ace) = P(Ace) \times \text{MonetaryValue}(Ace) = \frac{4}{52} \times 5 = 0.38.$$

In other words, if you repeated this bet a large number of times, overall you would come out, on average, 38 pence better off per bet. Therefore you would want to pay no more than 38p for such a bet.

1. In general, the expected monetary value of a project (or bet) is given by the formula

$$EMV = \sum P(\text{Event}) \times \text{ Monetary value of Event}$$

2. where the sum is over all possible events that make up the project.

Queuing or Waiting line theory-Simulation

WHY STUDY QUEUES?

Waiting for service is part of our daily life.

1. We wait to eat in restaurants,

2. we "queue up“ at the check-out counters in grocery stores, and

3. we "line up" for service in post offices.

And the waiting phenomenon is not an experience limited to human beings only:

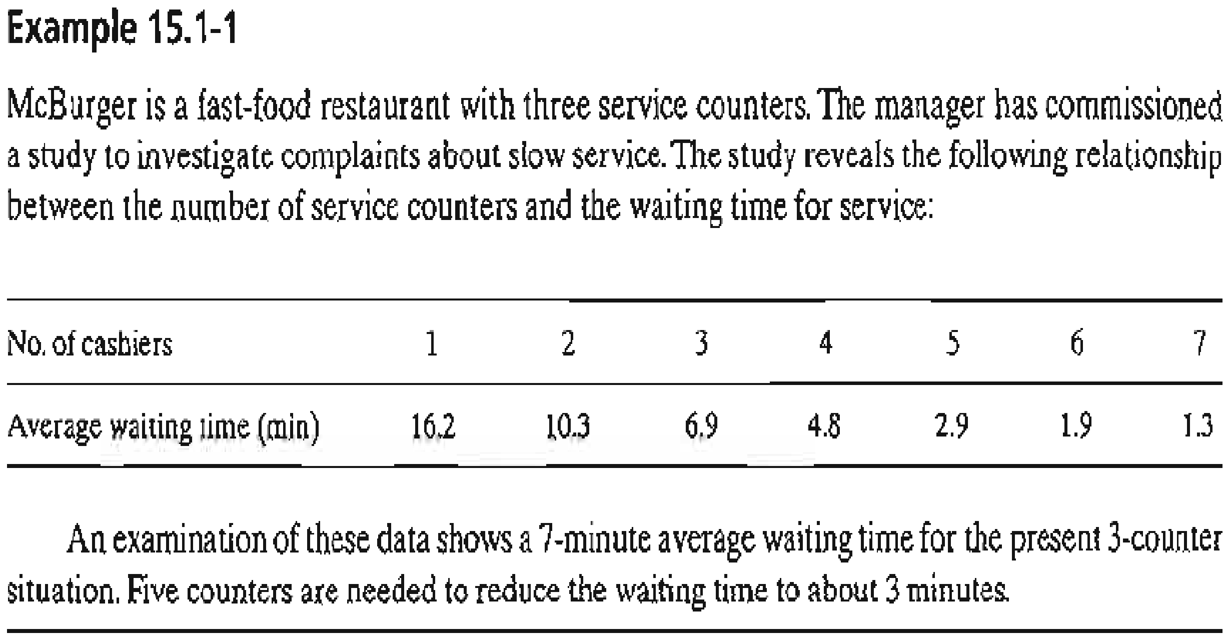
1. Jobs wait to be processed on a machine,

2. planes circle in a stack before given permission to land at an airport, and

3. cars stop at traffic lights.

Waiting cannot be eliminated completely without incurring inordinate expenses, and the goal is to reduce its adverse impact to "tolerable" levels.

The study of queues deals with quantifying the phenomenon of waiting in lines using representative measures of performance, such as average queue length, average waiting time in queue, and average facility utilization.



Example 15.1-1

McBurger is a fast-food restaurant with three service counters. The manager has commissioned a study to investigate complaints about slow service. The study reveals the following relationship between the number of service counters and the waiting time for service:

No. of cashiers	1	2	3	4	5	6	7
Average waiting time (min)	16.2	10.3	6.9	4.8	2.9	1.9	1.3

An examination of these data shows a 7-minute average waiting time for the present 3-counter situation. Five counters are needed to reduce the waiting time to about 3 minutes.

ELEMENTS OF A QUEUING MODEL

1. The principal actors in a queuing situation are the **customer** and the **server**.
2. Customers are generated from a **source**.
3. On arrival at a service **facility**, they can start service immediately or wait in a **queue** if the facility is busy.
4. When a facility completes a service, it automatically "pulls" a waiting customer, if any, from the queue.
5. If the queue is empty, the facility becomes idle until a new customer arrives.
6. From the standpoint of analyzing queues, the arrival of customers is represented by the inter-arrival time between successive customers, and the service is described by the service time per customer.
7. **Queue size** plays a role in the analysis of queues, and it may have a finite size, as in the buffer area between two successive machines, or it may be infinite, as in mail order facilities.

8. **Queue discipline**, which represents the order in which customers are selected from a queue, is an important factor in the analysis of queuing models. The most common discipline is

    1. first come, first served (FCFS).
    2. Other disciplines include last come, first served (LCFS) and
    3. service in random order (SIRO).

9. Customers may also be selected from the queue based on some order of **priority.**

ROLE OF EXPONENTIAL DISTRIBUTION

1. In most queuing situations, the arrival of customers occurs in a totally random fashion.
2. Randomness here means that the occurrence of an event (e.g., arrival of a customer or completion of a service) is not influenced by the length of time that has elapsed since the occurrence of the last event.
3. Random inter-arrival and service times are described quantitatively in queuing models by the exponential distribution, which is defined as

$$f(t) = \lambda e^{-\lambda t}, t > 0$$

Where

- t is the time between successive events
- λ is the arrival rate

Queuing models

Single server Poisson Queue model – I

(M/M/1):(∞/FIFO)

(Single server / Infinite Queue)

In this model we assume that arrival follows a Poisson distribution and services follows an exponential distribution.

In this model we assume the arrival rate is λ and service rate is μ

Generally Queuing models may be completely specified in the following symbol form:(a/b/c):(d/e) **Kendal's Notation**

where

1. a = Probability law for the arrival(or inter arrival)time,

2. b = Probability law according to which the customers are being served.

3. c = Number of service stations d = The maximum number allowed in the system(in service and waiting)

4. e = Queue Discipline

1. Deal with the study of waiting lines.
2. They are not optimization techniques;
3. they determine measures of performance of the waiting lines, such as

   1. average waiting time in queue

   2. average waiting time for service, and

   3. utilization of service facilities.

1. Simulation estimates the measures of performance by imitating the behavior of the real system.
2. Simulation is flexible and can be used to analyze practically any queuing situation.
3. The process of developing simulation models is costly in both time and resources.
4. The execution of simulation models, even on the fastest computer, is usually slow.

   Simulation is the best thing to observing a real system.

Queuing analysis

1. The objective of queuing analysis is to offer a reasonably satisfactory service to waiting customers.
2. Queuing theory determines the measures of performance of waiting lines, such as the average waiting time in queue and the productivity of the service facility, which can then be used to design the service installation.

MONTE CARLO SIMULATION

1. Monte Carlo technique is a modeling scheme that estimates stochastic or deterministic parameters based on random sampling.
2. Monte Carlo (MC) methods are a subset of computational algorithms that use the process of **repeated random sampling** to make numerical **estimations of unknown parameters**.

Steps in MONTE CARLO SIMULATION

Step 1: Identify the Transfer Equation. ...

Step 2: Define the Input Parameters. ...

Step 3: Create Random Data. ...

Step 4: **Simulate** and Analyze Process Output

Nature and organization of optimization problems

Optimization problems can be classified based on the type of constraints, nature of design variables, physical structure of the problem, nature of the equations involved, deterministic nature of the variables, permissible value of the design variables, separability of the functions and number of objective functions.

Classification based on existence of constraints.

1. Constrained optimization problems: which are subject to one or more constraints.
2. Unconstrained optimization problems: in which no constraints exist.

Classification based on the nature of the design variables

1. There are two broad categories of classification within this classification
2. First category : the objective is to find a set of design parameters that make a prescribed function of these parameters minimum or maximum subject to certain constraints.
3. Second category: the objective is to find a set of design parameters, which are all continuous functions of some other parameter, that minimizes an objective function subject to a set of constraints.

Classification based on the physical structure of the problem

1. Based on the physical structure, we can classify optimization problems are classified as optimal control and non-optimal control problems.
2. (i) An optimal control (OC) problem is a mathematical programming problem involving a number of stages, where each stage evolves from the preceding stage in a prescribed manner.

   It is defined by two types of variables: the control or design variables and state variables.

   Classification based on the physical structure of the problem

   The problems which are not optimal control problems are called non-optimal control problems.

   Classification based on the nature of the equations involved

   Based on the nature of expressions for the objective function and the constraints, optimization problems can be classified as linear, nonlinear, geometric and quadratic programming problems.

   (i)*Linear programming problem*

If the objective function and all the constraints are linear functions of the design variables, the mathematical programming problem is called a linear programming (LP) problem. often stated in the standard form :

$$\text{Find } \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{Bmatrix}$$

$$\text{Which maximizes } f(\mathbf{X}) = \sum_{i=1}^{n} c_i x_i$$

subject to

$$\sum_{i=1}^{n} a_{ij} x_i = b_j, \qquad j = 1, 2, \ldots, m$$

$$x_i \geq 0, \qquad j = 1, 2, \ldots, m$$

where c_i, a_{ij}, and b_j are constants.

(ii) ***Nonlinear programming problem***

If any of the functions among the objectives and constraint functions is nonlinear, the problem is called a nonlinear programming (NLP) problem this is the most general form of a programming problem.

(iii)***Geometric programming problem***

–A geometric programming (GMP) problem is one in which the objective function and constraints are expressed as polynomials in X.

(iv) ***Quadratic programming problem***A quadratic programming problem is the best behaved nonlinear programming problem with a quadratic objective function and linear constraints and is concave (for maximization problems).

Classification based on the permissible values of the decision variables

Under this classification problems can be classified asintegerand real-valuedprogramming problems

(i) ***Integer programming problem***

If some or all of the design variables of an optimization problem are restricted to take only integer (or discrete) values, the problem is called an integer programming problem.

(ii) ***Real-valued programming problem***

A real-valued problem is that in which it is sought to minimize or maximize a real function by systematically choosing the values of real variables from within an allowed set. When the allowed set contains only real values, it is called a real-valued programming problem.

Classification based on deterministic nature of the variables

Under this classification, optimization problems can be classified as deterministic and stochastic programming problems.

(i) ***Deterministic programming problem***

In this type of problems all the design variables are deterministic.

(ii) ***Stochastic programming problem***

In this type of an optimization problem some or all the parameters (design variables and/or pre-assigned parameters) are probabilistic (non deterministic or stochastic).

Classification based on separability of the functions

Based on the separability of the objective and constraint functions optimization problems can be classified as separableand non-separable programming problems.

(i) ***Separable programming problems***

In this type of a problem the objective function and the constraints are separable. A function is said to be separable if it can be expressed as the sum of *n* single-variable functions

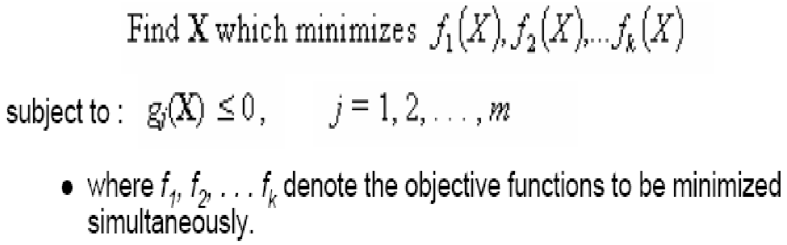
Classification based on the number of objective functions

Under this classification objective functions can be classified as single and multiobjective programming problems.

(i)***Single-objective programming problem***in which there is only a single objective.

(ii) ***Multi-objective programming problem***

A multiobjective programming problem can be stated as follows



Find X which minimizes $f_1(X), f_2(X), \ldots f_k(X)$

subject to : $g_j(X) \leq 0, \qquad j = 1, 2, \ldots, m$

- where $f_1, f_2, \ldots f_k$ denote the objective functions to be minimized simultaneously.

Scope of optimization

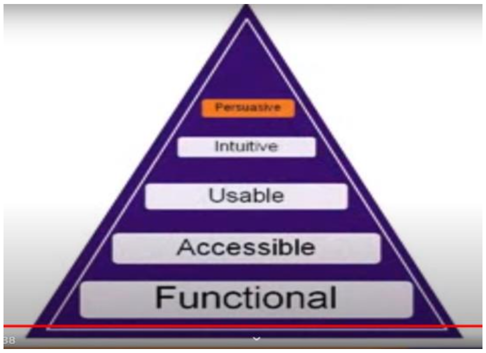
Computational aspects of optimization problems arising in such areas as Aerospace, Biomedicine, Economics, Meteorology, and Public Services (Health, Environment, Police, Fire, Transportation, etc.).

Examples are:

1. on-line and off-line computational techniques in modelling and control of dynamic systems;
2. trajectory analysis and computation;

3. optimization of decentralized systems (macro-economic systems) and systems with multicriteria;
4. optimization of resource allocation in urban systems;
5. optimization of pollution-control systems;
6. optimization of man-machine systems;
7. optimization of power systems operation.

Hierarchy of optimization



The Hierarchy of Optimization is the methodology behind conversion rate optimization. It provides a structured, systematic approach to improving the performance of your website and your conversions.

The Hierarchy of Optimization framework is based on Maslow's Hierarchy of Needs pyramid model.

In Maslow's Hierarchy of Needs, the most fundamental needs for human motivation are at the bottom. Only once these basic needs are satisfied can someone fulfill higher order psychological needs.

(Food, water, shelter, and safety trump belonging, esteem, and self-actualization. But combine all of the levels together and you have a happy, motivated individual.)

The five levels are:

1. Purpose

2. Accessible

3. Usable

4. Intuitive

5. Persuasive

1. Purpose

Purpose is at the base of your optimization hierarchy pyramid. Without a purpose, why even have a website?

2. Accessible

Having a purpose is great, but if your website can’t be accessed then you are up a creek without a paddle.

Does your website work?

Do all of the features on your website work?

Are their functions executable?

3. Usable

Is your website usable?

Is your website designed to work on multiple devices?

4. Intuitive

Intuitive and usable strongly go hand-in-hand with each other.

Can visitors find what they are looking for when they come to your website?

Yes? Great! … Is it where they expected to find it?

5. Persuasive

The top of the mountain is only the top because it has a whole mountain under it. At the same time, you can only get to the top of the mountain by climbing the rest of it.

- **Persuasive is where you bring your branding, marketing, and content strategies into play.**

Typical applications of optimization.

1. Design of aircraft and aerospace structures for minimum weight

2. Finding the optimal trajectories of space vehicles

3. Design of civil engineering structures such as frames, foundations, bridges, towers, chimneys, and dams for minimum cost

4. Minimum-weight design of structures for earthquake, wind, and other types of random loading

5. Design of water resources systems for maximum benefit

6. Optimal plastic design of structures

7. Optimum design of linkages, cams, gears, machine tools, and other mechanical components

8. Selection of machining conditions in metal-cutting processes for minimum production cost

9. Design of material handling equipment such as conveyors, trucks, and cranes for minimum cost

10. Design of pumps, turbines, and heat transfer equipment for maximum efficiency.

11. Optimum design of electrical machinery such as motors, generators, and transformers

12. Optimum design of electrical networks

13. Shortest route taken by a salesperson visiting various cities during one tour

14. Optimal production planning, controlling, and scheduling

15. Analysis of statistical data and building empirical models from experimental results to obtain the most accurate representation of the physical phenomenon

16. Optimum design of chemical processing equipment and plants

17. Design of optimum pipeline networks for process industries

18. Selection of a site for an industry

19. Planning of maintenance and replacement of equipment to reduce operating costs

20. Inventory control

21. Allocation of resources or services among several activities to maximize the benefit

22. Controlling the waiting and idle times and queueing in production lines to reduce the costs

23. Planning the best strategy to obtain maximum profit in the presence of a competitor

24. Optimum design of control systems

Module II

Essential features of optimization problems

STATEMENT OF AN OPTIMIZATION PROBLEM

An optimization or a mathematical programming problem can be stated as follows.

$$\text{Find } \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} \text{ which minimizes } f(\mathbf{X})$$

subject to the constraints

$$\begin{aligned} g_j(\mathbf{X}) &\le 0, \qquad j = 1,2,\ldots,m \\ l_j(\mathbf{X}) &= 0, \qquad j = 1,2,\ldots,p \end{aligned}$$

where

1. X is an n-dimensional vector called the *design vector,*

2. f(X) is termed the *objective Junction,* and

3. g_j (X) is known as *inequality* constraints and

4. I_j (X) are known as *equality* constraints,

The number of variables *n* and the number of constraints *m* and/or p need not be related in any way.

1. The problem stated is called a ***constrained optimization problem***.
2. Some optimization problems do not involve any constraints and can be stated as:

$$\text{Find } \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix} \text{ which minimizes } f(\mathbf{X})$$

Such problems are called **unconstrained optimization problems.**

Design Vector

Any engineering system is defined by a set of quantities:

1. Some are fixed and these are called **preassigned parameters.**
2. Some are viewed as variables during the design process and are called **design or decision variables** $x_i = 1,2,. . .,n$..

3. The design variables are collectively represented as a design vector

$$\mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}.$$

Design Constraints

1. Design variables have to satisfy certain specified functional and other requirements.

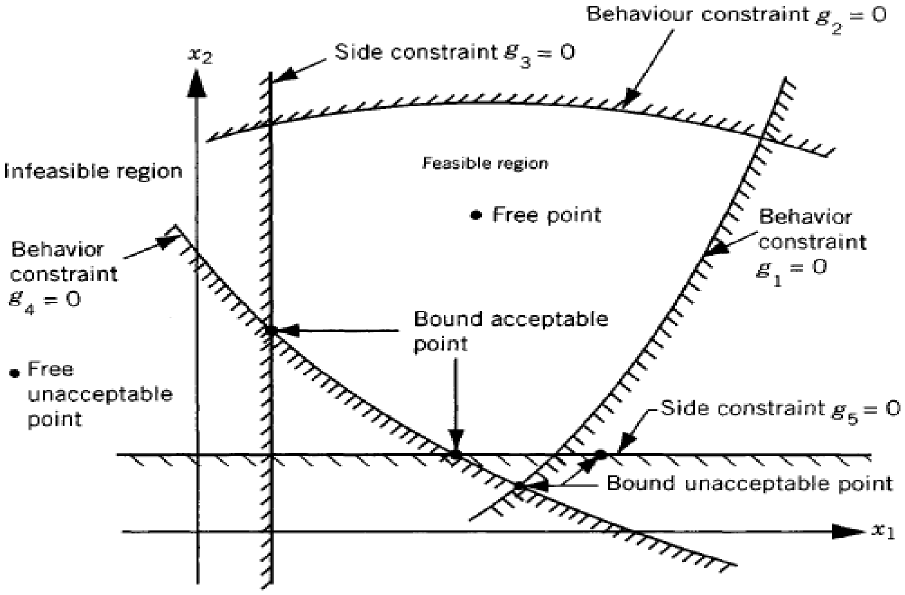
2. The restrictions that must be satisfied to produce an acceptable design are collectively called ***design constraints***

1. Constraints that represent limitations on the behavior or performance of the system are termed ***behavior* or *functional constraints***.
2. Constraints that represent physical limitations on design variables such as availability, fabricability, and transportability are known as **geometric or side constraints**.

Constraint Surface

1. The set of values of X that satisfy the equation g_j (X) = 0 forms a hypersurface in the design space and is called a ***constraint surface***.

2. The collection of all the constraint surfaces g_j (X) = 0, j = 1,2,. . .,m, which separates the acceptable region is called the ***composite constraint surface.***



Behaviour constraint $g_2 = 0$
x_2
Side constraint $g_3 = 0$
Infeasible region
Feasible region
Free point
Behavior constraint $g_4 = 0$
Behavior constraint $g_1 = 0$
Bound acceptable point
Free unacceptable point
Side constraint $g_5 = 0$
Bound unacceptable point
x_1

Figure 1.3 Constraint surfaces in a hypothetical two-dimensional design space.

1. A design point that lies on one or more than one constraint surface is called a ***bound point.***

2. the associated constraint in a *bound point* is called an ***active constraint.***

3. Design points that do not lie on any constraint surface are known *as* ***free points***.

Point can be identified as one of the following four types:

1. Free and acceptable point

2. Free and unacceptable point

3. Bound and acceptable point

4. Bound and unacceptable point

Objective Function

1. The criterion with respect to which the design is optimized, when expressed as a function of the design variables, is known as the ***criterion*** *or* ***merit*** or ***objective function***.

2. The choice of objective function is governed by the nature of problem.

An optimization problem involving multiple objective functions is known as a **multiobjective programming problem**.

Continuous functions

1. A **continuous function** is a **function** in that small changes in the input result in arbitrarily small changes in its output. If not **continuous**, a **function** is said to be discontinuous.

2. The term continuous refers to a function whose graph has no holes or breaks the optimization of continuous functions subjected to equality constraints:

   Minimize f = f(X)

   subject to

   $g_j(X) = 0, J = 1,2,. . .,m$

   where

   $$\mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}$$

   Here *m* is less than or equal to *n;*

   Discrete functions

If a function is discrete, it does *not* include all of the values between two given numbers, but rather only specific values in a particular range.

1. A bank account

2. The balance in a bank account is counted in dollars and cents, any change is countable and quantifiable. This is an example of a discrete function.

Unimodal functions

1. A *unimodal function* is one that has only one peak (maximum) or valley (minimum) in a given interval.

2. A function f(x) is unimodal if

(i) *X1* < *X2* < x* implies that f(x2) < f(x1),

and

(ii) *X2* > *X1* > x* implies that f(x1) < f(x2), where x* is the minimum point.

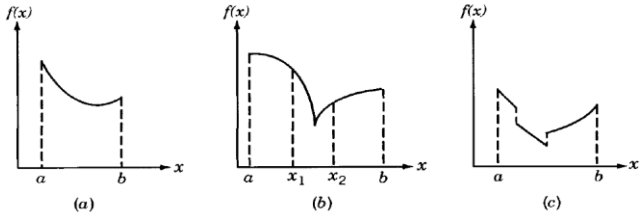


Figure 5.4 Unimodal function.

Convex and concave functions

Convex Function A function f(X) is said to be convex if for any pair of points

$$\mathbf{X}_1 = \begin{Bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_n^{(1)} \end{Bmatrix} \quad \text{and} \quad \mathbf{X}_2 = \begin{Bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ \vdots \\ x_n^{(2)} \end{Bmatrix}$$

and all λ, $0 \leq \lambda \leq 1$,

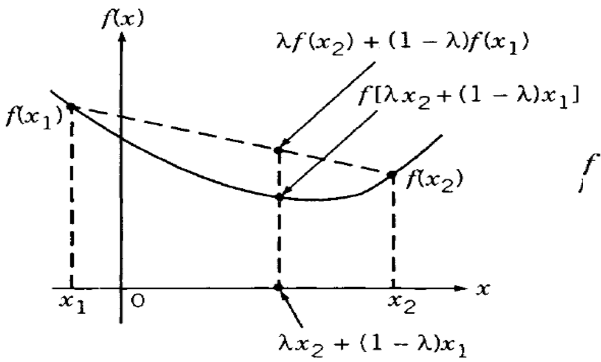
$$f[\lambda \mathbf{X}_2 + (1 - \lambda)\, \mathbf{X}_1] \leq \lambda f(\mathbf{X}_2) + (1 - \lambda)\, f(\mathbf{X}_1)$$

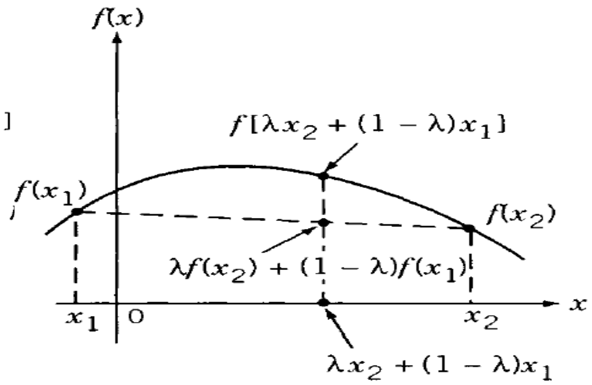
Concave Function A function Z(X) is called a concave function if for any two points X1 and X2, and for all O ≤ X ≤ 1

$$f[\lambda \mathbf{X}_2 + (1 - \lambda)\mathbf{X}_1] \geq \lambda f(\mathbf{X}_2) + (1 - \lambda) f(\mathbf{X}_1)$$

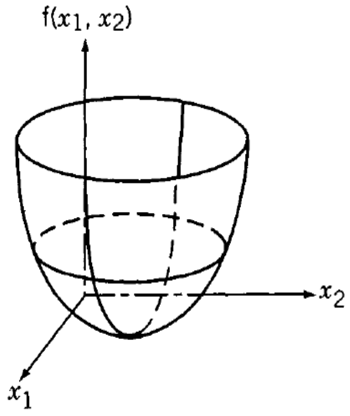
Negative of a convex function is a concave function, and vice versa.

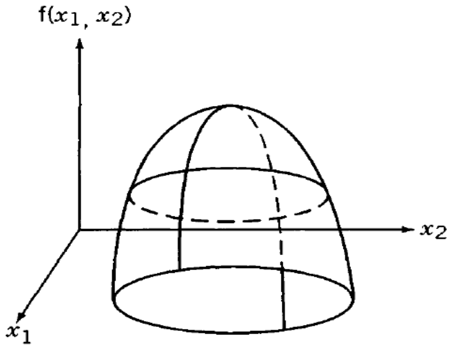
convex function in one variable





convex function in two variables





Hessian matrix

In mathematics, the Hessian matrix or Hessian is a square matrix of second-order partial derivatives of a scalar-valued function, or scalar field

Suppose $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a function taking as input a vector $x \in \mathbb{R}^n$ and outputting a scalar $f(x) \in \mathbb{R}$. If all second partial derivatives of f exist and are continuous over the domain of the function, then the Hessian matrix H of f is a square $n \times n$ matrix, usually defined and arranged as follows:

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

or, by stating an equation for the coefficients using indices i and j,

$$\mathbf{H}_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j}.$$

1. The Hessian matrix is a symmetric matrix
2. The determinant of the Hessian matrix is called the *Hessian determinant*.

Convex and concave functions

Theorem A.1 A function $f(\mathbf{X})$ is convex if for any two points $\mathbf{X}_1$ and $\mathbf{X}_2$, we have

$$f(\mathbf{X}_2) \geq f(\mathbf{X}_1) + \nabla f^T(\mathbf{X}_1)(\mathbf{X}_2 - \mathbf{X}_1)$$

Theorem A.2 A function $f(\mathbf{X})$ is convex if the Hessian matrix $\mathbf{H}(\mathbf{X}) = [\partial^2 f(\mathbf{X})/\partial x_i\, \partial x_j]$ is positive semidefinite.

Theorem A.3 Any local minimum of a convex function $f(\mathbf{X})$ is a global minimum.

Example A.1 **Determine whether the following functions are convex or concave.**

(a) $f(x) = e^x$

SOLUTION

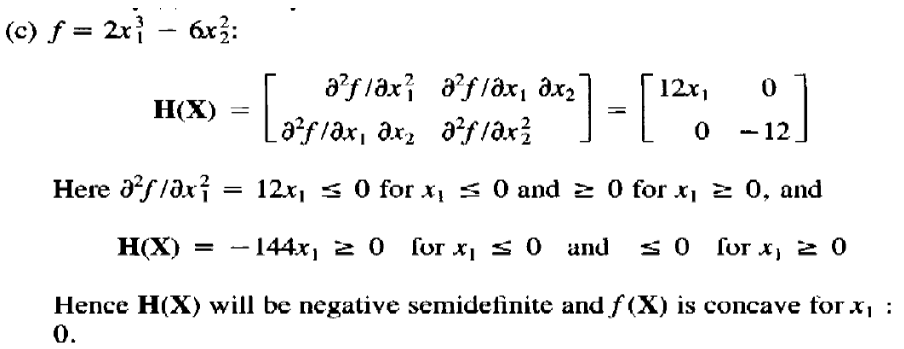
(a) $f(x = e^x$: $H(x) = d^2f/dx^2 = e^x > 0$ for all real values of x. Hence $f(x)$ is strictly convex.

(b) $f(x) = -8x^2$

SOLUTION

(b) $f(x) = -8x^2$: $H(x) = d^2f/dx^2 = -16 < 0$ for all real values of x. Hence $f(x)$ is strictly concave.

(c) $f(x_1, x_2) = 2x_1^3 - 6x_2^2$:



(c) $f = 2x_1^3 - 6x_2^2$:

$$\mathbf{H(X)} = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1\, \partial x_2 \\ \partial^2 f/\partial x_1\, \partial x_2 & \partial^2 f/\partial x_2^2 \end{bmatrix} = \begin{bmatrix} 12x_1 & 0 \\ 0 & -12 \end{bmatrix}$$

Here $\partial^2 f/\partial x_1^2 = 12x_1 \leq 0$ for $x_1 \leq 0$ and ≥ 0 for $x_1 \geq 0$, and

$$\mathbf{H(X)} = -144x_1 \geq 0 \quad \text{for } x_1 \leq 0 \quad \text{and} \quad \leq 0 \quad \text{for } x_1 \geq 0$$

Hence $\mathbf{H(X)}$ will be negative semidefinite and $f(\mathbf{X})$ is concave for x_1 :
0.

Investment costs and operating costs in objective function

Investment Costs

Investment Costs means the **costs** including but not limited to, construction **costs**, operating **costs**, development **costs** and other **costs** related to the Project up to the date of termination of the MOU.

Investment Costs means the hard and soft costs of acquisition, design, development and construction of the Company Facility, including without limitation

(i) the costs of acquiring the Property;

(ii) the costs to prepare the Property for any of the improvements constructed on or within the Property and constructing any required site work infrastructure such as streets and roads, water and electric utilities, gas utilities, drainage and related improvements and/or telecommunications and internet improvements;

(iii) the costs of obtaining all necessary governmental permits and approvals;

(iv) the costs of design, engineering, materials, labor, construction, and other services arising in connection with the design and construction of the Company Facility;

(v) all payments arising under any contracts entered into for the design or construction of the Company Facility;

(vi) legal costs and consultant fees;

(vii) reimbursements to any developer/contractor for the actual costs described above that are advanced to or on behalf of Company;

(viii) costs of furniture, fixtures, equipment and inventory; and

(ix) miscellaneous expenses.

Operating cost

Operating costs or operational costs, are the expenses which are related to the operation of a business, or to the operation of a device, component, piece of equipment or facility.

1. Operation cost, often referred to as operating cost, is the money that it takes to run your business.

2. These are the day-to-day business expenses required to keep the lights on and to have the staff necessary to sell and fulfill customer needs.

3. Operating costs also include the costs of buying or making your products and services. These are often called the cost of goods sold (COGS).

4. These are the costs that are subtracted from total revenues to generate the gross revenue numbers. Operating expenses are then subtracted from this, with taxes and interest on loans to determine the net profit of the company.

Different Types of Operating Costs

1. **Fixed costs**.

1. Fixed costs are those that do not change as sales rise or fall.

2 .**They do not reflect the productivity of a company, and a company must continue to pay them irrespective of its performance.**

3. Such costs include overhead, administrative expenses, insurance, security, and equipment, among other things.

2. Variable costs.

1. Variable costs are those that change as a company's sales and production rise or fall.

2. They include such line items as the cost of raw materials, production payroll, and electricity.

3. **As production increases, a company must buy more raw materials, hire more production personnel or use more electricity, incurring higher costs.**

3. Semi-variable costs.

1. **There is a third category of costs: Semi-variable costs, also known as a semi-fixed or mixed costs.**

2. Such costs may resemble fixed costs at or below a particular level of sales or production, but change as sales or production rise above that level.

3. An example is overtime wages: Below a certain level of production, overtime is non-existent and fixed; above that level, it becomes variable and rises or falls as production does.

How to Calculate Operating Costs

Total operating costs = Cost of goods sold (COGS) + operating expenses (OPEX)

Cost of goods sold(cost of sales)

1. Cost of goods sold, also called the cost of sales, are the expenses directly tied to the production of goods or services.

2. Subtracting COGS from revenues yields gross profit or loss.

The cost of goods sold includes the following:

1. Direct costs of material
2. Direct costs of labor
3. Direct costs of material
4. Rent of the plant or production facility
5. Benefits and wages for the production workers
6. Repair costs of equipment
7. Utility costs and taxes of the production facilities

Operating expenses

1. Operating expenses are the expenses a business incurs through its normal business operations that are not otherwise accounted for in the cost of goods sold.

2. operating expenses cannot be linked directly to the production of the products or services a company sells.

Operating expenses include:

1. Rent
2. Equipment
3. Inventory costs
4. Advertising and marketing
5. Payroll
6. Insurance premiums
7. Research and development

Optimizing profitably constraints

Every process has a constraint (bottleneck) and focusing improvement efforts on that constraint is the fastest and most effective path to improved profitability.

The Theory of Constraints is a methodology for identifying the most important limiting factor (i.e. constraint) that stands in the way of achieving a goal and then systematically improving that constraint until it is no longer the limiting factor.

Every complex system, including manufacturing processes, consists of multiple linked activities, one of which acts as a constraint upon the entire system (i.e. the constraint activity is the "weakest link in the chain").

The ultimate goal of most manufacturing companies is to make a profit – both in the short term and in the long term.

Tools for helping to achieve that goal, including:

1. **The Five Focusing Steps** (a methodology for identifying and eliminating constraints)
2. **The Thinking Processes** (tools for analyzing and resolving problems)
3. **Throughput Accounting** (a method for measuring performance and guiding management decisions)

A successful Theory of Constraints

implementation will have the following benefits:

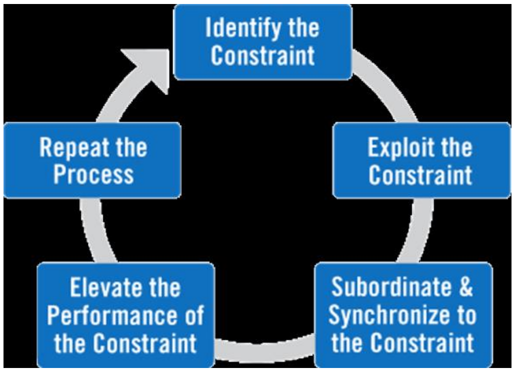
1. Increased profit (the primary goal of TOC for most companies)
2. Fast improvement (a result of focusing all attention on one critical area – the system constraint)
3. Improved capacity (optimizing the constraint enables more product to be manufactured)
4. Reduced lead times (optimizing the constraint results in smoother and faster product flow)
5. Reduced inventory (eliminating bottlenecks means there will be less work-in-process)

The core concept of the Theory of Constraints is that every process has a single constraint and that total process throughput can only be improved when the constraint is improved.

Spending time optimizing non-constraints will not provide significant benefits; only improvements to the constraint will further the goal (achieving more profit).

The Theory of Constraints uses a process known as the Five Focusing Steps to identify and eliminate constraints (i.e. bottlenecks).

The Five Focusing Steps


Identify the Constraint
Exploit the Constraint
Subordinate & Synchronize to the Constraint
Elevate the Performance of the Constraint
Repeat the Process

Step	Objective
Identify	Identify the current constraint (the single part of the process that limits the rate at which the goal is achieved).
Exploit	Make quick improvements to the throughput of the constraint using existing resources (i.e. make the most of what you have).
Subordinate	Review all other activities in the process to ensure that they are aligned with and truly support the needs of the constraint.
Elevate	If the constraint still exists (i.e. it has not moved), consider what further actions can be taken to eliminate it from being the constraint. Normally, actions are continued at this step until the constraint has been “broken” (until it has moved somewhere else). In some cases, capital investment may be required.
Repeat	The Five Focusing Steps are a continuous improvement cycle. Therefore, once a constraint is resolved the next constraint should immediately be addressed. This step is a reminder to never become complacent – aggressively improve the current constraint…and then immediately move on to the next constraint.

The Thinking Processes

1. The Thinking Processes are optimized for complex systems with many interdependencies (e.g. manufacturing lines).

2. They are designed as scientific “cause and effect” tools, which strive to first identify the root causes of undesirable effects (referred to as UDEs), and then remove the UDEs without creating new ones.

1. The Thinking Processes are used to answer the following three questions, which are essential to TOC:
2. What needs to be changed?
3. What should it be changed to?
4. What actions will cause the change?

Tools that have been formalized as part of the Thinking Processes include:

Tool	Role	Description
Current Reality Tree	Documents the current state.	Diagram that shows the current state, which is unsatisfactory and needs improvement. When creating the diagram, UDEs (symptoms of the problem) are identified and traced back to their root cause (the underlying problem).
Evaporating Cloud Tree	Evaluates potential improvements.	Diagram that helps to identify specific changes (called injections) that eliminate UDEs. It is particularly useful for resolving conflicts between different approaches to solving a problem. It is used as part of the process for progressing from the Current Reality Tree to the Future Reality Tree.
Future Reality Tree	Documents the future state.	Diagram that shows the future state, which reflects the results of injecting changes into the system that are designed to eliminate UDEs.
Strategy and Tactics Tree	Provides an action plan for improvement.	Diagram that shows an implementation plan for achieving the future state. Creates a logical structure that organizes knowledge and derives tactics from strategy. Note: this tool is intended to replace the formerly used Prerequisite Tree in the Thinking Processes.

Throughput Accounting

Throughput Accounting is an alternative accounting methodology that attempts to eliminate harmful distortions introduced from traditional accounting practices

Core Measures	Definition
Throughput	The rate at which customer sales are generated less truly variable costs (typically raw materials, sales commissions, and freight). Labor is not considered a truly variable cost unless pay is 100% tied to pieces produced.
Investment	Money that is tied up in physical things: product inventory, machinery and equipment, real estate, etc. Formerly referred to in TOC as Inventory.
Operating Expense	Money spent to create throughput, other than truly variable costs (e.g. payroll, utilities, taxes, etc.). The cost of maintaining a given level of capacity.

1. **Net Profit** = Throughput − Operating Expenses

2. **Return on Investment** = Net Profit / Investment
3. **Productivity = Throughput** / Operating Expenses
4. **Investment Turns = Throughput** / Investment

Internal and External constraints

1. An internal constraint is usually within the control of the business.

2. An external constraint is outside the business and is difficult to control.

Internal constraints

These are factors within the control of the business that are restricting it achieving its objectives

The main internal constraints are:-

1. **Finance** - has the business enough money to be able to finance growth?

   -If there was not enough cash within the business itself then is the business strong enough financially to be able to borrow money?

   -Is the business profitable?

2. **Marketing** - has the business got a strong marketing position?

   -Does it have a brand name and a suitable image?

   -Can it supply an increasing market should it grow?

3. **People** - have the staff the skills to do the jobs that they are expected to perform?

   - Does the business support training to make sure staff can do the job?

4. **Production** - has the business extra capacity to increase production if and when necessary?

External constraints

A constraint is something that stops you doing something. So, in business, an external constraint is something that exists outside of the business that stops the business doing whatever it wants.

Examples are:

1. Laws

2. Consumer Protection Agencies

3. Pressure Groups

4. The Government

External constraints are constraints that are thrust upon a company.

The company has no (or little) control over the constraint.

The company must build their product and systems around that constraint.

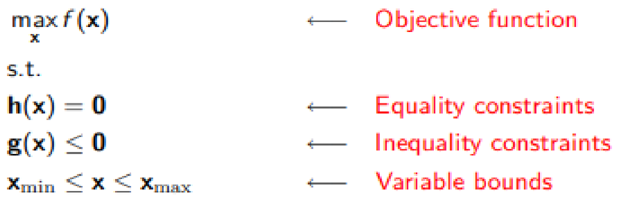
They must learn to live with it.

Often internal and external constraints go hand in hand and there is a fine dividing line between them.

For example interest rates are an external constraint and this causes an internal constraint in that it makes the borrowing money more expensive for the business.

Formulation of optimization problems

Optimization models (also called mathematical programs) represent problem choices as decision variables and seek values that maximize (or minimize) objective functions of the decision variables subject to constraints on variable values expressing the limits on possible decision choices.



$\max_{\mathbf{x}} f(\mathbf{x})$ ⟵ Objective function

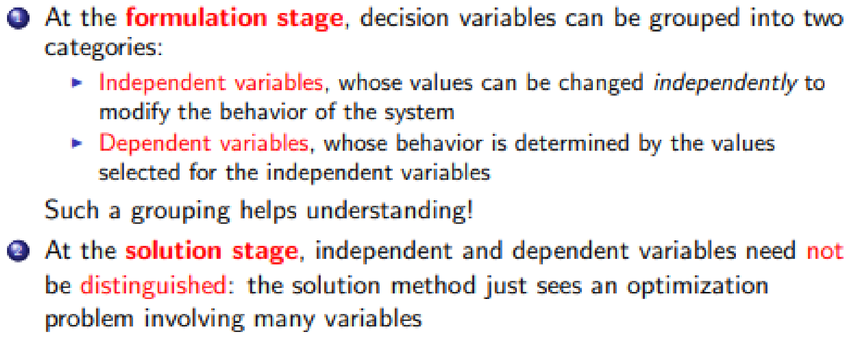
s.t.

$\mathbf{h}(\mathbf{x}) = \mathbf{0}$ ⟵ Equality constraints

$\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ ⟵ Inequality constraints

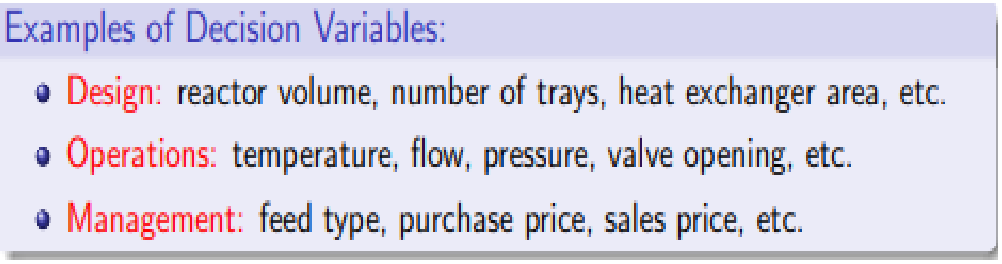
$\mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max}$ ⟵ Variable bounds

Decision Variables, x



- At the **formulation stage**, decision variables can be grouped into two categories:
  - Independent variables, whose values can be changed *independently* to modify the behavior of the system
  - Dependent variables, whose behavior is determined by the values selected for the independent variables

  Such a grouping helps understanding!
- At the **solution stage**, independent and dependent variables need not be distinguished: the solution method just sees an optimization problem involving many variables



Examples of Decision Variables:

- Design: reactor volume, number of trays, heat exchanger area, etc.
- Operations: temperature, flow, pressure, valve opening, etc.
- Management: feed type, purchase price, sales price, etc.

Module III

Necessary and sufficient conditions for optimum of unconstrained functions

Points of maxima and minima (extrema).

An extreme point of a function $f(\mathbf{X})$ defines either a maximum or a minimum of the function. Mathematically, a point $\mathbf{X}_0 = (x_1^0, \ldots, x_j^0, \ldots, x_n^0)$ is a maximum if

$$f(\mathbf{X}_0 + \mathbf{h}) \leq f(\mathbf{X}_0)$$

for all $\mathbf{h} = (h_1, \ldots, h_j, \ldots, h_n)$ where $|h_j|$ is sufficiently small for all j. In other words, $\mathbf{X}_0$ is a maximum if the value of f at every point in the neighborhood of $\mathbf{X}_0$ does not exceed $f(\mathbf{X}_0)$. In a similar manner, $\mathbf{X}_0$ is a minimum if

$$f(\mathbf{X}_0 + \mathbf{h}) \geq f(\mathbf{X}_0)$$

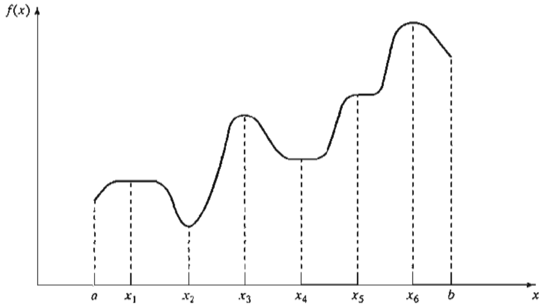


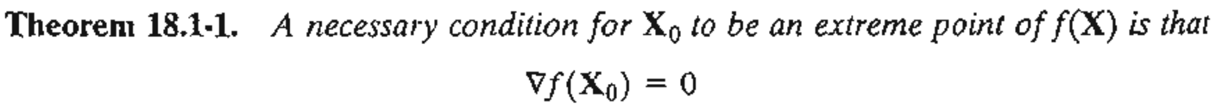
FIGURE 18.1
Examples of extreme points for a single-variable function

Although x_1 (in Figure 18.1) is a maximum point, it differs from remaining local maxima in that the value of f corresponding to at least one point in the neighborhood of x_1 equals $f(x_1)$. In this respect, x_1 is a **weak maximum**, whereas x_3 and x_6 are **strong maxima**. In general, for $\mathbf{h}$ as defined earlier, $\mathbf{X}_0$ is a weak maximum if $f(\mathbf{X}_0 + \mathbf{h}) \leq f(\mathbf{X}_0)$ and a strong maximum if $f(\mathbf{X}_0 + \mathbf{h}) < f(\mathbf{X}_0)$.

In Figure 18.1, the first derivative (slope) of f equals zero at all extrema. However, this property is also satisfied at **inflection** and **saddle** points, such as x_5. If a point with zero slope (gradient) is not an extremum (maximum or minimum), then it must be an inflection or a saddle point.

Necessary condition

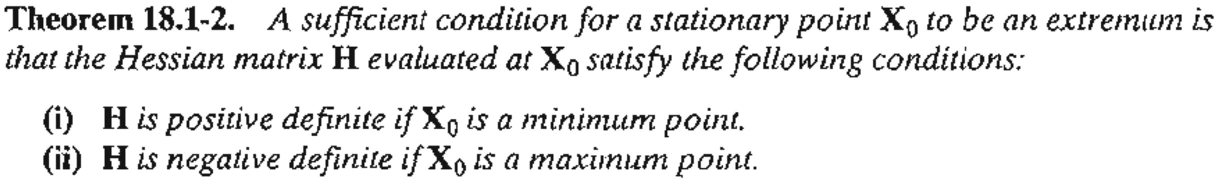
It is assumed that the first and second partial derivatives *of f(X)* are continuous for all X.



Theorem 18.1-1. *A necessary condition for $\mathbf{X}_0$ to be an extreme point of $f(\mathbf{X})$ is that*

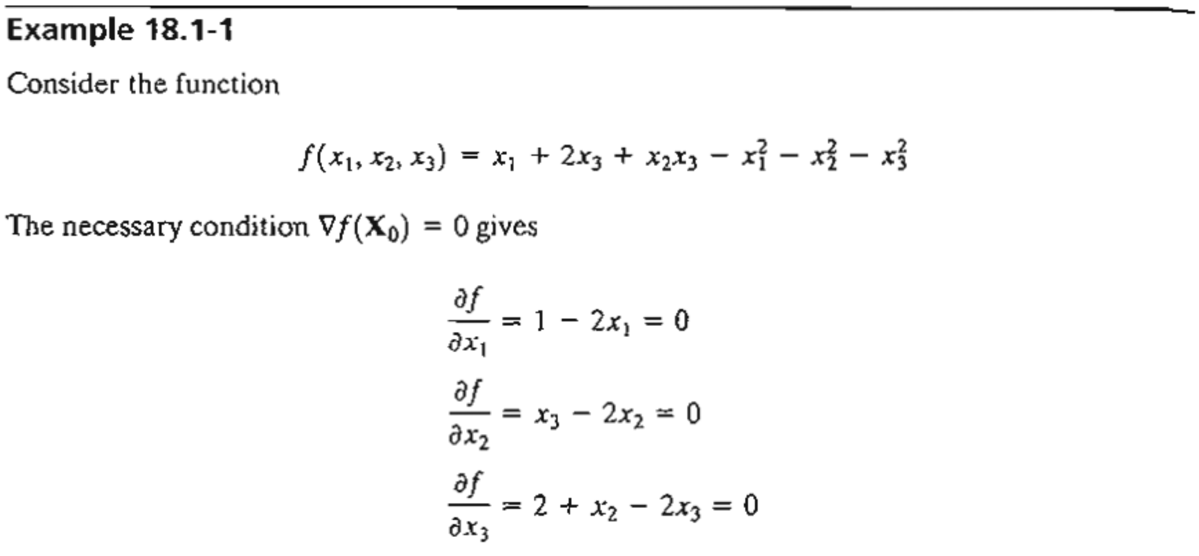
$$\nabla f(\mathbf{X}_0) = 0$$

Sufficient condition



Theorem 18.1-2. *A sufficient condition for a stationary point $\mathbf{X}_0$ to be an extremum is that the Hessian matrix $\mathbf{H}$ evaluated at $\mathbf{X}_0$ satisfy the following conditions:*

(i) $\mathbf{H}$ *is positive definite if* $\mathbf{X}_0$ *is a minimum point.*
(ii) $\mathbf{H}$ *is negative definite if* $\mathbf{X}_0$ *is a maximum point.*



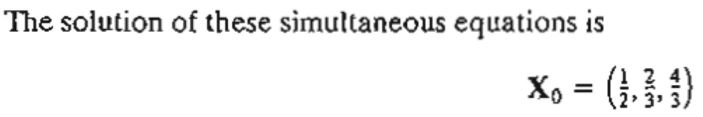
Example 18.1-1

Consider the function

$$f(x_1, x_2, x_3) = x_1 + 2x_3 + x_2x_3 - x_1^2 - x_2^2 - x_3^2$$

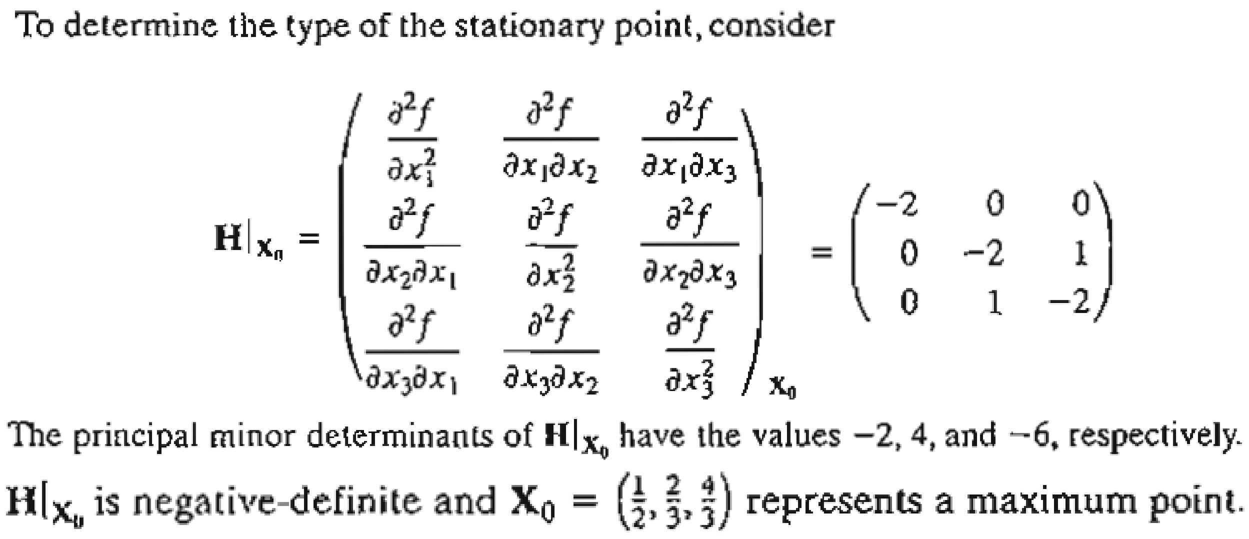
The necessary condition $\nabla f(\mathbf{X}_0) = 0$ gives

$$\frac{\partial f}{\partial x_1} = 1 - 2x_1 = 0$$
$$\frac{\partial f}{\partial x_2} = x_3 - 2x_2 = 0$$
$$\frac{\partial f}{\partial x_3} = 2 + x_2 - 2x_3 = 0$$



The solution of these simultaneous equations is

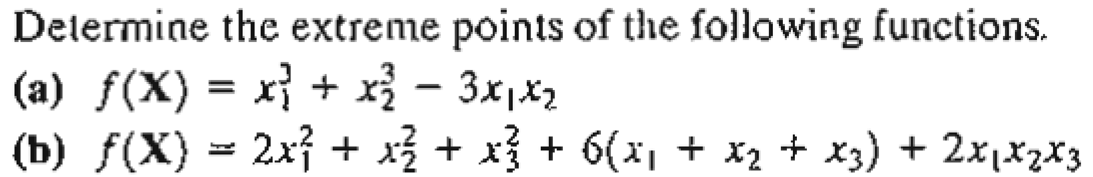
$$\mathbf{X}_0 = \left(\tfrac{1}{2}, \tfrac{2}{3}, \tfrac{4}{3}\right)$$



To determine the type of the stationary point, consider

$$\mathbf{H}|_{\mathbf{X}_0} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{pmatrix}_{\mathbf{X}_0} = \begin{pmatrix} -2 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 1 & -2 \end{pmatrix}$$

The principal minor determinants of $\mathbf{H}|_{\mathbf{X}_0}$ have the values -2, 4, and -6, respectively. $\mathbf{H}|_{\mathbf{X}_0}$ is negative-definite and $\mathbf{X}_0 = \left(\frac{1}{2}, \frac{2}{3}, \frac{4}{3}\right)$ represents a maximum point.



Determine the extreme points of the following functions.

(a) $f(\mathbf{X}) = x_1^3 + x_2^3 - 3x_1x_2$

(b) $f(\mathbf{X}) = 2x_1^2 + x_2^2 + x_3^2 + 6(x_1 + x_2 + x_3) + 2x_1x_2x_3$

Numerical methods for unconstrained functions

The Newton-Raphson Method

1. The necessary condition equations, $\nabla f(\mathbf{X}) = \mathbf{0}$, may be difficult to solve numerically

2. The Newton-Raphson method is an iterative procedure for solving simultaneous nonlinear equations.

3. The Newton method was originally developed by Newton for solving nonlinear equations and later refined by Raphson, and hence the method is also known as *Newton-Raphson method* in the literature of numerical analysis.

4.The method requires both the first- and second-order derivatives of f(X).

Consider the quadratic approximation of the function $f(\lambda)$ at $\lambda = \lambda_i$ using the Taylor's series expansion:

$$f(\lambda) = f(\lambda_i) + f'(\lambda_i)(\lambda - \lambda_i) + \tfrac{1}{2} f''(\lambda_i)(\lambda - \lambda_i)^2 \qquad (5.63)$$

By setting the derivative of Eq. (5.63) equal to zero for the minimum of $f(\lambda)$, we obtain

$$f'(\lambda) = f'(\lambda_i) + f''(\lambda_i)(\lambda - \lambda_i) = 0 \qquad (5.64)$$

If λ_i denotes an approximation to the minimum of $f(\lambda)$, Eq. (5.64) can be rearranged to obtain an improved approximation as

$$\lambda_{i+1} = \lambda_i - \frac{f'(\lambda_i)}{f''(\lambda_i)} \qquad (5.65)$$

Thus the *Newton method*, Eq. (5.65), is equivalent to using a quadratic approximation for the function $f(\lambda)$ and applying the necessary conditions. The iterative process given by Eq. (5.65) can be assumed to have converged when the derivative, $f'(\lambda_{i+1})$, is close to zero:

$$|f'(\lambda_{i+1})| \leq \epsilon \qquad (5.66)$$

where ϵ is a small quantity.

Determine the stationary points of the function

$$g(x) = (3x - 2)^2(2x - 3)^2$$

Starting with $x_0 = 10$

To determine the stationary points, we need to solve

$$f(x) \equiv g'(x) = 72x^3 - 234x^2 + 241x - 78 = 0$$

Thus, for the Newton-Raphson method, we have

$$f'(x) = 216x^2 - 468x + 241$$

$$x_{k+1} = x_k - \frac{72x^3 - 234x^2 + 241x - 78}{216x^2 - 468x + 241}$$

Starting with $x_0 = 10$ the following table provides the successive iterations:

k	x_k	$\dfrac{f(x_k)}{f'(x_k)}$	x_{k+1}
0	10.000000	2.978923	7.032108
1	7.032108	1.976429	5.055679
2	5.055679	1.314367	3.741312
3	3.741312	0.871358	2.869995
4	2.869995	0.573547	2.296405
5	2.296405	0.371252	1.925154
6	1.925154	0.230702	1.694452
7	1.694452	0.128999	1.565453
8	1.565453	0.054156	1.511296
9	1.511296	.0108641	1.500432
10	1.500432	.00043131	1.500001

The method converges to $x = 1.5$.

Numerical methods for unconstrained functions

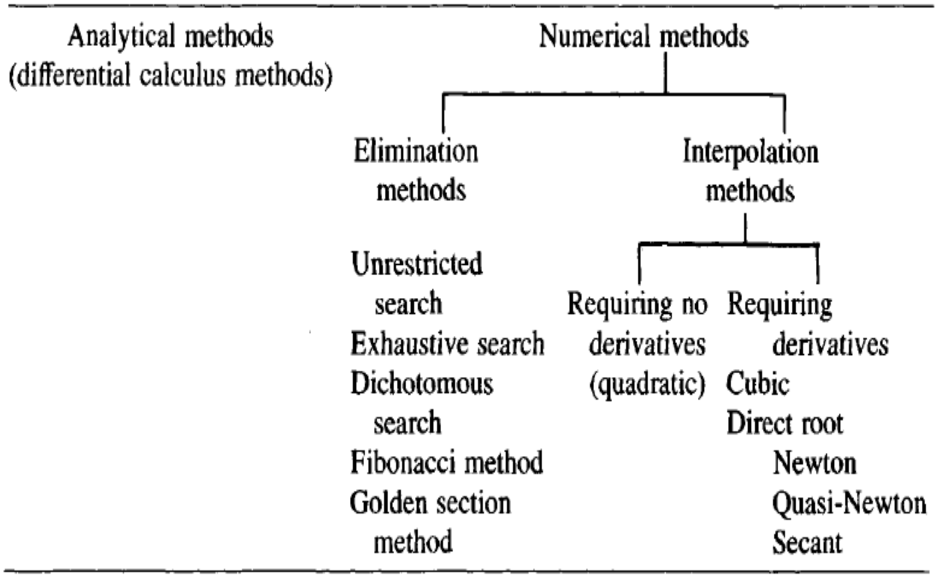
The basic philosophy of most of the numerical methods of optimization is to produce a sequence of improved approximations to the optimum according to the following scheme.

1. Start with an initial trial point $\mathbf{X}_1$.
2. Find a suitable direction $\mathbf{S}_i$ ($i = 1$ to start with) which points in the general direction of the optimum.
3. Find an appropriate step length λ_i^* for movement along the direction $\mathbf{S}_i$.
4. Obtain the new approximation $\mathbf{X}_{i+1}$ as

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \lambda_i^* \mathbf{S}_i \qquad (5.1)$$

5. Test whether $\mathbf{X}_{i+1}$ is optimum. If $\mathbf{X}_{i+1}$ is optimum, stop the procedure. Otherwise, set a new $i = i + 1$ and repeat step (2) onward.

TABLE 5.1 One-Dimensional Minimization Methods



Analytical methods (differential calculus methods)	Numerical methods		
	Elimination methods	Interpolation methods	
		Requiring no derivatives (quadratic)	Requiring derivatives
	Unrestricted search Exhaustive search Dichotomous search Fibonacci method Golden section method		Cubic Direct root Newton Quasi-Newton Secant

ELIMINATION METHODS

UNRESTRICTED SEARCH

1. In most practical problems, the optimum solution is known to lie within restricted ranges of the design variables.

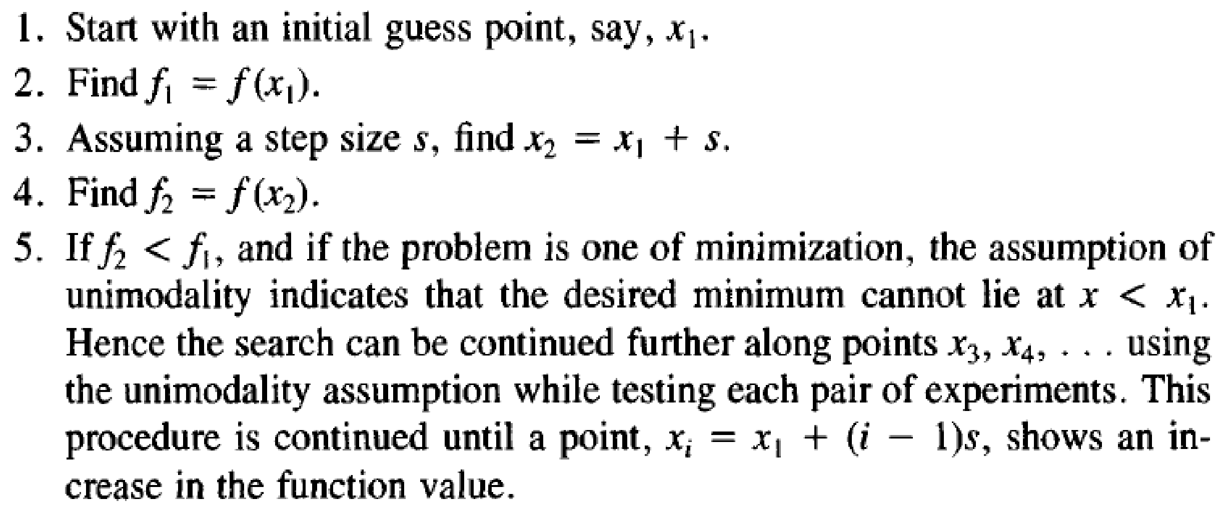
2. In some cases this range is not known, and hence the search has to be made with no restrictions on the values of the variables.

Search with Fixed Step Size

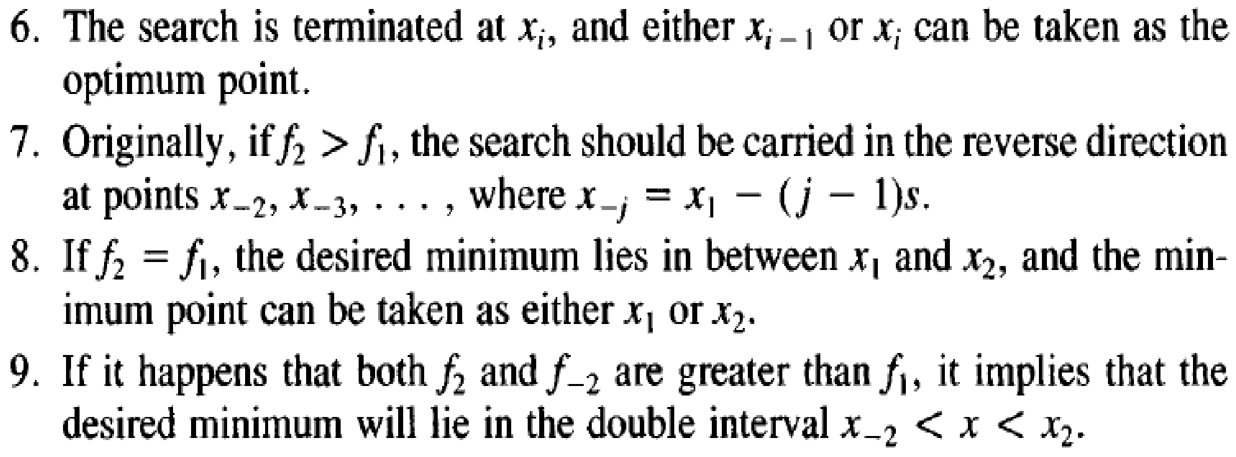
1. The most elementary approach for such a problem is to use a fixed step size and move from an initial guess point in a favorable direction (positive or negative).

2. The step size used must be small in relation to the final accuracy desired.

This method is described in the following steps.



1. Start with an initial guess point, say, x_1.
2. Find $f_1 = f(x_1)$.
3. Assuming a step size s, find $x_2 = x_1 + s$.
4. Find $f_2 = f(x_2)$.
5. If $f_2 < f_1$, and if the problem is one of minimization, the assumption of unimodality indicates that the desired minimum cannot lie at $x < x_1$. Hence the search can be continued further along points $x_3, x_4, \ldots$ using the unimodality assumption while testing each pair of experiments. This procedure is continued until a point, $x_i = x_1 + (i - 1)s$, shows an increase in the function value.



6. The search is terminated at x_i, and either x_{i-1} or x_i can be taken as the optimum point.
7. Originally, if $f_2 > f_1$, the search should be carried in the reverse direction at points $x_{-2}, x_{-3}, \ldots$, where $x_{-j} = x_1 - (j - 1)s$.
8. If $f_2 = f_1$, the desired minimum lies in between x_1 and x_2, and the minimum point can be taken as either x_1 or x_2.
9. If it happens that both f_2 and f_{-2} are greater than f_1, it implies that the desired minimum will lie in the double interval $x_{-2} < x < x_2$.

Linear Programming

1. *Linear programming* is an optimization method applicable for the solution of problems in which the objective function and the constraints appear as linear functions of the decision variables.

2. The constraint equations in a linear programming problem may be in the form of equalities or inequalities

3. The linear programming type of optimization problem was first recognized in the 1930s by economists while developing methods for the optimal allocation of resources.

4. During World War II the U.S. Air Force sought more effective procedures of allocating resources and turned to linear programming.

5. George B.Dantzig, who was a member of the Air Force group, formulated the general linear programming problem and devised the simplex method of solution in 1947.

6. Linear programming is considered a revolutionary development that permits us to make optimal decisions in complex situations.

7. At least four Nobel Prizes were awarded for contributions related to linear programming.

8. The simplex method continues to be the most efficient and popular method for solving general LP problems.

APPLICATIONS OF LINEAR PROGRAMMING

1. One of the early industrial applications of linear programming has been made in the petroleum refineries.

   In general, an oil refinery has a choice of buying crude oil from several different sources with differing compositions and at differing prices.

   It can manufacture different products, such as aviation fuel, diesel fuel, and gasoline, in varying quantities.

   The constraints may be due to the restrictions on the quantity of the crude oil available from a particular source, the capacity of the refinery to produce a particular product, and so on.

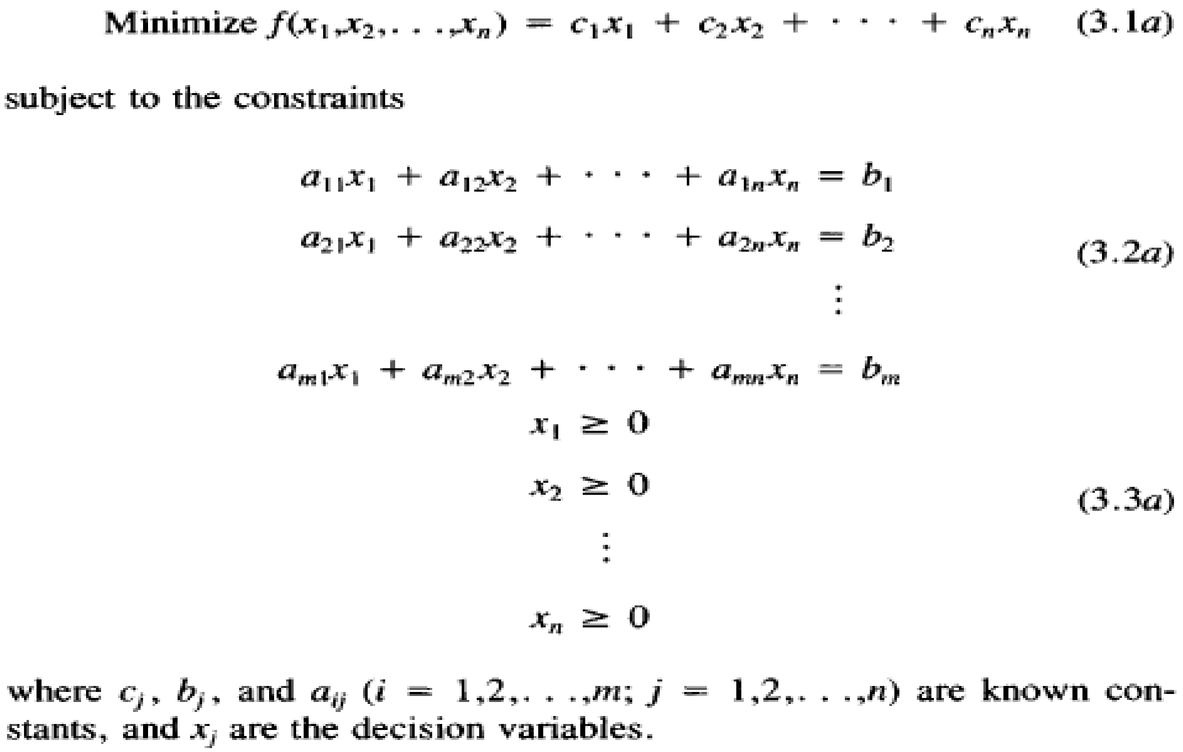
   A mix of the purchased crude oil and the manufactured products is sought that gives the maximum profit.

2. One of the early industrial applications of linear programming has been made in the petroleum refineries.
3. The optimal production plan in a manufacturing firm can also be decided using linear programming.
4. In the food-processing industry, linear programming has been used to determine the optimal shipping plan for the distribution of a particular product from different manufacturing plants to various warehouses.
5. In the iron and steel industry, linear programming was used to decide the types of products to be made in their rolling mills to maximize the profit.
6. The optimal production plan in a manufacturing firm can also be decided using linear programming.
7. Paper mills use it to decrease the amount of trim losses.
8. The optimal routing of messages in a communication network and the routing of aircraft and ships can also be decided using linear programming.
9. Linear programming has also been applied to formulate and solve several types of engineering design problems, such as the plastic design of frame structures.

STANDARD FORM OF A LINEAR PROGRAMMING PROBLEM

The general linear programming problem can be stated in the following standard form:

1. *Scalar form*



$$\text{Minimize } f(x_1,x_2,\ldots,x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n \qquad (3.1a)$$

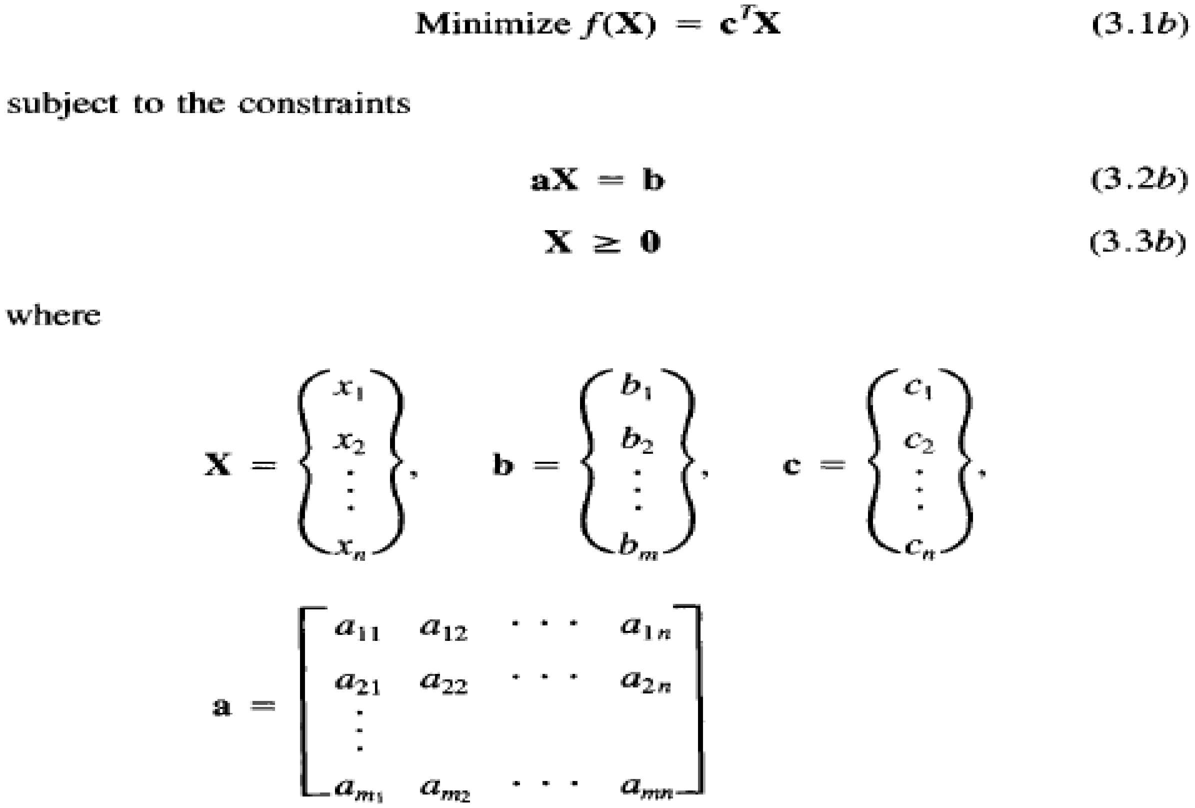
subject to the constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \qquad (3.2a)$$

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ &\vdots \\ x_n &\geq 0 \end{aligned} \qquad (3.3a)$$

where c_j, b_j, and a_{ij} $(i = 1,2,\ldots,m;\ j = 1,2,\ldots,n)$ are known constants, and x_j are the decision variables.

2. *Matrix form*



$$\text{Minimize } f(\mathbf{X}) = \mathbf{c}^T\mathbf{X} \tag{3.1b}$$

subject to the constraints

$$\mathbf{aX} = \mathbf{b} \tag{3.2b}$$

$$\mathbf{X} \geq \mathbf{0} \tag{3.3b}$$

where

$$\mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}, \quad \mathbf{b} = \begin{Bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{Bmatrix}, \quad \mathbf{c} = \begin{Bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{Bmatrix},$$

$$\mathbf{a} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m_1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Characteristics of a linear programming

The characteristics of a linear programming problem, stated in the standard form, are:

1. The objective function is of the minimization type.

2. All the constraints are of the equality type.

3. All the decision variables are nonnegative.

Feasible Solution **In** a linear programming problem, any solution that satisfies the constraints

aX = b

X ≥ 0

is called a feasible solution.

Definitions

Basic Solution A basic solution is one in which *n* — *m* variables are set equal to zero. A basic solution can be obtained by setting *n* — *m* variables to zero and solving the constraint Eqs. aX = b simultaneously.

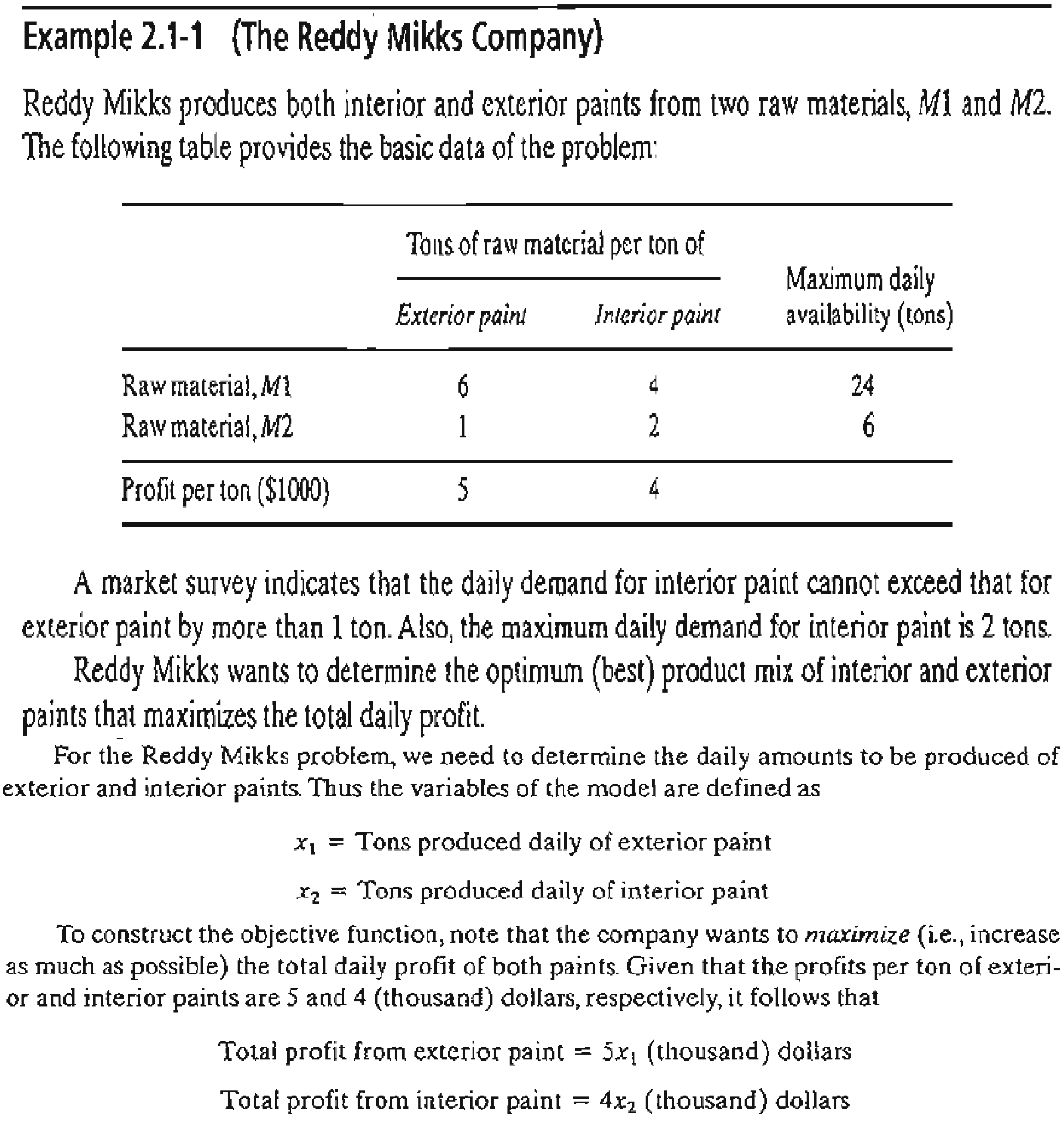
Basis The collection of variables not set equal to zero to obtain the basic solution is called the basis.

Basic Feasible Solution This is a basic solution that satisfies the nonnegativity conditions of Eq. X ≥ 0

Nondegenerate Basic Feasible Solution This is a basic feasible solution that has got exactly *m* positive *Xi*.

Optimal Solution A feasible solution that optimizes the objective function is called an optimal solution.

Optimal Basic Solution This is a basic feasible solution for which the objective function is optimal.



Example 2.1-1 (The Reddy Mikks Company)

Reddy Mikks produces both interior and exterior paints from two raw materials, *M*1 and *M*2. The following table provides the basic data of the problem:

	Tons of raw material per ton of		Maximum daily availability (tons)
	Exterior paint	*Interior paint*	
Raw material, *M*1	6	4	24
Raw material, *M*2	1	2	6
Profit per ton ($1000)	5	4	

A market survey indicates that the daily demand for interior paint cannot exceed that for exterior paint by more than 1 ton. Also, the maximum daily demand for interior paint is 2 tons.

Reddy Mikks wants to determine the optimum (best) product mix of interior and exterior paints that maximizes the total daily profit.

For the Reddy Mikks problem, we need to determine the daily amounts to be produced of exterior and interior paints. Thus the variables of the model are defined as

$$x_1 = \text{Tons produced daily of exterior paint}$$

$$x_2 = \text{Tons produced daily of interior paint}$$

To construct the objective function, note that the company wants to *maximize* (i.e., increase as much as possible) the total daily profit of both paints. Given that the profits per ton of exterior and interior paints are 5 and 4 (thousand) dollars, respectively, it follows that

$$\text{Total profit from exterior paint} = 5x_1 \text{ (thousand) dollars}$$

$$\text{Total profit from interior paint} = 4x_2 \text{ (thousand) dollars}$$

Letting z represent the total daily profit (in thousands of dollars), the objective of the company is

$$\text{Maximize } z = 5x_1 + 4x_2$$

Next, we construct the constraints that restrict raw material usage and product demand. The raw material restrictions are expressed verbally as

$$\begin{pmatrix}\text{Usage of a raw material}\\ \text{by both paints}\end{pmatrix} \leq \begin{pmatrix}\text{Maximum raw material}\\ \text{availability}\end{pmatrix}$$

The daily usage of raw material $M1$ is 6 tons per ton of exterior paint and 4 tons per ton of interior paint. Thus

$$\text{Usage of raw material } M1 \text{ by exterior paint} = 6x_1 \text{ tons/day}$$

$$\text{Usage of raw material } M1 \text{ by interior paint} = 4x_2 \text{ tons/day}$$

Hence

$$\text{Usage of raw material } M1 \text{ by both paints} = 6x_1 + 4x_2 \text{ tons/day}$$

In a similar manner,

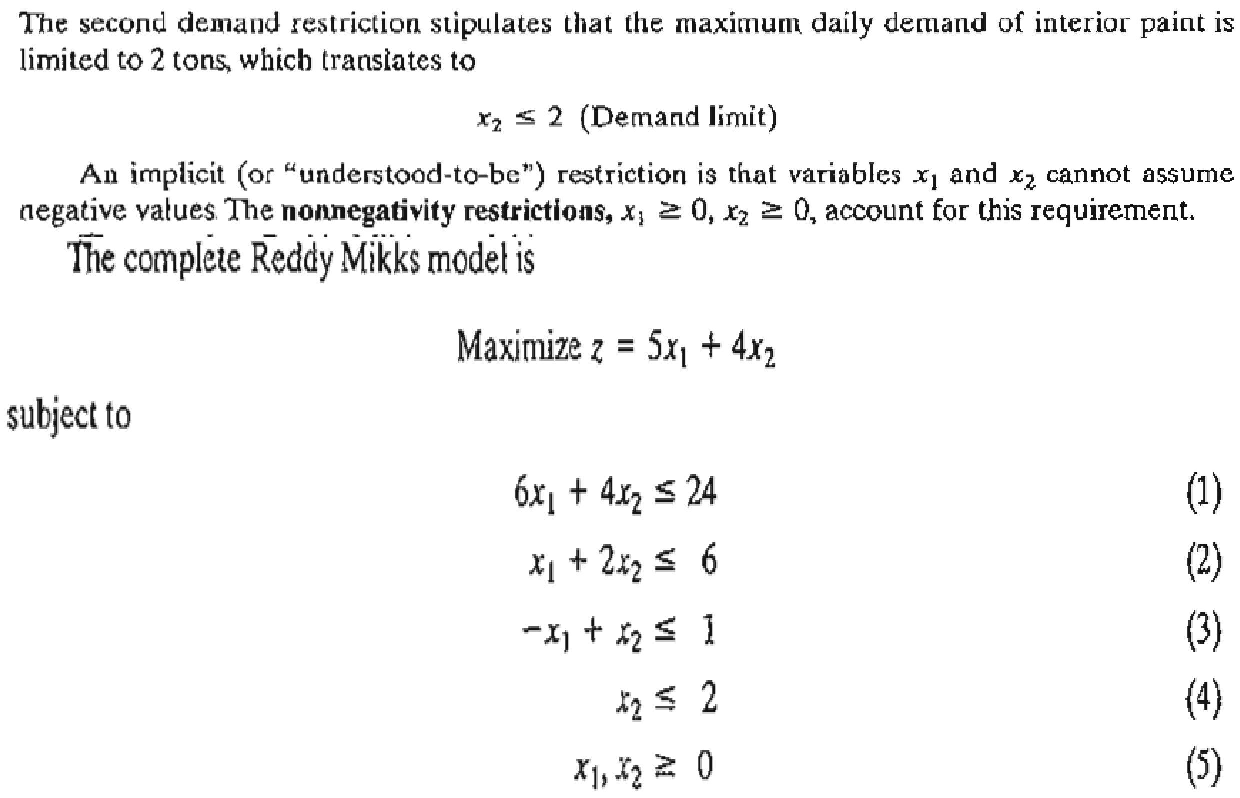
$$\text{Usage of raw material } M2 \text{ by both paints} = 1x_1 + 2x_2 \text{ tons/day}$$

Because the daily availabilities of raw materials $M1$ and $M2$ are limited to 24 and 6 tons, respectively, the associated restrictions are given as

$$\begin{aligned} 6x_1 + 4x_2 &\leq 24 \quad \text{(Raw material } M1\text{)} \\ x_1 + 2x_2 &\leq \ \ 6 \quad \text{(Raw material } M2\text{)} \end{aligned}$$

The first demand restriction stipulates that the excess of the daily production of interior over exterior paint, $x_2 - x_1$, should not exceed 1 ton, which translates to

$$x_2 - x_1 \leq 1 \qquad \text{(Market limit)}$$



The second demand restriction stipulates that the maximum daily demand of interior paint is limited to 2 tons, which translates to

$$x_2 \leq 2 \text{ (Demand limit)}$$

An implicit (or "understood-to-be") restriction is that variables x_1 and x_2 cannot assume negative values. The **nonnegativity restrictions,** $x_1 \geq 0$, $x_2 \geq 0$, account for this requirement.

The complete Reddy Mikks model is

$$\text{Maximize } z = 5x_1 + 4x_2$$

subject to

$$6x_1 + 4x_2 \leq 24 \quad (1)$$

$$x_1 + 2x_2 \leq 6 \quad (2)$$

$$-x_1 + x_2 \leq 1 \quad (3)$$

$$x_2 \leq 2 \quad (4)$$

$$x_1, x_2 \geq 0 \quad (5)$$

GRAPHICAL LP SOLUTION

The graphical procedure includes two steps:

1. Determination of the feasible solution space.

2. Determination of the optimum solution from among all the feasible points in the solution space.

Solution of a Maximization Model

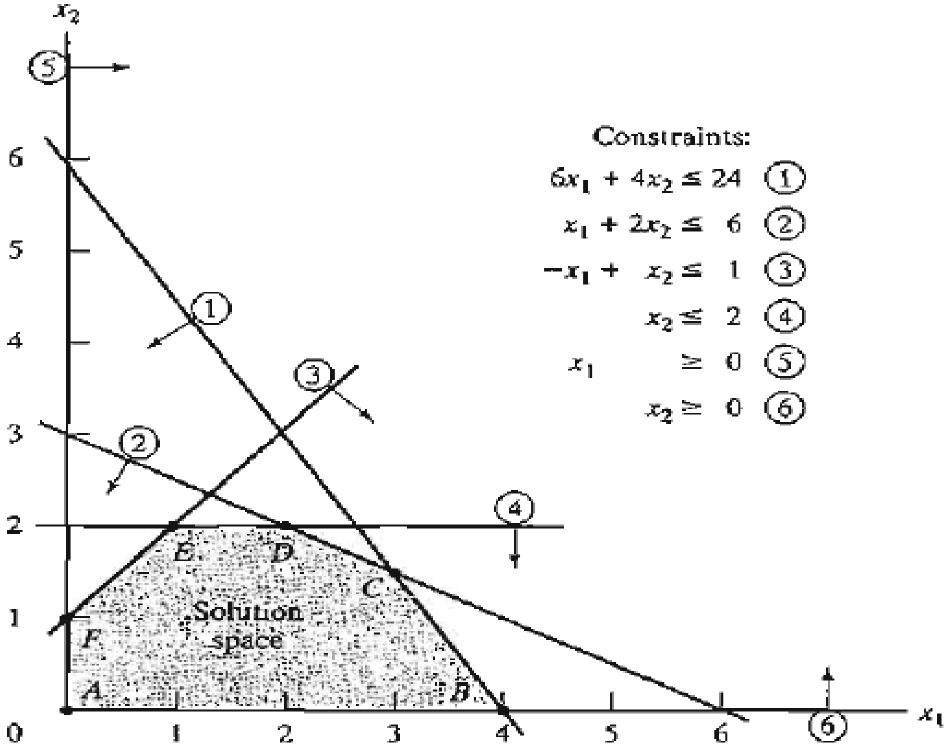
This example solves the Reddy Mikks model of Example 2.1-1.

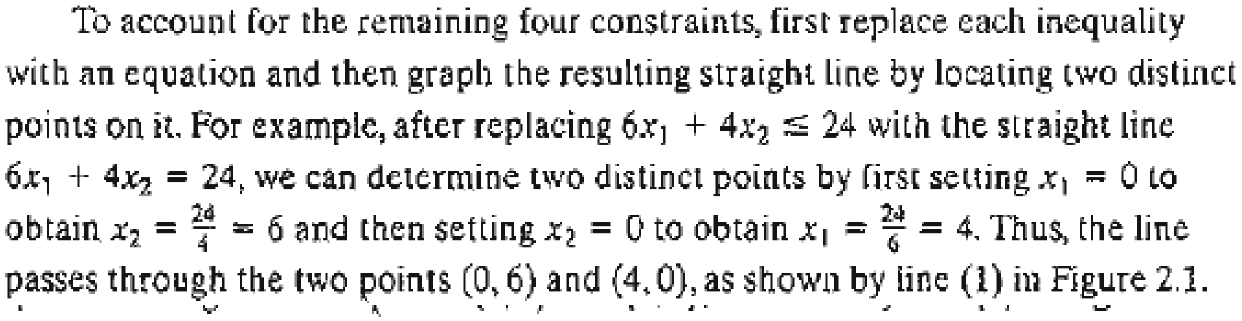
Step 1. *Determination ofthe Feasible Solution Space:*

First, we account for the nonnegativity constraints $x_1 \ge 0$ and $x_2 \ge 0$. In Figure 2.1, the horizontal axis x_1 and the vertical axis x_2 represent the exterior- and interior-paint variables, respectively. Thus, the nonnegativity of the variables restricts the solution-space area to the first quadrant that lies above the x_1-axis and to the right of the x_2-axis.

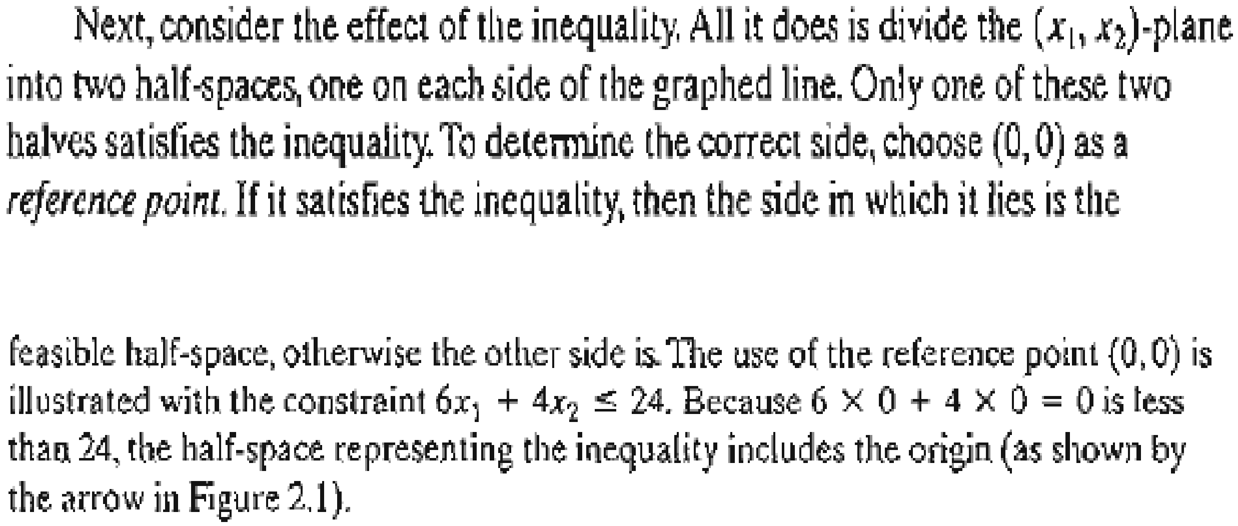
FIGURE 2.1

Feasible space of the Reddy Mikks model

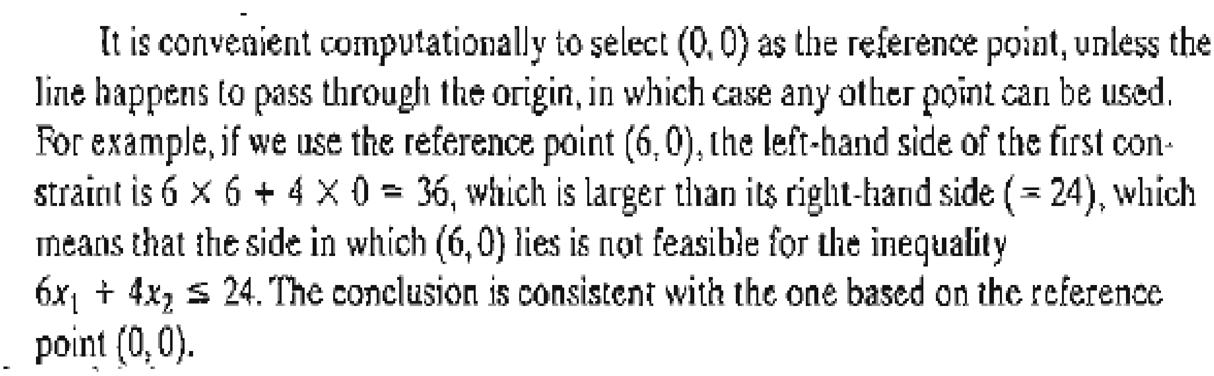




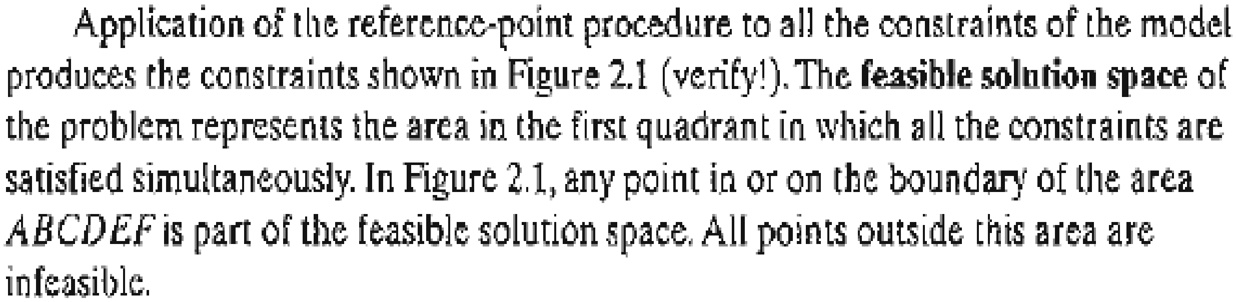
To account for the remaining four constraints, first replace each inequality with an equation and then graph the resulting straight line by locating two distinct points on it. For example, after replacing $6x_1 + 4x_2 \leq 24$ with the straight line $6x_1 + 4x_2 = 24$, we can determine two distinct points by first setting $x_1 = 0$ to obtain $x_2 = \frac{24}{4} = 6$ and then setting $x_2 = 0$ to obtain $x_1 = \frac{24}{6} = 4$. Thus, the line passes through the two points (0, 6) and (4, 0), as shown by line (1) in Figure 2.1.



Next, consider the effect of the inequality. All it does is divide the (x_1, x_2)-plane into two half-spaces, one on each side of the graphed line. Only one of these two halves satisfies the inequality. To determine the correct side, choose (0, 0) as a *reference point*. If it satisfies the inequality, then the side in which it lies is the feasible half-space, otherwise the other side is. The use of the reference point (0, 0) is illustrated with the constraint $6x_1 + 4x_2 \leq 24$. Because $6 \times 0 + 4 \times 0 = 0$ is less than 24, the half-space representing the inequality includes the origin (as shown by the arrow in Figure 2.1).



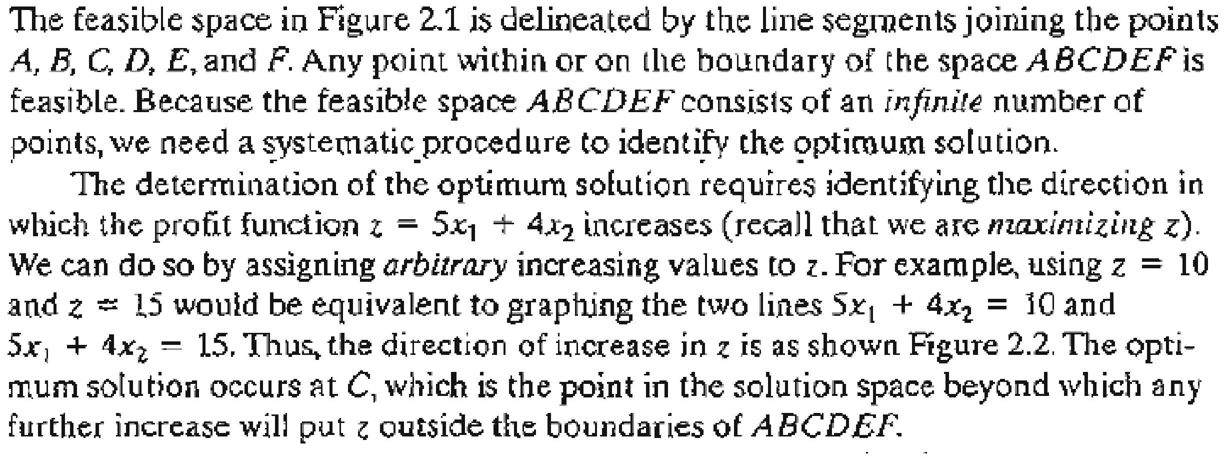
It is convenient computationally to select (0, 0) as the reference point, unless the line happens to pass through the origin, in which case any other point can be used. For example, if we use the reference point (6, 0), the left-hand side of the first constraint is $6 \times 6 + 4 \times 0 = 36$, which is larger than its right-hand side (= 24), which means that the side in which (6, 0) lies is not feasible for the inequality $6x_1 + 4x_2 \leq 24$. The conclusion is consistent with the one based on the reference point (0, 0).



Application of the reference-point procedure to all the constraints of the model produces the constraints shown in Figure 2.1 (verify!). The **feasible solution space** of the problem represents the area in the first quadrant in which all the constraints are satisfied simultaneously. In Figure 2.1, any point in or on the boundary of the area $ABCDEF$ is part of the feasible solution space. All points outside this area are infeasible.

Step 2.

Determination ofthe Optimum Solution:



The feasible space in Figure 2.1 is delineated by the line segments joining the points A, B, C, D, E, and F. Any point within or on the boundary of the space $ABCDEF$ is feasible. Because the feasible space $ABCDEF$ consists of an *infinite* number of points, we need a systematic procedure to identify the optimum solution.

The determination of the optimum solution requires identifying the direction in which the profit function $z = 5x_1 + 4x_2$ increases (recall that we are *maximizing* z). We can do so by assigning *arbitrary* increasing values to z. For example, using $z = 10$ and $z = 15$ would be equivalent to graphing the two lines $5x_1 + 4x_2 = 10$ and $5x_1 + 4x_2 = 15$. Thus, the direction of increase in z is as shown Figure 2.2. The optimum solution occurs at C, which is the point in the solution space beyond which any further increase will put z outside the boundaries of $ABCDEF$.

The values of x_1 and x_2 associated with the optimum point C are determined by solving the equations associated with lines (1) and (2)—that is,

$$6x_1 + 4x_2 = 24$$
$$x_1 + 2x_2 = 6$$

The solution is $x_1 = 3$ and $x_2 = 1.5$ with $z = 5 \times 3 + 4 \times 1.5 = 21$. This calls for a daily product mix of 3 tons of exterior paint and 1.5 tons of interior paint. The associated daily profit is \$21,000.

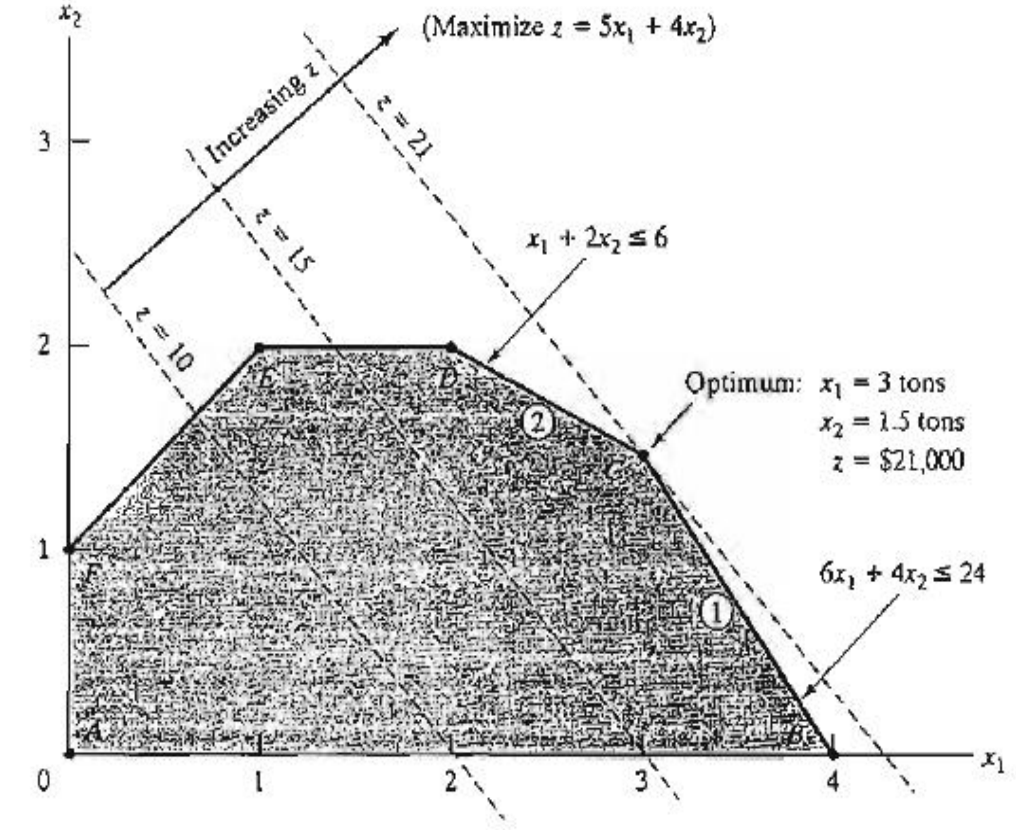


FIGURE 2.2

Optimum solution of the Reddy Mikks model

MOTIVATION OF THE SIMPLEX METHOD

one way to find the optimal solution of the given linear programming problem is to generate all the basic solutions and pick the one that is feasible and corresponds to the optimal value of the objective function.

If there are *m* equality constraints in *n* variables with $n \geq$ m, a basic solution can be obtained by setting any of the $n — m$ variables equal to zero.

The number of basic solutions to be inspected is thus equal to the number of ways in which m variables can be selected from a set of n variables, that is,

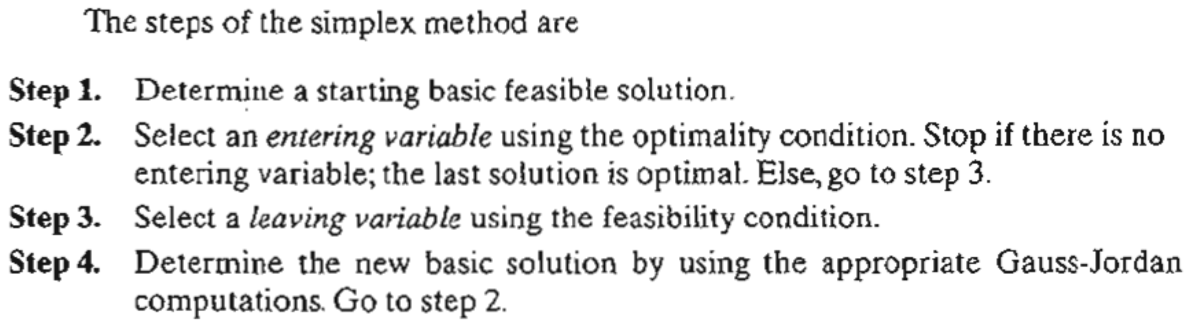
$$\binom{n}{m} = \frac{n!}{(n - m)!\ m!}$$

1. we do not have to inspect all these basic solutions since many of them will be infeasible.

2. The simplex method of Dantzig is a powerful scheme for obtaining a basic feasible solution; if the solution is not optimal, the method provides for finding a neighboring basic feasible solution that has a lower or equal value of f.

3. Rather than enumerating *all* the basic solutions (corner points) of the LP problem the simplex method investigates only a "select few" of these solutions.

Iterative Nature of the Simplex Method



The steps of the simplex method are

Step 1. Determine a starting basic feasible solution.

Step 2. Select an *entering variable* using the optimality condition. Stop if there is no entering variable; the last solution is optimal. Else, go to step 3.

Step 3. Select a *leaving variable* using the feasibility condition.

Step 4. Determine the new basic solution by using the appropriate Gauss-Jordan computations. Go to step 2.

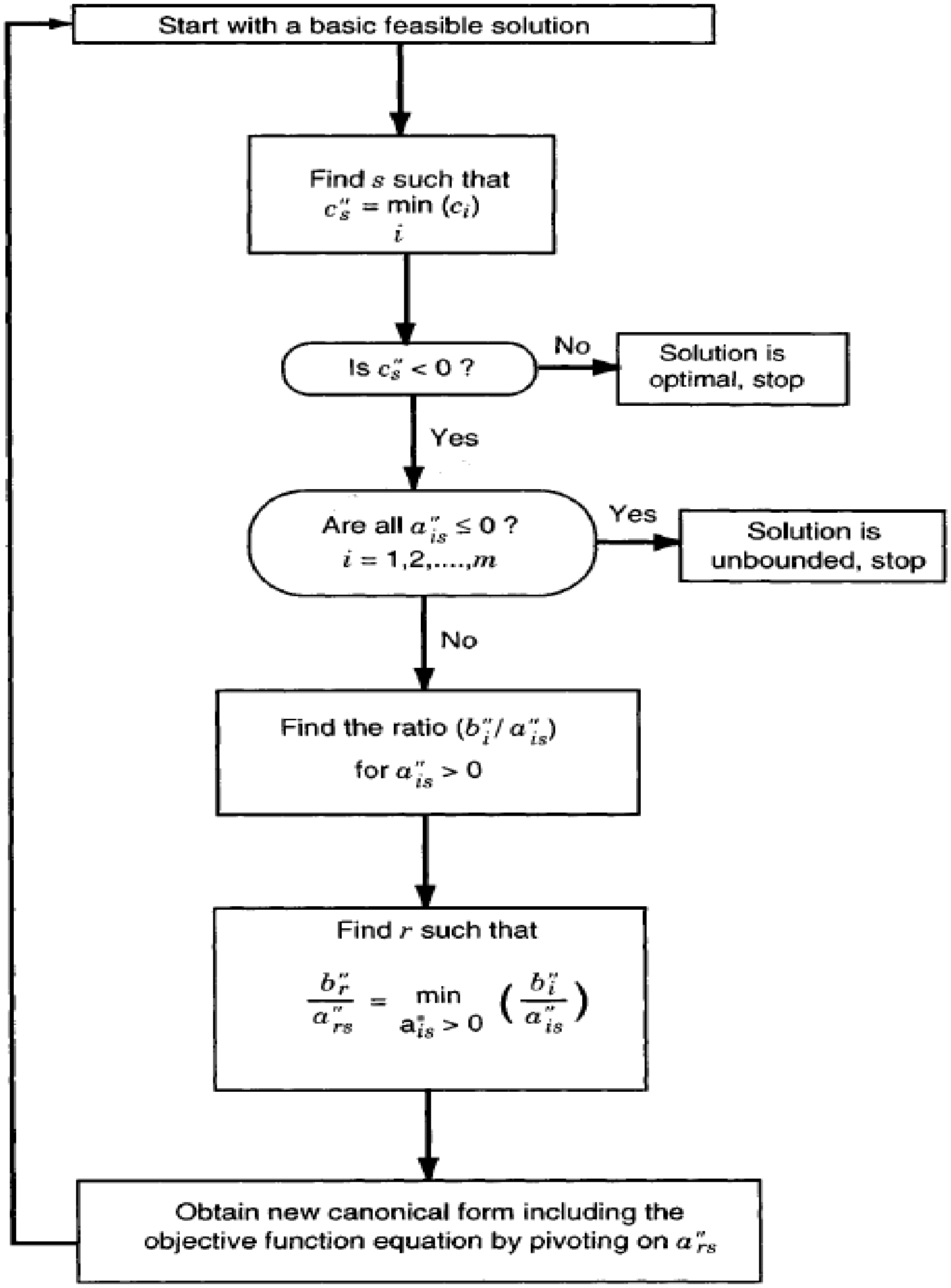
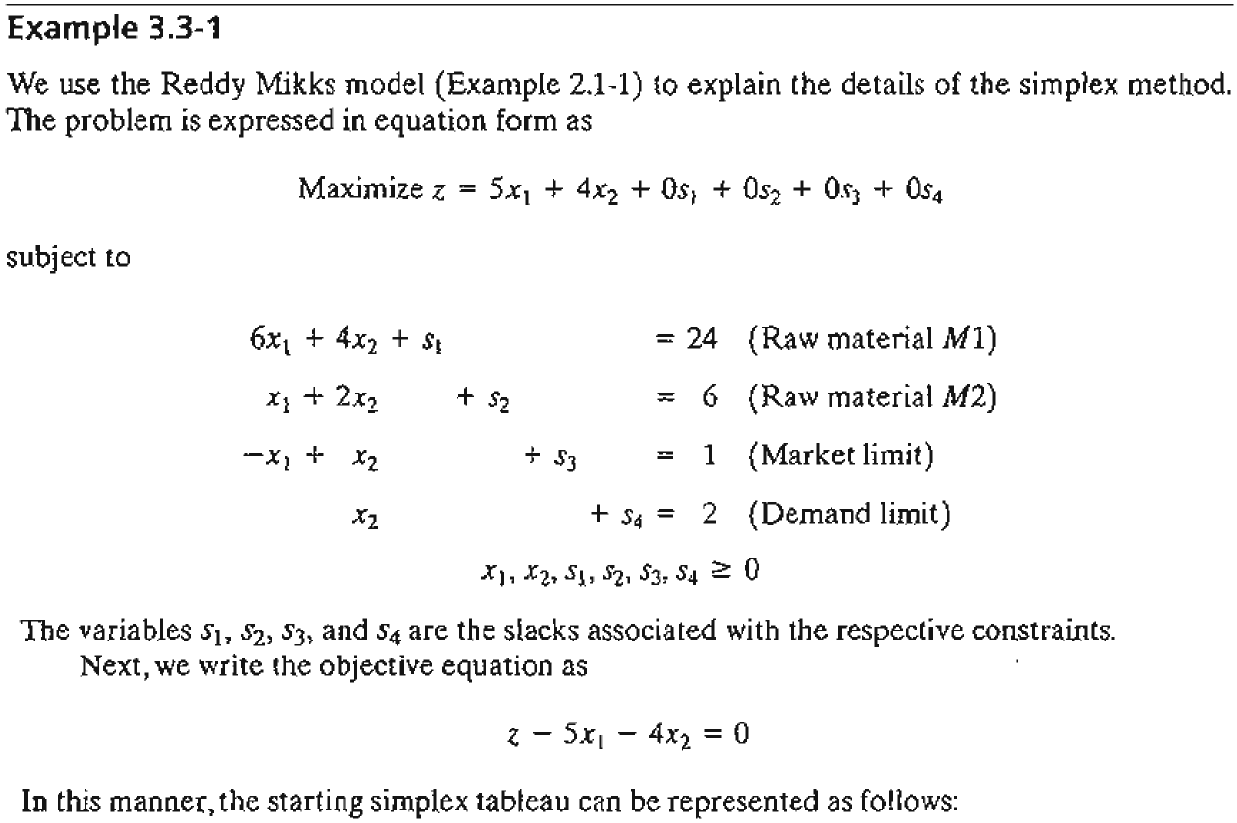

Start with a basic feasible solution
Find s such that $c''_s = \min_i (c_i)$
Is $c''_s < 0$?
No
Solution is optimal, stop
Yes
Are all $a''_{is} \leq 0$? $i = 1,2,....,m$
Yes
Solution is unbounded, stop
No
Find the ratio (b''_i / a''_{is}) for $a''_{is} > 0$
Find r such that $\frac{b''_r}{a''_{rs}} = \min_{a''_{is} > 0} \left(\frac{b''_i}{a''_{is}}\right)$
Obtain new canonical form including the objective function equation by pivoting on a''_{rs}

Figure 3.14 Flowchart for finding the optimal solution by the simplex algorithm.



Example 3.3-1

We use the Reddy Mikks model (Example 2.1-1) to explain the details of the simplex method. The problem is expressed in equation form as

$$\text{Maximize } z = 5x_1 + 4x_2 + 0s_1 + 0s_2 + 0s_3 + 0s_4$$

subject to

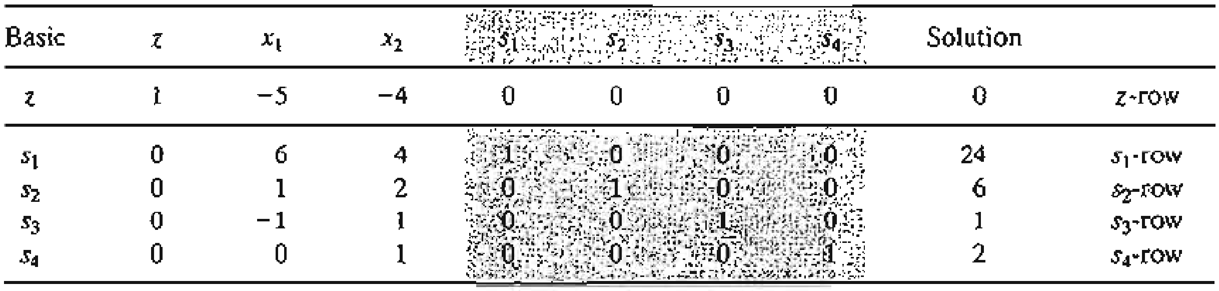
$$\begin{aligned} 6x_1 + 4x_2 + s_1 \quad &= 24 \quad \text{(Raw material } M1\text{)}\\ x_1 + 2x_2 + s_2 \quad &= 6 \quad \text{(Raw material } M2\text{)}\\ -x_1 + x_2 + s_3 \quad &= 1 \quad \text{(Market limit)}\\ x_2 + s_4 &= 2 \quad \text{(Demand limit)}\\ x_1, x_2, s_1, s_2, s_3, s_4 &\geq 0 \end{aligned}$$

The variables s_1, s_2, s_3, and s_4 are the slacks associated with the respective constraints.

Next, we write the objective equation as

$$z - 5x_1 - 4x_2 = 0$$

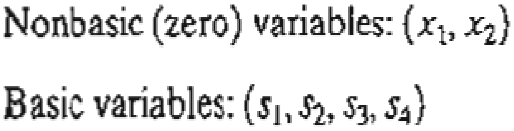
In this manner, the starting simplex tableau can be represented as follows:



Basic	z	x_1	x_2	s_1	s_2	s_3	s_4	Solution	
z	1	−5	−4	0	0	0	0	0	z-row
s_1	0	6	4	1	0	0	0	24	s_1-row
s_2	0	1	2	0	1	0	0	6	s_2-row
s_3	0	−1	1	0	0	1	0	1	s_3-row
s_4	0	0	1	0	0	0	1	2	s_4-row

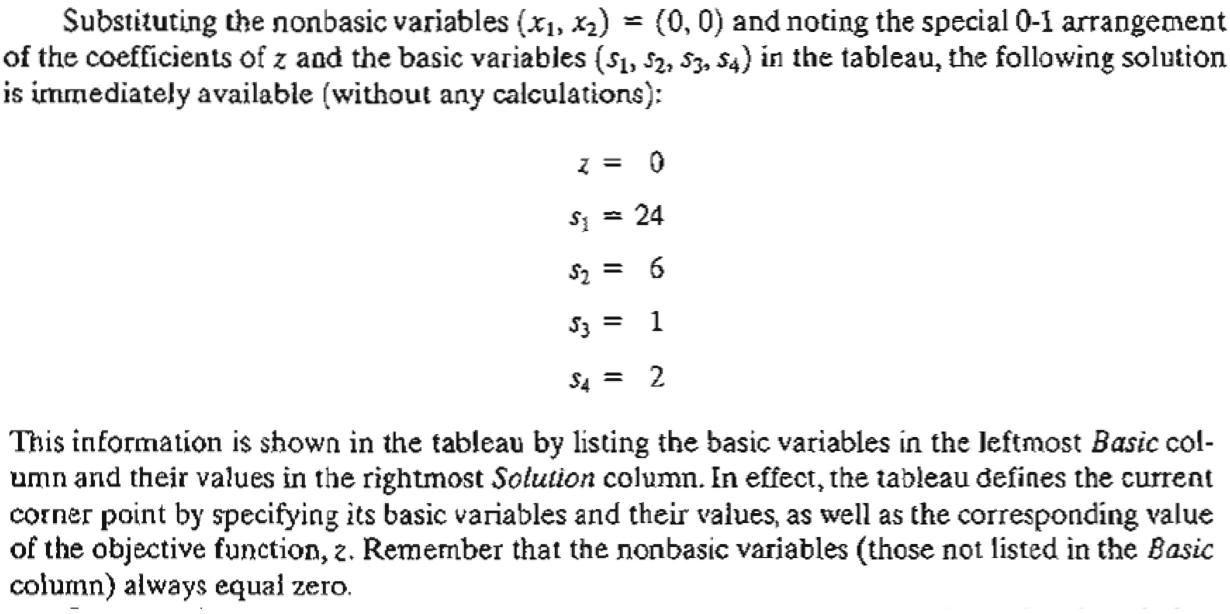
The design of the tableau specifies the set of basic and non-basic variables as well as provides the solution associated with the starting iteration

The simplex iterations start at the origin *(X1, X2)* = (0,0) whose associated set of nonbasic and basic variables are defined as



Nonbasic (zero) variables: (x_1, x_2)

Basic variables: (s_1, s_2, s_3, s_4)



Substituting the nonbasic variables $(x_1, x_2) = (0, 0)$ and noting the special 0-1 arrangement of the coefficients of z and the basic variables (s_1, s_2, s_3, s_4) in the tableau, the following solution is immediately available (without any calculations):

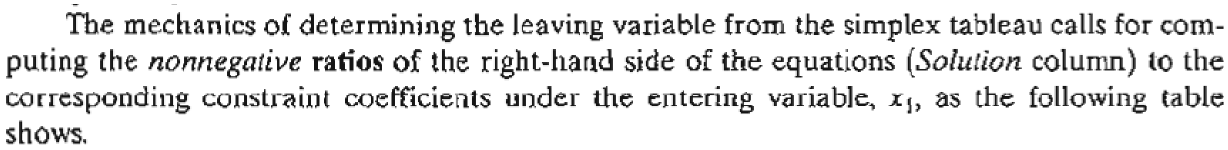
$$\begin{aligned} z &= 0 \\ s_1 &= 24 \\ s_2 &= 6 \\ s_3 &= 1 \\ s_4 &= 2 \end{aligned}$$

This information is shown in the tableau by listing the basic variables in the leftmost *Basic* column and their values in the rightmost *Solution* column. In effect, the tableau defines the current corner point by specifying its basic variables and their values, as well as the corresponding value of the objective function, z. Remember that the nonbasic variables (those not listed in the *Basic* column) always equal zero.

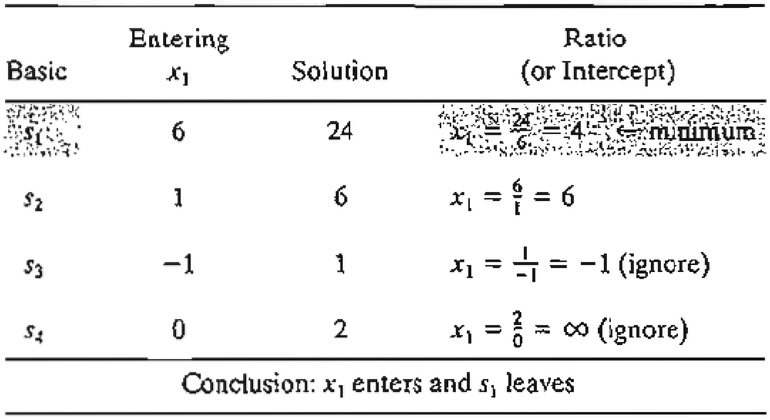
Is the starting solution optimal? The objective function z = 5x1 + 4x2 shows that the solution

can be improved by increasing X1 or X2.

X1 with the most positive coefficient is selected as the entering variable. Equivalently, because the simplex tableau expresses the objective function as z - 5x1 - 4x2 = 0, the entering variable will correspond to the variable with the most negative coefficient in the objective equation. This rule is referred to as the optimality condition.



The mechanics of determining the leaving variable from the simplex tableau calls for computing the *nonnegative* **ratios** of the right-hand side of the equations (*Solution* column) to the corresponding constraint coefficients under the entering variable, x_1, as the following table shows.



Basic	Entering x_1	Solution	Ratio (or Intercept)
s_1	6	24	$x_1 = \frac{24}{6} = 4 \leftarrow$ minimum
s_2	1	6	$x_1 = \frac{6}{1} = 6$
s_3	−1	1	$x_1 = \frac{1}{-1} = -1$ (ignore)
s_4	0	2	$x_1 = \frac{2}{0} = \infty$ (ignore)
Conclusion: x_1 enters and s_1 leaves			

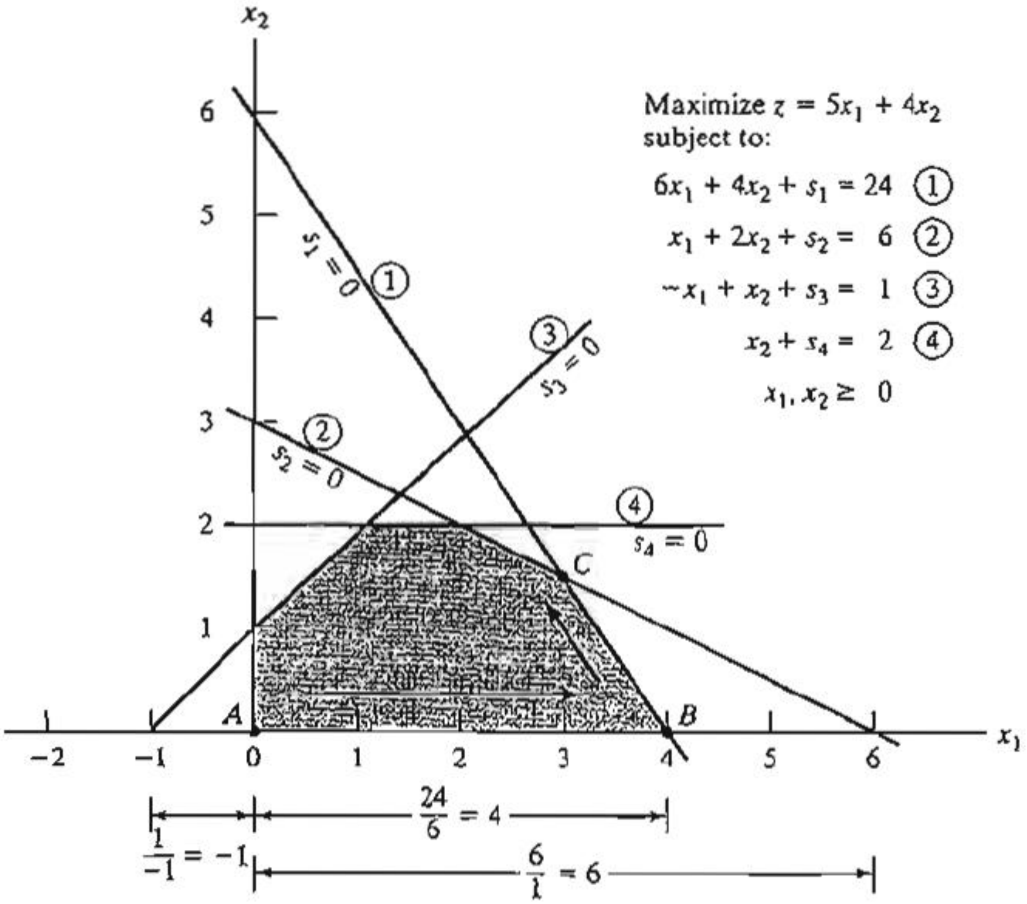


FIGURE 3.5

Graphical interpretation of the simplex method ratios in the Reddy Mikks model

The *minimum nonnegative* ratio automatically identifies the current basic variable s_1 as the leaving variable and assigns the entering variable x_1 the new value of 4.

How do the computed ratios determine the leaving variable and the value of the entering variable? Figure 3.5 shows that the computed ratios are actually the intercepts of the constraints with the entering variable (x_1) axis. We can see that the value of x_1 must be increased to 4 at corner point B, which is the smallest nonnegative intercept with the x_1-axis. An increase beyond B is infeasible. At point B, the current basic variable s_1 associated with constraint 1 assumes a zero value and becomes the *leaving variable*. The rule associated with the ratio computations is referred to as the **feasibility condition** because it guarantees the feasibility of the new solution.

The new solution point B is determined by "swapping" the entering variable x_1 and the leaving variable s_1 in the simplex tableau to produce the following sets of nonbasic and basic variables:

Nonbasic (zero) variables at B: (s_1, x_2)

Basic variables at B: (x_1, s_2, s_3, s_4)

The swapping process is based on the **Gauss-Jordan row operations**. It identifies the entering variable column as the **pivot column** and the leaving variable row as the **pivot row**. The intersection of the pivot column and the pivot row is called the **pivot element**. The following tableau is a restatement of the starting tableau with its pivot row and column highlighted.

	Basic	z	Enter ↓ x_1	x_2	s_1	s_2	s_3	s_4	Solution	
	z	1	−5	−4	0	0	0	0	0	
Leave ←	s_1	0	6	4	1	0	0	0	24	Pivot row
	s_2	0	1	2	0	1	0	0	6	
	s_3	0	−1	1	0	0	1	0	1	
	s_4	0	0	1	0	0	0	1	2	
			Pivot column							

1. *Pivot row*

   a. Replace the leaving variable in the *Basic* column with the entering variable.

   b. New pivot row = Current pivot row ÷ Pivot element

2. *All other rows, including z*

New Row = (Current row) − (Its pivot column coefficient) ×
(New pivot row)

These computations are applied to the preceding tableau in the following manner:

1. Replace s_1 in the *Basic* column with x_1:

New x_1-row = Current s_1-row ÷ 6

$= \frac{1}{6}(0\ 6\ 4\ 1\ 0\ 0\ 0\ 24)$

$= \left(0\ 1\ \frac{2}{3}\ \frac{1}{6}\ 0\ 0\ 0\ 4\right)$

2. New z-row = Current z-row − (-5) × New x_1-row

$= \left(1\ -5\ -4\ 0\ 0\ 0\ 0\ 0\right) - (-5) \times \left(0\ 1\ \frac{2}{3}\ \frac{1}{6}\ 0\ 0\ 0\ 4\right)$

$= \left(1\ 0\ -\frac{2}{3}\ \frac{5}{6}\ 0\ 0\ 0\ 20\right)$

3. New s_2-row = Current s_2-row − (1) × New x_1-row

$= (0\ 1\ 2\ 0\ 1\ 0\ 0\ 6) - (1) \times \left(0\ 1\ \frac{2}{3}\ \frac{1}{6}\ 0\ 0\ 0\ 4\right)$

$= \left(0\ 0\ \frac{4}{3}\ -\frac{1}{6}\ 1\ 0\ 0\ 2\right)$

4. New s_3-row = Current s_3-row − (-1) × New x_1-row

$= (0\ -1\ 1\ 0\ 0\ 1\ 0\ 1) - (-1) \times \left(0\ 1\ \frac{2}{3}\ \frac{1}{6}\ 0\ 0\ 0\ 4\right)$

$= \left(0\ 0\ \frac{5}{3}\ \frac{1}{6}\ 0\ 1\ 0\ 5\right)$

5. New s_4-row = Current s_4-row − (0) × New x_1-row

$= (0\ 0\ 1\ 0\ 0\ 0\ 1\ 2) - (0)\left(0\ 1\ \frac{2}{3}\ \frac{1}{6}\ 0\ 0\ 0\ 4\right)$

$= (0\ 0\ 1\ 0\ 0\ 0\ 1\ 2)$

The new basic solution is (x_1, s_2, s_3, s_4), and the new tableau becomes

				↓					
	Basic	z	x_1	x_2	s_1	s_2	s_3	s_4	Solution
	z	1	0	$-\frac{2}{3}$	$\frac{5}{6}$	0	0	0	20
	x_1	0	1	$\frac{2}{3}$	$\frac{1}{6}$	0	0	0	4
←	s_2	0	0	$\frac{4}{3}$	$-\frac{1}{6}$	1	0	0	2
	s_3	0	0	$\frac{5}{3}$	$\frac{1}{6}$	0	1	0	5
	s_4	0	0	1	0	0	0	1	2

Observe that the new tableau has the same properties as the starting tableau. When we set the new nonbasic variables x_2 and s_1 to zero, the *Solution* column automatically yields the new basic solution ($x_1 = 4, s_2 = 2, s_3 = 5, s_4 = 2$). This "conditioning" of the tableau is the result of the application of the Gauss-Jordan row operations. The corresponding new objective value is $z = 20$, which is consistent with

$$\text{New } z = \text{Old } z + \text{New } x_1\text{-value} \times \text{its objective coefficient}$$
$$= 0 + 4 \times 5 = 20$$

In the last tableau, the *optimality condition* shows that x_2 is the entering variable. The feasibility condition produces the following

	Entering		
Basic	x_2	Solution	Ratio
x_1	$\frac{2}{3}$	4	$x_2 = 4 \div \frac{2}{3} = 6$
s_2	$\frac{4}{3}$	2	$x_2 = 2 \div \frac{4}{3} = 1.5$ (minimum)
s_3	$\frac{5}{3}$	5	$x_2 = 5 \div \frac{5}{3} = 3$
s_4	1	2	$x_2 = 2 \div 1 = 2$

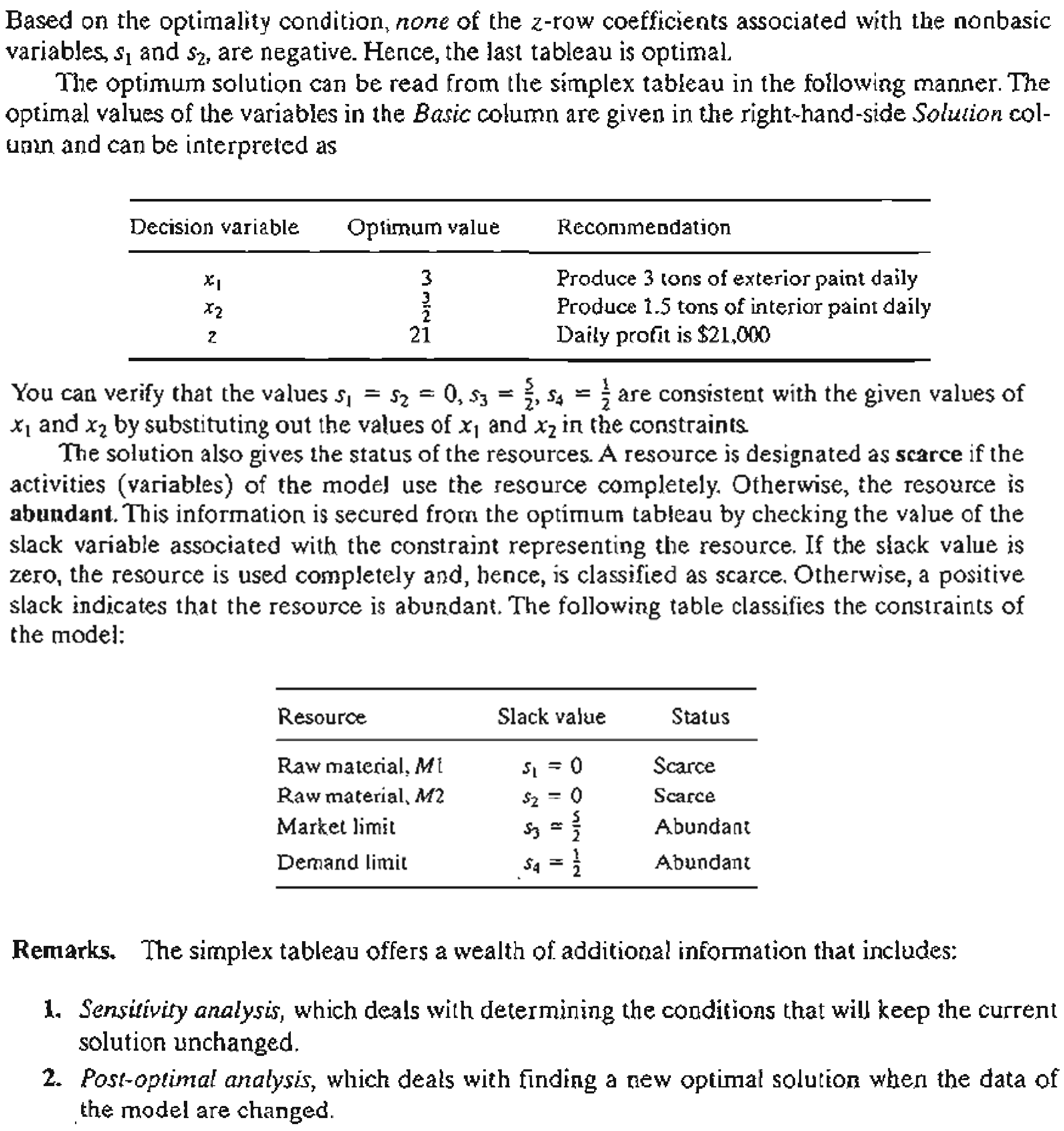
Thus, s_2 leaves the basic solution and new value of x_2 is 1.5. The corresponding increase in z is $\frac{2}{3}x_2 = \frac{2}{3} \times 1.5 = 1$, which yields new $z = 20 + 1 = 21$.

Replacing s_2 in the *Basic* column with entering x_2, the following Gauss-Jordan row operations are applied:

1. New pivot x_2-row = Current s_2-row $\div \frac{4}{3}$
2. New z-row = Current z-row $- \left(-\frac{2}{3}\right) \times$ New x_2-row
3. New x_1-row = Current x_1-row $- \left(\frac{2}{3}\right) \times$ New x_2-row
4. New s_3-row = Current s_3-row $- \left(\frac{5}{3}\right) \times$ New x_2-row
5. New s_4-row = Current s_4-row $- (1) \times$ New x_2-row

These computations produce the following tableau:

Basic	z	x_1	x_2	s_1	s_2	s_3	s_4	Solution
z	1	0	0	$\frac{3}{4}$	$\frac{1}{2}$	0	0	21
x_1	0	1	0	$\frac{1}{4}$	$-\frac{1}{2}$	0	0	3
x_2	0	0	1	$-\frac{1}{8}$	$\frac{3}{4}$	0	0	$\frac{3}{2}$
s_3	0	0	0	$\frac{3}{8}$	$-\frac{5}{4}$	1	0	$\frac{5}{2}$
s_4	0	0	0	$\frac{1}{8}$	$-\frac{3}{4}$	0	1	$\frac{1}{2}$



Based on the optimality condition, *none* of the z-row coefficients associated with the nonbasic variables, s_1 and s_2, are negative. Hence, the last tableau is optimal.

The optimum solution can be read from the simplex tableau in the following manner. The optimal values of the variables in the *Basic* column are given in the right-hand-side *Solution* column and can be interpreted as

Decision variable	Optimum value	Recommendation
x_1	3	Produce 3 tons of exterior paint daily
x_2	$\frac{3}{2}$	Produce 1.5 tons of interior paint daily
z	21	Daily profit is $21,000

You can verify that the values $s_1 = s_2 = 0$, $s_3 = \frac{5}{2}$, $s_4 = \frac{1}{2}$ are consistent with the given values of x_1 and x_2 by substituting out the values of x_1 and x_2 in the constraints.

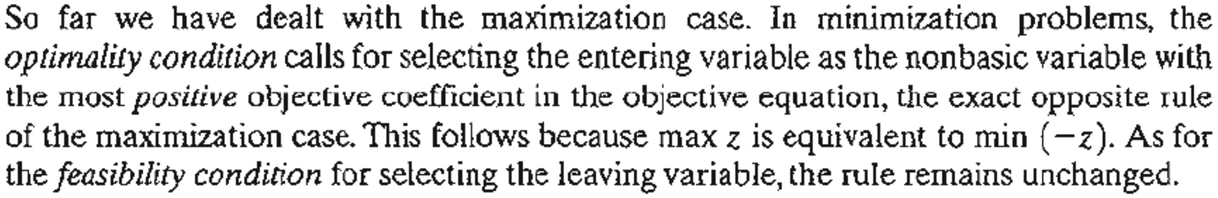
The solution also gives the status of the resources. A resource is designated as **scarce** if the activities (variables) of the model use the resource completely. Otherwise, the resource is **abundant**. This information is secured from the optimum tableau by checking the value of the slack variable associated with the constraint representing the resource. If the slack value is zero, the resource is used completely and, hence, is classified as scarce. Otherwise, a positive slack indicates that the resource is abundant. The following table classifies the constraints of the model:

Resource	Slack value	Status
Raw material, $M1$	$s_1 = 0$	Scarce
Raw material, $M2$	$s_2 = 0$	Scarce
Market limit	$s_3 = \frac{5}{2}$	Abundant
Demand limit	$s_4 = \frac{1}{2}$	Abundant

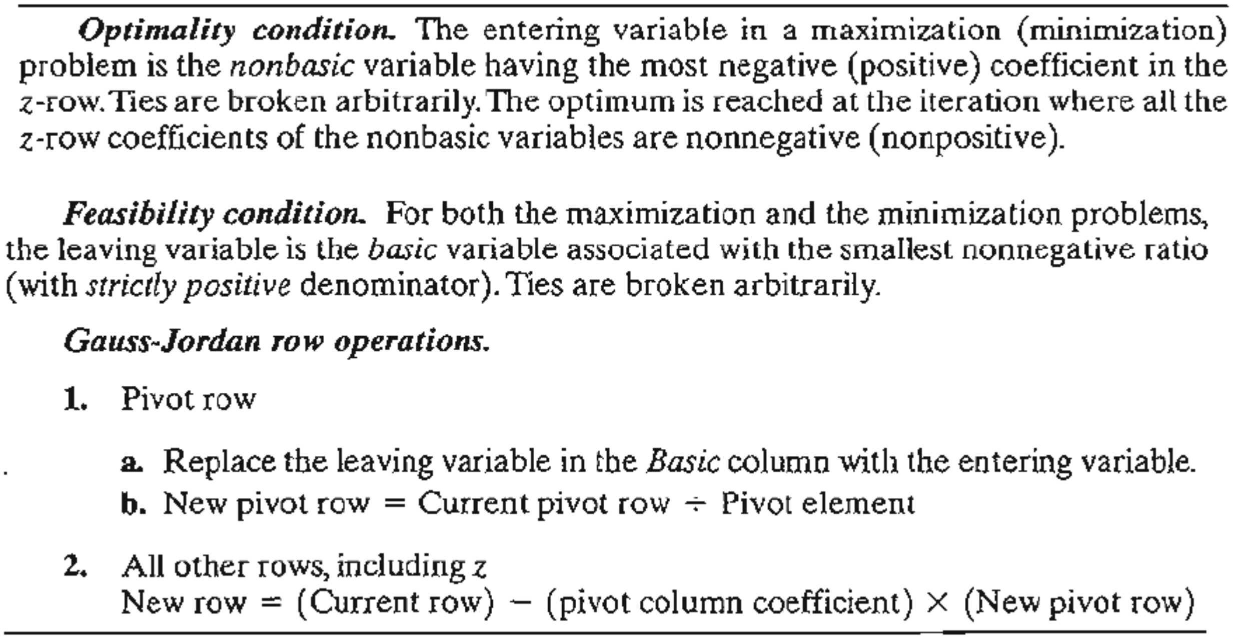
Remarks. The simplex tableau offers a wealth of additional information that includes:

1. *Sensitivity analysis*, which deals with determining the conditions that will keep the current solution unchanged.
2. *Post-optimal analysis*, which deals with finding a new optimal solution when the data of the model are changed.

Summary of the Simplex Method



So far we have dealt with the maximization case. In minimization problems, the *optimality condition* calls for selecting the entering variable as the nonbasic variable with the most *positive* objective coefficient in the objective equation, the exact opposite rule of the maximization case. This follows because max z is equivalent to min $(-z)$. As for the *feasibility condition* for selecting the leaving variable, the rule remains unchanged.



Optimality condition. The entering variable in a maximization (minimization) problem is the *nonbasic* variable having the most negative (positive) coefficient in the z-row. Ties are broken arbitrarily. The optimum is reached at the iteration where all the z-row coefficients of the nonbasic variables are nonnegative (nonpositive).

Feasibility condition. For both the maximization and the minimization problems, the leaving variable is the *basic* variable associated with the smallest nonnegative ratio (with *strictly positive* denominator). Ties are broken arbitrarily.

Gauss-Jordan row operations.

1. Pivot row

a. Replace the leaving variable in the *Basic* column with the entering variable.
b. New pivot row = Current pivot row ÷ Pivot element

2. All other rows, including z
New row = (Current row) − (pivot column coefficient) × (New pivot row)

Apparent difficulties in the Simplex method.

1. Given n decision variables, you can always find a problem instance where the algorithm requires $O(2^n)$ operations and pivots to arrive at a solution.
2. Not so great for large problems, because pivoting operations become expensive.
3. Simplex method Involves understanding of many conceptual technical aspects. These cannot be understood by any manager not conversant with the subject.
4. Graphic solution method has lot of applications and is relatively short and simple. However, it has limitations and cannot be applied to problems with more than two variables in the objective function.
5. Simplex method of LPP can be applied to problems with more than two variables in the objective function, the procedure adopted is complicated and long. It may need computation of 4 to 5 simplex tables and can test the patience of the problem solver. Computers are of course helpful in such cases.

6. Linear programming problems need lot of expertise, time and are cumbersome. A number of steps have to be adopted to proceed in a systematic manner before one can arrive at the solution.

7. LPP does not lead to ‘a unique’ optimal solution. It can provide different types of solutions like feasible solution, infeasible solution, unbounded solution, degenerate solution etc.

8. It gives absurd or impractical results in many solutions.

9. LPP model makes many assumptions in the values of objective function and constraint variables, like the rate of profit. In fact, such assumptions may not be right.

10. The whole approach to the solution is based on the linearity of the functions i.e., all the variables involved in the problem increase or decrease in a linear manner. This assumption does not hold good in all cases. In many cases, the objective function may assume the form of a quadratic equation.

Module IV

4.1 TRANSPORTATION PROBLEM

1. Transportation problems are also linear programming problems and can be solved by simplex method.
2. The transportation model is a special class of linear programs that deals with shipping a commodity from *sources* (e.g., factories) to *destinations* (e.g., warehouses).
3. **The objective is to determine the shipping schedule that minimizes the total shipping cost while satisfying supply and demand limits.**
4. The application of the transportation model can be extended to other areas of operation, including inventory
control, employment scheduling, and personnel assignment.

DEFINITION OF THE TRANSPORTATION MODEL

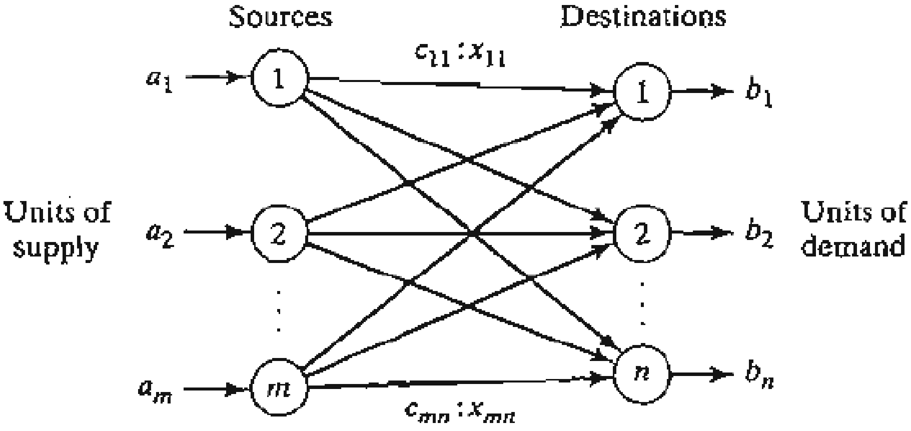


Figure 4.1 Representation of the transportation model with nodes and arcs

The general problem is represented by the network in Figure 4.1. There are *m* sources and *n* destinations, each represented by a node. The arcs represent the routes linking the sources and the destinations. Arc *(i,* j) joining source *i* to destination *j* carries two pieces of information: the transportation cost per unit, c_{ij}, and the amount shipped, x_{ij}. The amount of supply at source i is a_i and the amount of demand at destination j is b_j • The objective of the model is to determine the unknowns x_{ij} that will minimize the total transportation cost while satisfying all the supply and demand restrictions.

Example 4.1
A scooter production company produces scooters at the units situated at various places (called origins) and supplies them to the places where the depot (called destination) are situated.
Here the availability as well as requirements of the various depots are finite and constitute the limited resources.

This type of problem is known as **distribution or transportation problem** in which the key idea is to minimize the cost or the time of transportation.

MATHEMATICAL FORMULATION OF TRANSPORTATION PROBLEM

Let there be three units, producing scooter, say, A_1, A_2 and A_3 from where the scooters are to be supplied to four depots say B_1, B_2, B_3 and B_4.

Let the number of scooters produced at A_1, A_2 and A_3 be a_1, a_2 and a_3 respectively and the demands at the depots be b_2, b_1,
b_3 and b_4 respectively.

Assume the condition

$a_1+a_2+a_3 = b_1+b_2+b_3+b_4$

i.e., all scooters produced are supplied to the different depots. Let the cost of transportation of one scooter from A_1 to B_1 be c_{11}. Similarly, the cost of transportations in other casus are also shown in the Figure 4.2 and Table 1.

Let out of a_1 scooters available at A_1, x_{11} be taken at B_1 depot, x_{12} be taken at B_2 depot and to other depots as well, as
shown in the following figure and table 1.

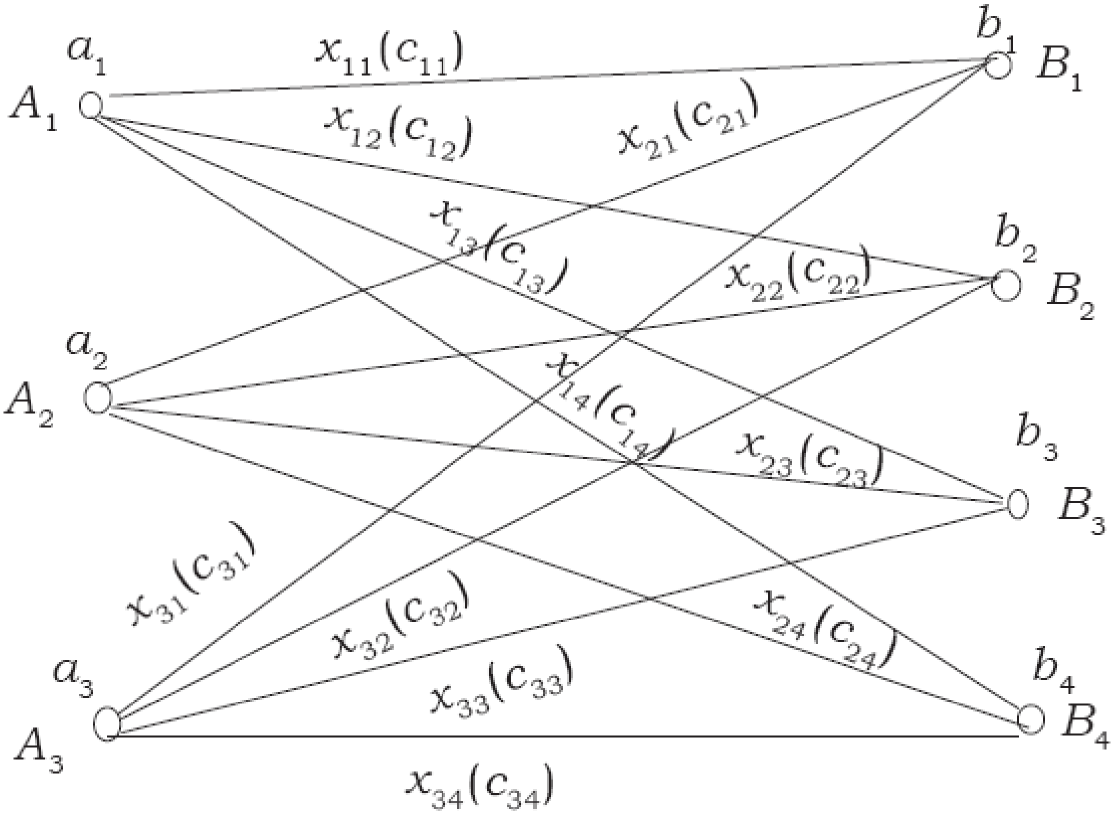


Figure 4.2

Total number of scooters to be transported form A_1 to all destination, i.e., B_1, B_2, B_3, and B_4 must be equal to a_1.

$\therefore \qquad x_{11}+x_{12}+x_{13}+x_{14} = a_1 \qquad (1)$

Similarly, from A_2 and A_3 the scooters transported be equal
to a_2 and a_3 respectively.

$\therefore \qquad x_{21}+x_{22}+x_{23}+x_{24} = a_2 \qquad (2)$

and $\qquad x_{31}+x_{32}+x_{33}+x_{34} = a_3 \qquad (3)$

On the other hand it should be kept in mind that the total number of scooters delivered to B_1 from all units must be equal to b_1, i.e.,

$x_{11}+x_{21}+x_{31} = b_1$ (4)

Similarly, $x_{12}+x_{22}+x_{32} = b_2$ (5)

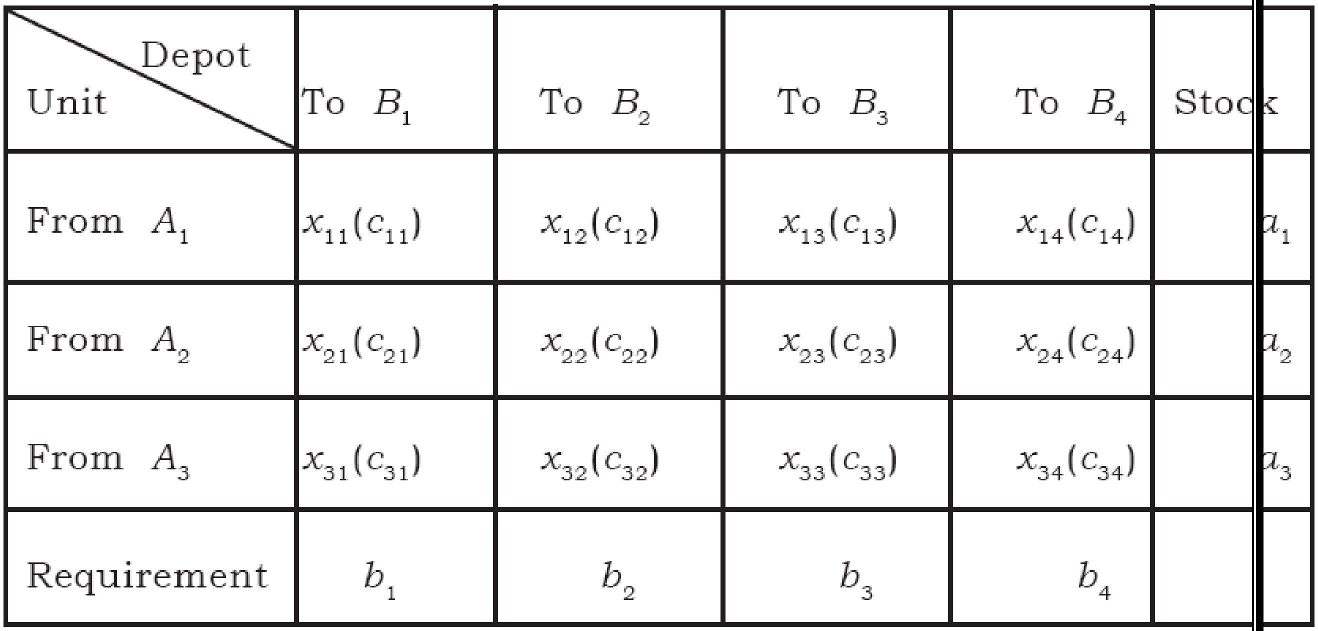
$x_{13}+x_{23}+x_{33} = b_3$ (6)

$x_{14}+x_{24}+x_{34} = b_4$ (7)

With the help of the above information we can construct thefollowing table :

Table 1

$\sum_{i,j}$



Depot Unit	To B_1	To B_2	To B_3	To B_4	Stock
From A_1	$x_{11}(c_{11})$	$x_{12}(c_{12})$	$x_{13}(c_{13})$	$x_{14}(c_{14})$	a_1
From A_2	$x_{21}(c_{21})$	$x_{22}(c_{22})$	$x_{23}(c_{23})$	$x_{24}(c_{24})$	a_2
From A_3	$x_{31}(c_{31})$	$x_{32}(c_{32})$	$x_{33}(c_{33})$	$x_{34}(c_{34})$	a_3
Requirement	b_1	b_2	b_3	b_4	

The cost of transportation from A_i (i=1,2,3) to B_j (j=1,2,3,4) will be equal to

$$S = \sum_{i,j} c_{ij} \, x_{ij} \, , \qquad (8)$$

where the symbol put before $c_{ij} x_{ij}$ signifies that the quantities $c_{ij} x_{ij}$ must be summed over all i = 1,2,3 and all
j = 1,2,3,4.

Thus we come across a linear programming problem given by equations (1) to (7) and a linear function (8).

(A) Feasible Solution (F.S.)

A set of non-negative allocations $x_{ij} \geq 0$ which satisfies the row and column restrictions is known as feasible solution.

(B) Basic Feasible Solution (B.F.S.)

A feasible solution to a m-origin and n-destination problem is said to be basic feasible solution if the number of positive allocations are (m+n–1).

If the number of allocations in a basic feasible solutions are less than (m+n–1), it is called degenerate basic feasible solution (DBFS) (otherwise non-degenerate).

(C) Optimal Solution

A feasible solution (not necessarily basic) is said to be optimal if it minimizes the total transportation cost.

THE STEPS OF THE TRANSPORTATION ALGORITHM

Step 1. Determine a starting basic feasible solution, and go to step 2.

Step 2. Use the optimality condition of the simplex method to determine the entering variable from among all the nonbasic variables. If the optimality condition is satisfied, stop. Otherwise, go to step 3.

Step 3. Use the feasibility condition of the simplex method to determine the leaving variable from among all the current basic variables, and find the new basic solution. Return to step 2.

Step 1. Determine a starting basic feasible solution

There are three different methods to obtain the initial basic feasible solution viz.

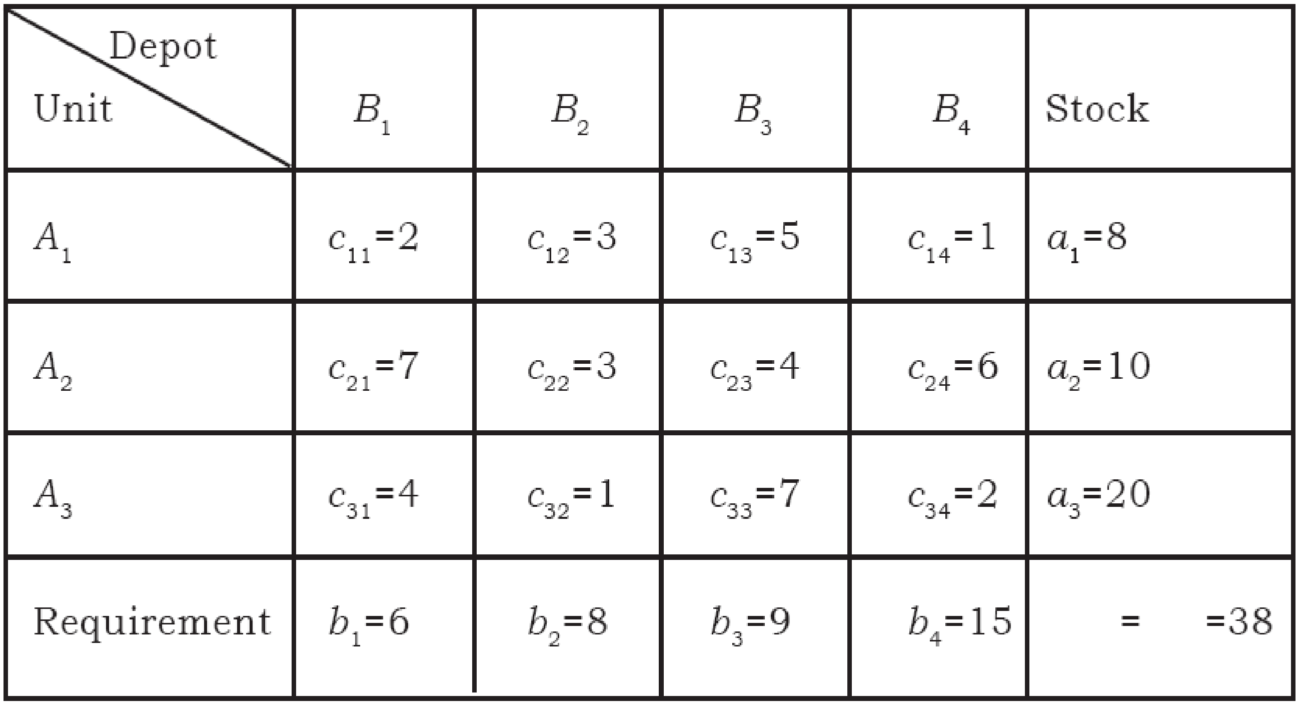
(I) North-West corner rule

(II) Lowest cost entry method

(III) Vogel's approximation method

Example:

Table 2



Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	$c_{11}=2$	$c_{12}=3$	$c_{13}=5$	$c_{14}=1$	$a_1=8$
A_2	$c_{21}=7$	$c_{22}=3$	$c_{23}=4$	$c_{24}=6$	$a_2=10$
A_3	$c_{31}=4$	$c_{32}=1$	$c_{33}=7$	$c_{34}=2$	$a_3=20$
Requirement	$b_1=6$	$b_2=8$	$b_3=9$	$b_4=15$	= =38

(All terms are in hundreds)

(I) North-West corner rule

In this method we distribute the available units in rows and column in such a way that the sum will remain the same. The method starts at the northwest-corner cell (route) of the tableau (variable xu). We have to follow the steps given below.

Step 1. Allocate as much as possible to the selected cell, and adjust the associated amounts of supply and

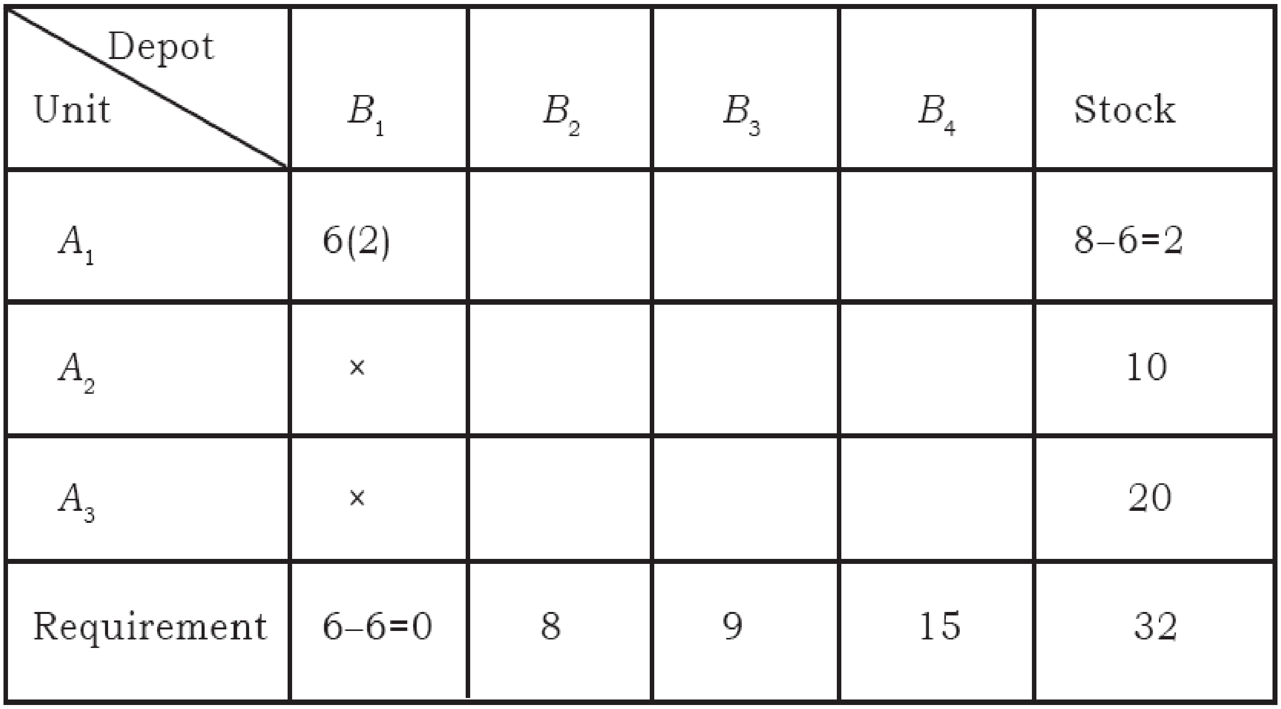
demand by subtracting the allocated amount.

Step 2. Cross out the row or column with zero supply or demand to indicate that no further assignments can be made in that row or column. If both a row and a column net to zero simultaneously, *cross out one only*, and leave a zero supply (demand) in the uncrossed-out TOW (column).

Step 3. If *exactly one* row or column is left uncrossed out, stop. Otherwise, move to the cell to the right if a
column has just been crossed out or below if a row has been crossed out. Go to step 1.

(a) Start allocations from north-west corner, i.e., from (1,1) position. Here min (a_1, b_1), i.e., min (8,6)=6 units. Therefore, the maximum possible units that can be allocated to this position is 6, and write it as 6(2) in the (1,1) position of the table. This completes the allocation in the first column and cross the other positions, i.e., (2,1) and (3,1) in the column. (see Table 3)

Table 3



Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	6(2)				8–6=2
A_2	×				10
A_3	×				20
Requirement	6–6=0	8	9	15	32

(b) After completion of step (a), come across the position (1,2). Here min (8–6,8)=2 units can be allocated to this position and write it as 2(3). This completes the allocations in the first row and cross the other positions, i.e., (1,3) and (1,4) in this row (see Table 4).

Table 4

Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	6(2)	2(3)	×	×	2–2=0
A_2	×				10
A_3	×				20
Requirement	0	8–2=6	9	15	30

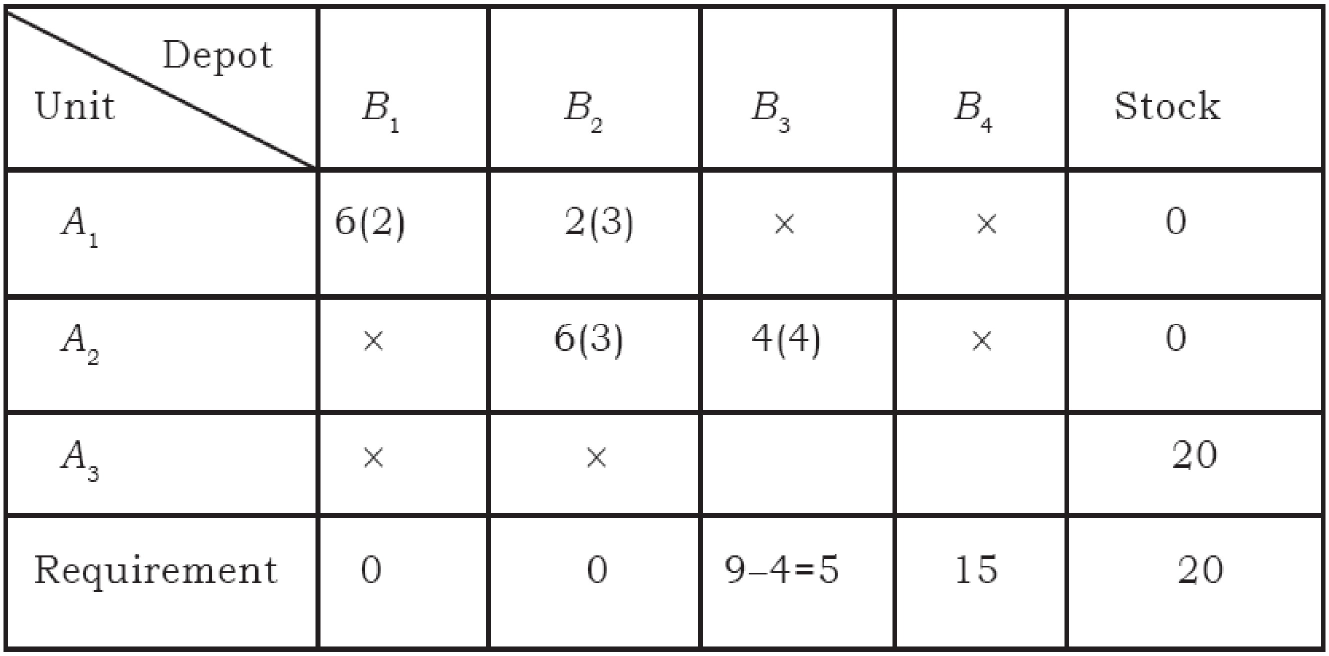
(c) Now come to second row, here the position (2,1) is already been struck off, so consider the position (2,2). Here min (10,8–2)=6 units can be allocated to this position and write it as 6(3). This completes the allocations in second column so strike off the position (3,2) (see Table 5)

Table 5

Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	6(2)	2(3)	×	×	0
A_2	×	6(3)			10–6=4
A_3	×	×			20
Requirement	0	0	9	15	24

(d) Again consider the position (2,3). Here, min (10–6,9)=4 units can be allocated to this position and write it as 4(4). This completes the allocations in second row so struck off the position (2,4) (see Table 6).

Table 6

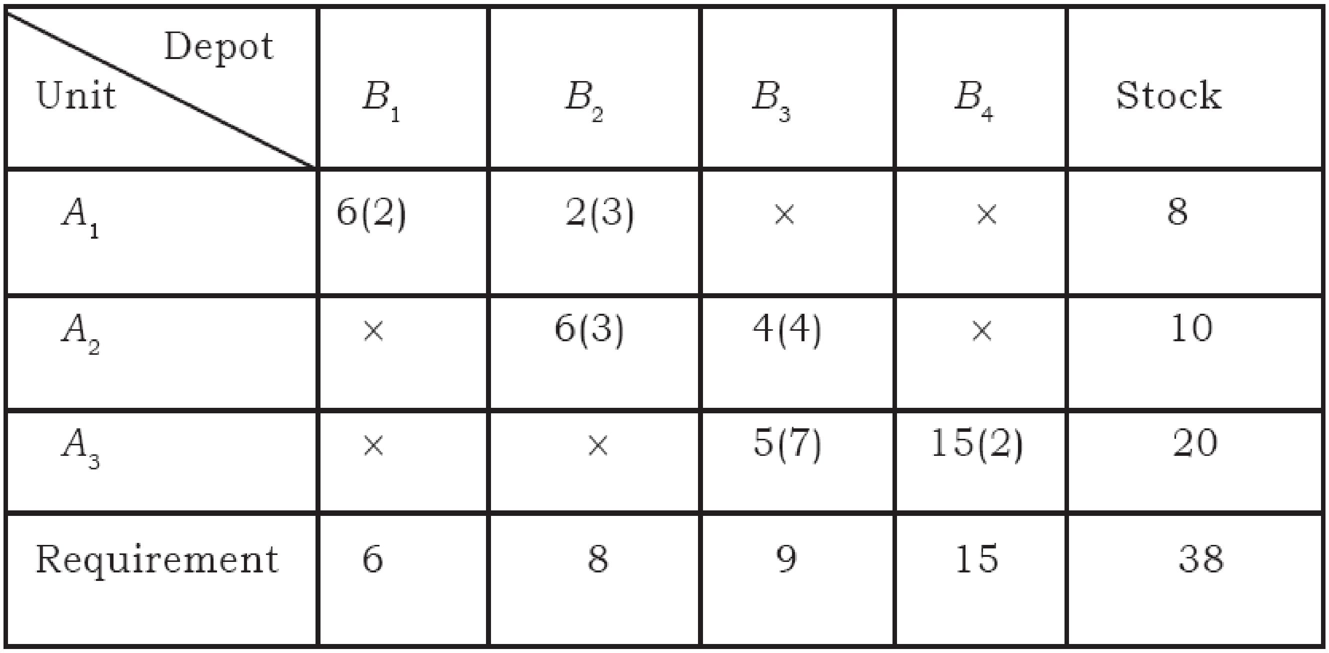


Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	6(2)	2(3)	×	×	0
A_2	×	6(3)	4(4)	×	0
A_3	×	×			20
Requirement	0	0	9–4=5	15	20

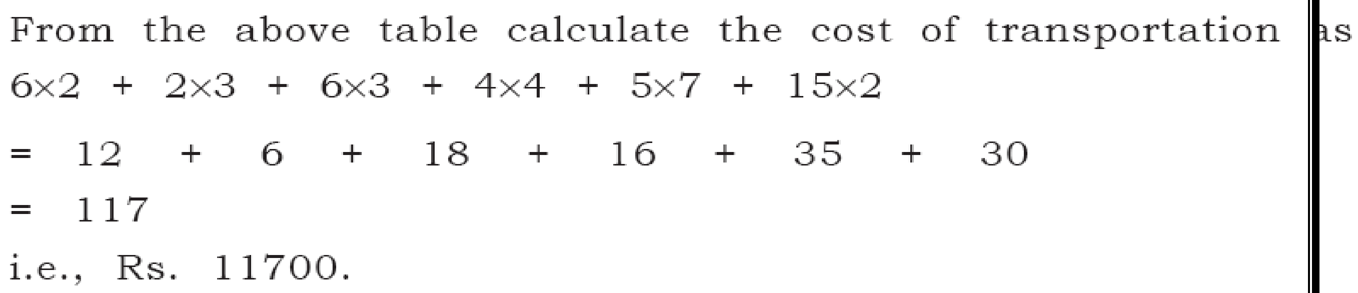
(e) In the third row, positions (3,1) and (3,2) are already been struck off so consider the position (3,3) and allocate it the maximum possible units, i.e., min (20,9–4)=5 units and write it as 5(7). Finally, allocate the remaining units to the position (3,4), i.e., 15 units to this position and write it as 15(2).

Keeping in mind all the allocations done in the above method complete the table as follows:

Table 7



Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	6(2)	2(3)	×	×	8
A_2	×	6(3)	4(4)	×	10
A_3	×	×	5(7)	15(2)	20
Requirement	6	8	9	15	38



From the above table calculate the cost of transportation as
$6\times2 + 2\times3 + 6\times3 + 4\times4 + 5\times7 + 15\times2$
$= 12 + 6 + 18 + 16 + 35 + 30$
$= 117$
i.e., Rs. 11700.

(II) Lowest cost entry method

The least-cost method finds a better starting solution by concentrating on the cheapest routes. The method assigns as much as possible to the cell with the smallest unit cost (ties are broken arbitrarily). Next, the satisfied row or column is crossed out and the amounts of supply and demand are adjusted accordingly.

(a) In this method we start with the lowest cost position. Here it is (1,4) and (3,2) positions, allocate the maximum possible units to these positions, i.e., 8 units to the position (1,4) and 8 units to position (3,2), write them as 8(1) and 8(1) respectively, then strike off the other positions in row 1 and also in column 2, since all the available units are distributed to these positions.

(b) Consider the next higher cost positions, i.e., (1,1) and (3,4) positions, but the position (1,1) is already been struck off so we can't allocate any units to this position. Now allocate the maximum possible units to position (3,4), i.e., 7 units as required by the place and write it as 7(2). Hence the allocations in the column 4 is complete, so strike off the (2,4) position.

(c) Again consider the next higher cost position, i.e., (1,2) and (2,2) positions, but these positions are already been struck off so we cannot allocate any units to these positions.

(d) Consider the next higher positions, i.e., (2,3) and (3,1) positions, allocate the maximum possible units to these positions, i.e., 9 units to position (2,3) and 5 units to position (3,1), write them as 9(4) and 5(4) respectively. In this way allocation in column 3 is complete so strike off the (3,3) position.

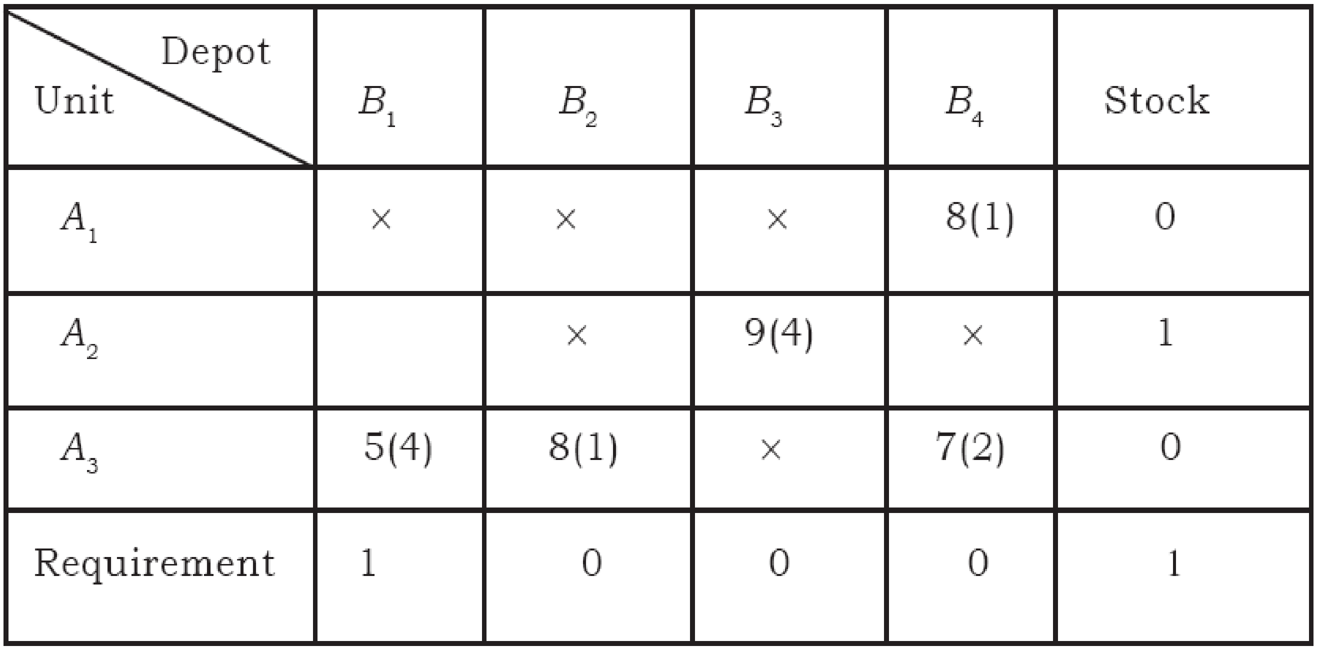
Table 8

Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	×	×	×	8(1)	0
A_2		×			10
A_3		8(1)			12
Requirement	6	0	9	7	22

Table 9

Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	×	×	×	8(1)	0
A_2		×		×	10
A_3		8(1)		7(2)	5
Requirement	6	0	9	0	15

Tavle 10

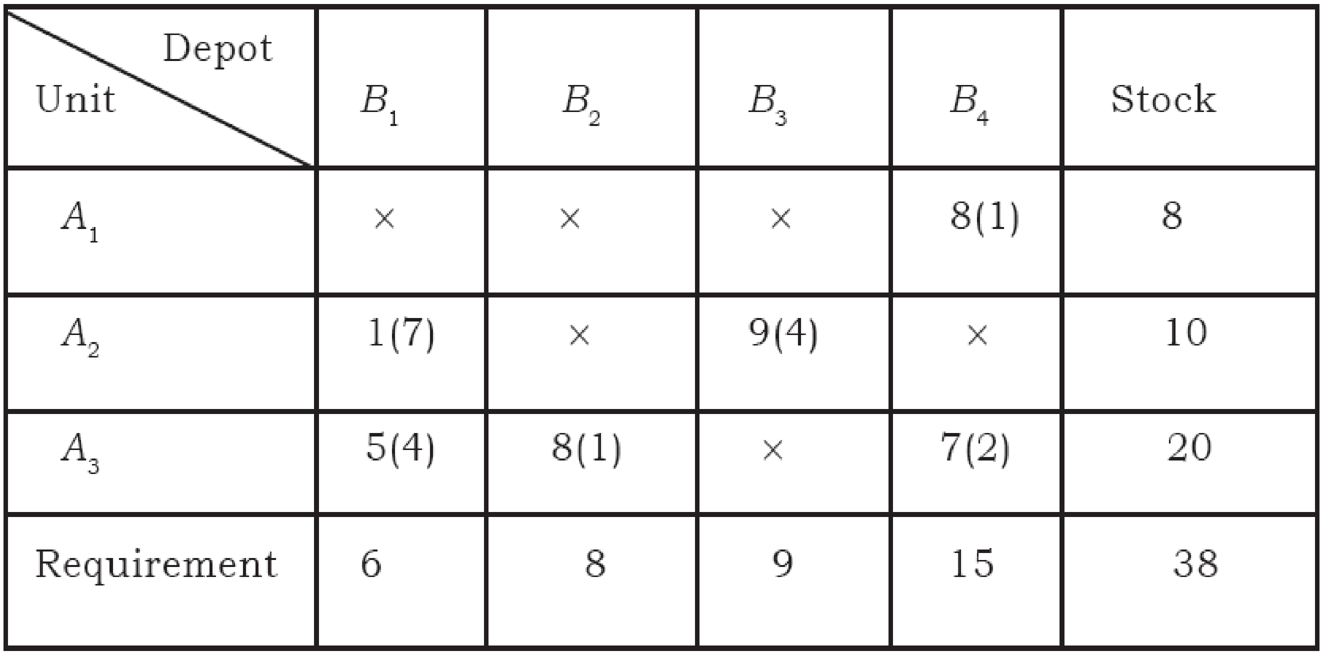


Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	×	×	×	8(1)	0
A_2		×	9(4)	×	1
A_3	5(4)	8(1)	×	7(2)	0
Requirement	1	0	0	0	1

(e) Now only the position (2,1) remains and it automatically takes the alloation 1 to complete the total in this row, therefore, write it as 1(7).

With the help of the above facts complete the allocation table as given below.

Table 11



Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	×	×	×	8(1)	8
A_2	1(7)	×	9(4)	×	10
A_3	5(4)	8(1)	×	7(2)	20
Requirement	6	8	9	15	38

From the above facts, calculate the cost of transportation as

$$8\times1 + 1\times7 + 9\times4 + 5\times4 + 8\times1 + 7\times2$$

= 8 + 7 + 36 + 20 + 8 + 14

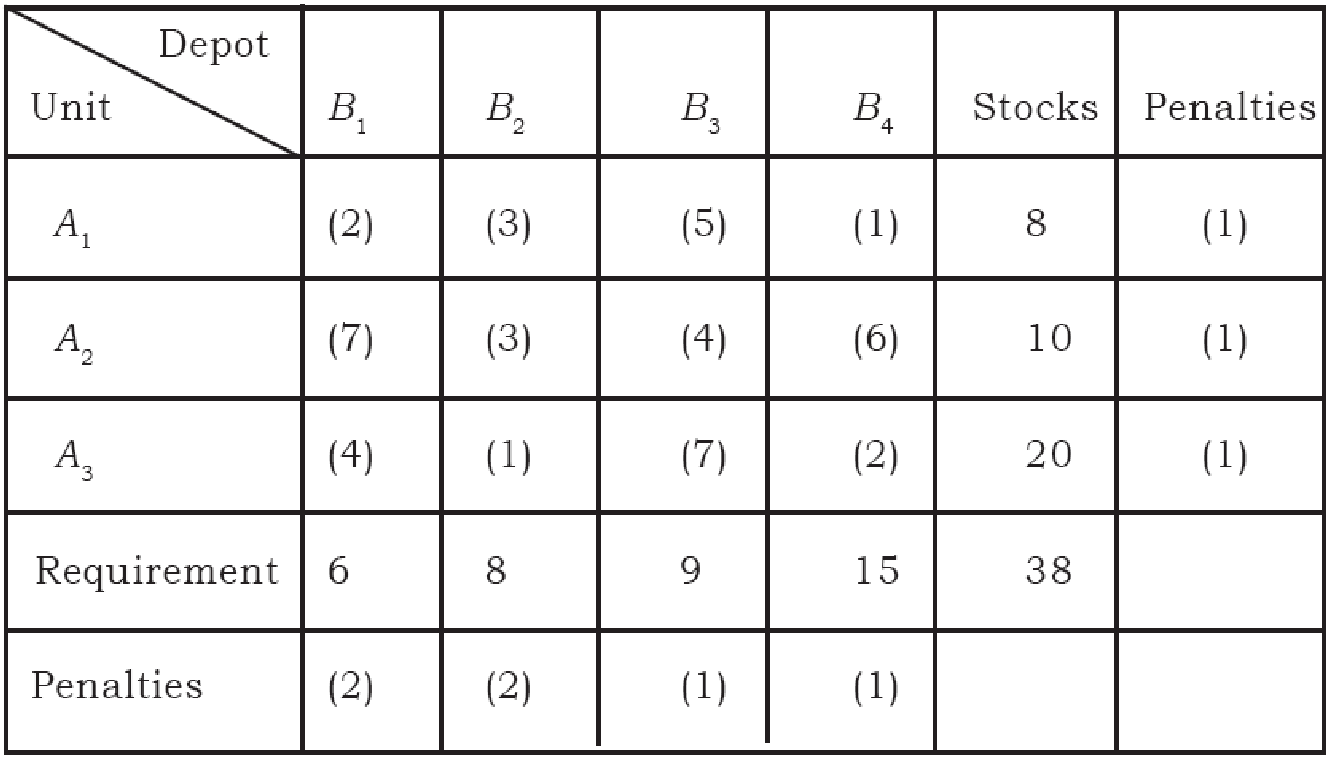
= 93 i.e., Rs. 9300.

(III) Vogel's approximation method
VAM is an improved version of the least-cost method that generally, but not always, produces better starting solutions.

Step 1. For each row (column), determine a penalty measure by subtracting the *smallest* unit cost element in the row
(column) from the *next smallest* unit cost element in the same row (column).
Step 2. Identify the row or column with the largest penalty. Break ties arbitrarily. Allocate as much as possible to the
variable with the least unit cost in the selected row or column. Adjust the supply and demand, and cross out the satisfied row *or* column. If a row and a column are satisfied simultaneously, only one of the two is crossed out, and the remaining row (column) is assigned zero supply (demand).
Step 3. (a) If exactly one row or column with zero supply or demand remains uncrossed out, stop.
(b) If one row (column) with *positive* supply (demand) remains uncrossed out, determine the basic variables
in the row (column) by the least-cost method. Stop.
(c) If all the uncrossed out rows and columns have (remaining) zero supply and demand, determine the *zero*
basic variables by the least-cost method. Stop.
(d) Otherwise, go to step 1.

(a_1) Write the difference of minimum cost and next to minimum cost against each row in the penalty column.
(This difference is known as penalty).
(a_2) Write the difference of minimum cost and next to minimum cost against each column in the penalty row.
(This difference is known as penalty).
We obtain the table as given below.

Table 12

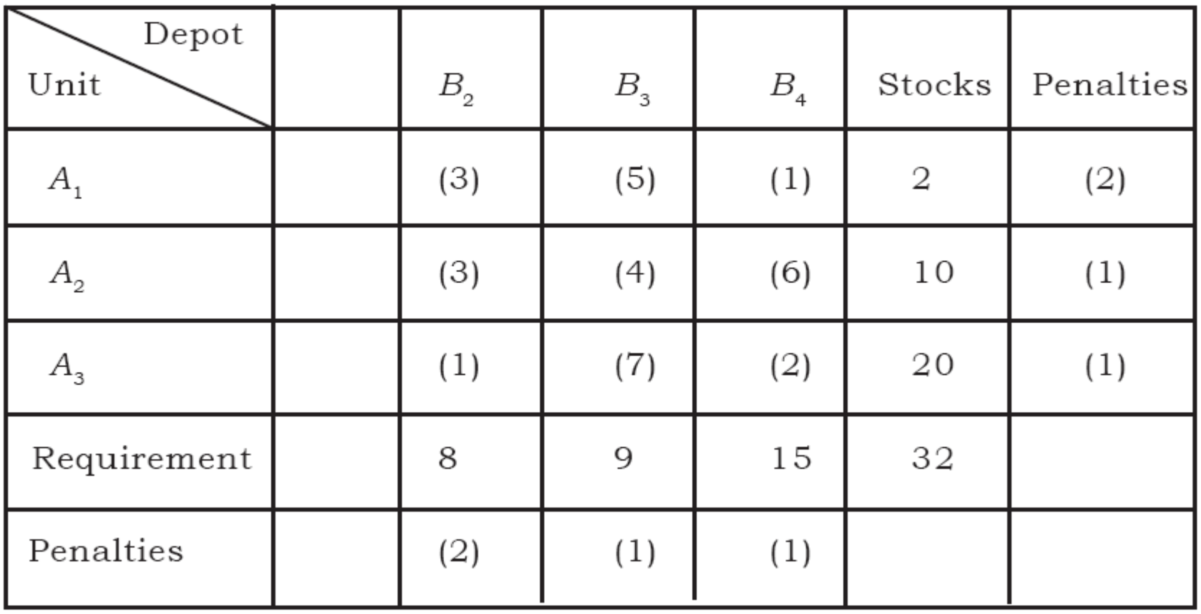


Depot Unit	B_1	B_2	B_3	B_4	Stocks	Penalties
A_1	(2)	(3)	(5)	(1)	8	(1)
A_2	(7)	(3)	(4)	(6)	10	(1)
A_3	(4)	(1)	(7)	(2)	20	(1)
Requirement	6	8	9	15	38	
Penalties	(2)	(2)	(1)	(1)		

↑

(b) Identify the maximum penalties. In this case it is at column one and at column two. Consider any of the two columns, (here take first column) and allocate the maximum units to the place where the cost is minimum (here the position (1,1) has minimum cost so allocate the maximum possible units, i.e., 6 units to this positon). Now write the remaining stock in row one. After removing the first column and then by repeating the step (a), we obtain as follows:

Table 13

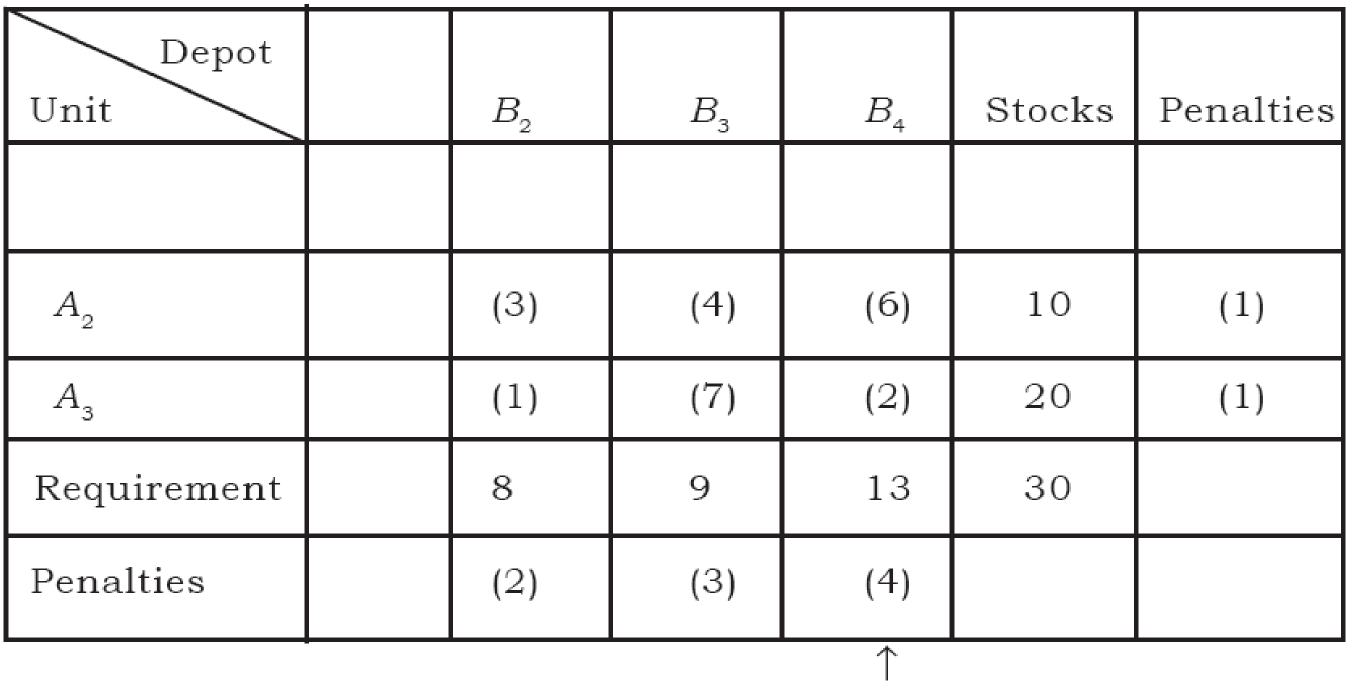


Depot Unit		B_2	B_3	B_4	Stocks	Penalties	
A_1		(3)	(5)	(1)	2	(2)	←
A_2		(3)	(4)	(6)	10	(1)	
A_3		(1)	(7)	(2)	20	(1)	
Requirement		8	9	15	32		
Penalties		(2)	(1)	(1)			

↑

(c) Identify the maximum penalties. In this case it is at row one and at column two. Consider any of the two (let it be first row) and allocate the maximum possible units to the place where the cost is minimum (here the position (1,4) has minimum cost so allocate the maximum possible units, i.e., 2 units to this position). Now write the remaining stock in column four. After removing the first row and by repeating the step(a), we obtain table 14 as given below.

Table 14



Depot / Unit		B_2	B_3	B_4	Stocks	Penalties
A_2		(3)	(4)	(6)	10	(1)
A_3		(1)	(7)	(2)	20	(1)
Requirement		8	9	13	30	
Penalties		(2)	(3)	(4)		

↑

(d) Identify the maximum penalties. In this case it is at column four. Now allocate the maximum possible units to the minimum cost position (here it is at (3,4) position and allocate maximum possible units, i.e., 13 to this positon). Now write the remaining stock in row three. After removing the fourth column and then by repeating the step (a) we obtain table 15 as given below.

Table 15

Depot Unit		B_2	B_3		Stock	Penalties
A_2		(3)	(4)		10	(1)
A_3		(1)	(7)		7	(6)
Requirement		8	9			
Penalties		(2)	(3)			

(e) Identify the maximum penalties. In this case it is at row three. Now allocate the maximum possible units to the minimum cost position (here it is at (3,2) position and allocate maximum possible units, i.e., 7 to this position). Now in orderto complete the sum, (2,2) position will take 1 unit and (2,3) position will be allocated 9 units.

This completes the allocation and with the help of the above informations draw table 16 as under.

Table 16

Depot Unit	B_1	B_2	B_3	B_4	Stock
A_1	6(2)			2(1)	8
A_2		1(3)	9(4)		10
A_3		7(1)		13(2)	20
Requirement	6	8	9	15	38

From the above facts calculate the cost of transportation as

$$6\times2 + 2\times1 + 1\times3 + 9\times4 + 7\times1 + 13\times2$$

= 12 + 2 + 3 + 36 + 7 + 26

= 86

i.e., Rs. 8600.

Iterative Computations of the Transportation Algorithm

After determining the starting solution (using any of the three methods), we use the following algorithm to determine the optimum solution:

Step 1. Use the simplex *optimality condition* to determine the *entering variable* as the current nonbasic variable

that can improve the solution. If the optimality condition is satisfied, stop. Otherwise, go to step 2.

Step 2. Determine the *leaving variable* using the simplex *feasibility condition.* Change the basis, and return to

step 1.

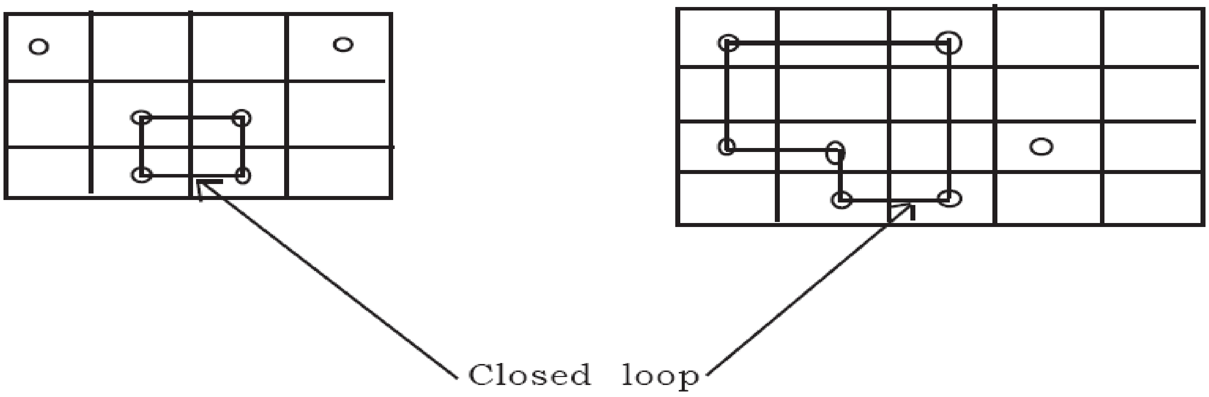
Test for Optimization :

Solutions so obtained may be optimal or may not be optimal, so it becomes essential for us to test for optimization.

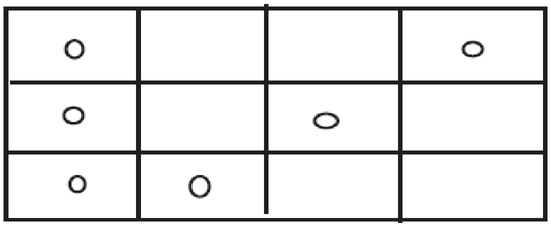
Definition: A basic feasible solution of an (m · n) transportation problem is said to be **non-degenerate** if it has following two properties :

(a) Initial basic feasible solution must contain exactly m+n–1 number of individual allocations.

(b) These allocations must be in independent positions. Independent positions of a set of allocations means that it is always impossible to form any closed loop through these allocations. See fig. given below.



Non - independent positions



Independent positons

Algorithm for optimality test :

In order to test for optimality we should follow the procedure as given below:

Step 1: Start with B.F.S. consisting of m+n–1 allocations in independent positions.

Step 2: Determine a set of m+n numbers u_i (i=1,2,....m) and v_j (j=1,2,...n) such that for each occupied cells (r,s)

$c_{rs} = u_r+v_s$

Step 3: Calculate cell evaluations (unit cost difference) d_j for each empty cell (i,j) by using the formula

$d_{ij} = c_{ij} – (u_i+v_j)$

Step 4: Examine the matrix of cell evaluation d_{ij} for negative entries and conclude that

(i) If all $d_{ij} > 0 \Rightarrow$ Solution is optimal and unique.

(ii) If all $d_{ij} \geq 0 \Rightarrow$ At least one $d_{ij} = 0$
$\Rightarrow$ Solution is optimal and alternate solution also exists.

(iii) If at least one $d_{ij} < 0 \Rightarrow$ Solution is not optimal. If it is so, further improvement is required by
repeating the above process. See step 5 and onwards.

Step 5: (i) See the most negative cell in the matrix [d_{ij}].

(ii) Allocate θ to this empty cell in the final allocation table. Subtract and add the amount of this
allocation to other corners of the loop in order to restore feasibility.

(iii) The value of θ, in general is obtained by equating to zero the minimum of the allocations containing
–θ (not + θ) only at the corners of the closed loop.

(iv)Substitute the value of θ and find a fresh allocation table.

Step 6: Again, apply the above test for optimality till you find all $d_{ij} \geq 0$

Note:

The maximum value of θ is determined based on two conditions.

1. Supply limits and demand requirements remain satisfied.
2. Shipments through all routes remain nonnegative.

These two conditions determine the maximum value of θ and the leaving variable in the following manner. First, construct a ***closed loop*** that starts and ends at the entering variable cell. The loop consists of *connected horizontal* and *vertical* segments only (no diagonals are allowed).
Except for the entering variable cell, each corner of the closed loop must coincide with a basic variable.

Computational demonstration for optimality test

Consider the initial basic feasible solution as obtained by Vogel's approximation method [Table (16)].

Step 1: (i) In this table number of allocations = 3+4–1=6.
(ii) All the positions of allocations are independent.

Step 2: Determine a set of (m+n), i.e., (3+4) numbers u_1, u_2, u_3, and v_1, v_2, v_3, and v_4. for each occupied cells.
For this consider the row or column in which the allocations are maximum (here, let us take first row). Now, take u_1 as an arbitrary constant (say zero) then by using $c_{ij} = u_i+v_j$ try to find all u_i and v_j as

Table 20

	B_1	B_2	B_3	B_4	u_i
A_1	2			1	0
A_2		3	4		3
A_3		1		2	1
v_j	2	0	1	1	

$c_{11} = 2 = u_1+v_1 = 0+v_1 \Rightarrow v_1 = 2$

then $c_{14} = 1 = u_1+v_4 = 0+v_4 \Rightarrow v_4 = 1$

then $c_{34} = 2 = u_3+v_4 = u_3+1 \Rightarrow u_3 = 1$

then $c_{32} = 1 = u_3+v_2 = 1+v_2 \Rightarrow v_2 = 0$

then $c_{22} = 3 = u_2+v_2 = u_2+0 \Rightarrow u_2 = 3$

then $c_{23} = 4 = u_2+v_3 = 3+v_3 \Rightarrow v_3 = 1$

Thus u_1 =0, u_2=3, u_3=1 and v_1 =2, v_2=0, v_3=1 and v_4=1.

Step 3: Cost matrix for the empty positons

i.e., $[\ c_{ij}\ ]$ = $$\begin{bmatrix} \bullet & 3 & 5 & \bullet \\ 7 & \bullet & \bullet & 6 \\ 4 & \bullet & 7 & \bullet \end{bmatrix}$$

Matrix [u_i + v_j] for empty positions

$$\text{i.e., } [\ u_i + v_j\ ] = \begin{bmatrix} \bullet & 0 & 1 & \bullet \\ 5 & \bullet & \bullet & 4 \\ 3 & \bullet & 2 & \bullet \end{bmatrix}$$

$$\therefore\ d_{ij} = [c_{ij}] - [u_i + v_j] = \begin{bmatrix} \bullet & 3 & 4 & \bullet \\ 2 & \bullet & \bullet & 2 \\ 1 & \bullet & 5 & \bullet \end{bmatrix}$$

Step 4: Here all $d_{ij} > 0$ $\Rightarrow$ Solution obtained by Vogel's approximation method is an optimal solution

Example 2

For the transportation problem

Table 21

Warehouse → Factory	W_1	W_2	W_3	W_4	Factory Capacity
F_1	19	30	50	10	7
F_2	70	30	40	60	9
F_3	40	8	70	20	18
Warehouse Requirement	5	8	7	14	34

the initial basic feasible solution obtained by Vogel’s approximation method is given below.

Table 22

	W_1	W_2	W_3	W_4	Available
F_1	5(19)			2(10)	7
F_2			7(40)	2(60)	9
F_3		8(8)		10(20)	18
Requirement	5	8	7	14	

Test this for optimality.

Solution :

Step 1: Number of allocations = 3+4–1=6 and they are in independent positon.

Step 2:

					u_i
	(19)			(10)	10
			(40)	(60)	60
		(8)		(20)	20
v_j	9	−12	−20	0	

Step 3:

$[c_{ij}]$ =

	30	50	
70	30		
40		70	

$$[u_i+v_j]=\begin{array}{c|cccc|c} & & & & & u_i \\ \hline & & -2 & -10 & & 10 \\ & 69 & 48 & & & 60 \\ & 29 & & 0 & & 20 \\ \hline v_j & 9 & -12 & -20 & 0 & \end{array}$$

$$d_{ij}=[c_{ij}]-[u_i+v_j]=\begin{array}{|c|c|c|c|} \hline & 32 & 60 & \\ \hline 1 & -18 & & \\ \hline 11 & & 70 & \\ \hline \end{array}$$

Step 4: Since one $d_{ij} < 0$, therefore, the solution is not optimal. See step 5 and onwards for to find new allocations and test of optimality

Step 5: (ii)

					Available
	5(19)			2(10)	7
		+θ	7(40)	2(60) −θ	9
		8(8) −θ		10(20) +θ	18
Required	5	8	7	14	

(iii) $\min [\,8 - \theta,\ 2 - \theta\,] = 0$

i.e., $2 - \theta = 0$

i.e., $\theta = 2$

∴

					Available
	5(19)			2(10)	7
		2(30)	7(40)		9
		6(8)		12(20)	18
Required	5	8	7	14	

This improved basic feasible solution gives the cost for this solution as 5(19)+2(10)+2(30)+7(40)+6(8)+12(20) = Rs.743.

Step 6: Test this improved solution for optimality by repeating steps 1,2,3 and 4. In each step, following matrices are obtained:

Matrix $[c_{ij}]$ for empty cells

•	(30)	(50)	•
70	•	•	(60)
(40)	•	(70)	•

Matrix for u_i and v_j

					u_i
	•(19)			•(10)	−10
		•(30)	•(40)		22
		•(8)		•(20)	0
v_j	29	8	18	20	

Matrix $[u_i+v_j\,]$ for empty cells

•	−2	8	•
51	•	•	42
29	•	18	•

Matrix for $d_{ij}=[c_{ij}]-[u_i+v_j\,]$ for empty cells

•	32	42	•
19	•	•	18
11	•	52	•

Since all $d_{ij} > 0$, therefore, the solution given as improved basic feasible solution is an optimal solution with minimum cost = Rs.743.

ASSIGNMENT PROBLEM

Suppose we have n resources to which we want to assign to n tasks on a one-to-one basis. Suppose also that we know the cost of assigning a given resource to a given task. We wish to find an optimal assignment–one which minimizes total cost.

"The best person for the job" is an apt description of the assignment model. The situation can be illustrated by the assignment of workers with varying degrees of skill to jobs. A job that happens to match a worker's skill costs less than one in which the operator is not as skillful. The objective of the model is to determine the minimum-cost assignment of workers to jobs.

The **assignment problem** refers to the class of linear programming problems that involve determining the most efficient assignment of
1. people to projects
2. salespeople to territories
3. contracts to bidders
4.jobs to machines, etc.

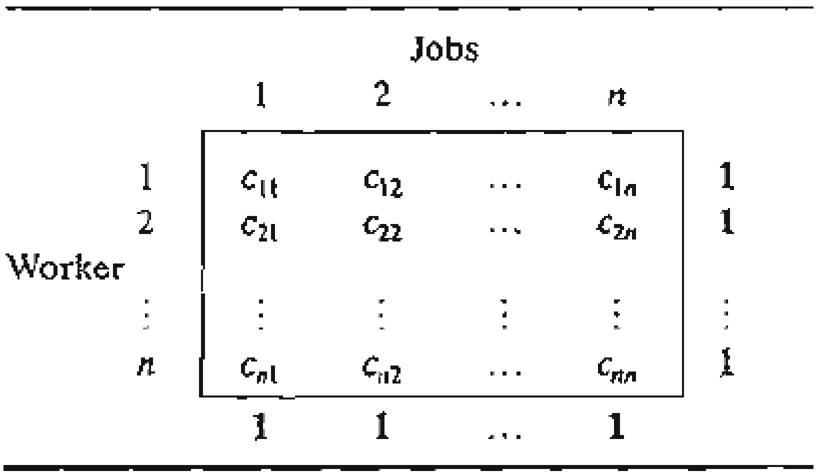
Each assignment problem has associated with it a table, or matrix. Generally, the rows contain the objects or people we wish to assign, and the columns comprise the tasks or things we want them assigned to. The numbers in the table are the costs associated with each particular assignment.

An assignment problem can be viewed as a transportation problem in which
1. the capacity from each source (or person to be assigned) is 1 and
2. the demand at each destination (or job to be done) is 1.

Mathematical form of assignment problem

The general assignment model with *n* workers and *n* jobs is represented in Table 23

Table 23



		Jobs				
		1	2	…	n	
	1	c_{11}	c_{12}	…	c_{1n}	1
	2	c_{21}	c_{22}	…	c_{2n}	1
Worker						
	⋮	⋮	⋮	⋮	⋮	⋮
	n	c_{n1}	c_{n2}	…	c_{nn}	1
		1	1	…	1	

The element *Cij* represents the cost of assigning worker *i* to job *j (i, j* = 1, 2, ... , *n)*. There is no loss of generality in assuming that the number of workers always equals the number of jobs, because we can always add fictitious workers or fictitious jobs to satisfy this assumption.

The assignment model is actually a special case of the transportation model in which the workers represent the sources, and the jobs represent the destinations. The supply (demand) amount at each source (destination) exactly equals 1. The cost of "transporting" worker *i* to job *j* is *Cij'* In effect, the assignment model can be solved directly as a regular transportation model. Nevertheless, the fact that all the supply and demand amounts equal 1 has led to the development of a simple solution algorithm called the Hungarian method. Although the new solution method appears totally unrelated to the transportation model, the algorithm is actually rooted in the simplex method, just as the transportation model is.

The Mathematical Model:

Let ci,j be the cost of assigning the ith resource to the jth task. We define the cost matrix to be the n × n matrix

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \vdots & \vdots & & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,n} \end{bmatrix}.$$

An assignment is a set of n entry positions in the cost matrix, no two of which lie in the same row or column.

The sum of the n entries of an assignment is its cost. An assignment with the smallest possible cost is called an
optimal assignment.

The Hungarian Method

The Hungarian method is an algorithm which finds an optimal assignment for a given cost matrix.

The Hungarian method of assignment provides us with an efficient means of finding the optimal solution without having to make a direct comparison of every assignment option. It operates on a principle of matrix reduction, which means that by subtracting and adding appropriate numbers in the cost table or matrix, we can
reduce the problem to a matrix of opportunity costs. Opportunity costs show the relative penalties associated with assigning any person to a project as opposed to making the best or least-cost assignment. We would like to make assignments such that the opportunity cost for each assignment is zero.

Theorem: If a number is added to or subtracted from all of the entries of any one row or column of a cost matrix, then on optimal assignment for the resulting cost matrix is also an optimal assignment for the original cost matrix.

The Hungarian Method: The following algorithm applies the above theorem to a given n × n cost matrix to find an optimal assignment.

Step 1. Subtract the smallest entry in each row from all the entries of its row.

Step 2. Subtract the smallest entry in each column from all the entries of its column.

Step 3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.

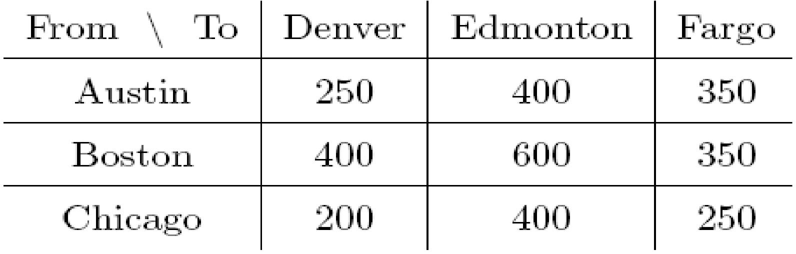
Step 4. Test for Optimality:

(i) If the minimum number of covering lines is n, an optimal assignment of zeros is possible and we are finished.

(ii) If the minimum number of covering lines is less than n, an optimal assignment of zeros is not yet possible. In that case, proceed to Step 5.

Step 5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to Step 3.

Example 1: You work as a sales manager for a toy manufacturer, and you currently have three salespeople on the road meeting buyers. Your salespeople are in Austin, TX; Boston, MA; and Chicago, IL. You want them to fly to three other cities: Denver, CO; Edmonton, Alberta; and Fargo, ND. The table below shows the cost of airplane tickets in dollars between these cities.



From \ To	Denver	Edmonton	Fargo
Austin	250	400	350
Boston	400	600	350
Chicago	200	400	250

Where should you send each of your salespeople in order to minimize airfare?

Step 1. Subtract 250 from Row 1, 350 from Row 2, and 200 from Row 3.

$$\begin{bmatrix} 250 & 400 & 350 \\ 400 & 600 & 350 \\ 200 & 400 & 250 \end{bmatrix} \sim \begin{bmatrix} 0 & 150 & 100 \\ 50 & 250 & 0 \\ 0 & 200 & 50 \end{bmatrix}$$

Step 2. Subtract 0 from Column 1, 150 from Column 2, and 0 from Column 3.

$$\begin{bmatrix} 0 & 150 & 100 \\ 50 & 250 & 0 \\ 0 & 200 & 50 \end{bmatrix} \sim \begin{bmatrix} 0 & 0 & 100 \\ 50 & 100 & 0 \\ 0 & 50 & 50 \end{bmatrix}$$

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} \enclose{horizontalstrike,verticalstrike}{0} & \enclose{horizontalstrike}{0} & \enclose{horizontalstrike}{100} \\ \enclose{horizontalstrike,verticalstrike}{50} & \enclose{horizontalstrike}{100} & \enclose{horizontalstrike}{0} \\ \enclose{verticalstrike}{0} & 50 & 50 \end{bmatrix}$$

Step 4. Since the minimal number of lines is 3, an optimal assignment of zeros is possible and we are finished.

Since the total cost for this assignment is 0, it must be an optimal assignment.

$$\begin{bmatrix} 0 & \boxed{0} & 100 \\ 50 & 100 & \boxed{0} \\ \boxed{0} & 50 & 50 \end{bmatrix}$$

Here is the same assignment made to the original cost matrix.

$$\begin{bmatrix} 250 & \boxed{400} & 350 \\ 400 & 600 & \boxed{350} \\ \boxed{200} & 400 & 250 \end{bmatrix}$$

Example 2: A construction company has four large bulldozers located at four different garages. The bulldozers are to be moved to four different construction sites. The distances in miles between the bulldozers and the construction sites are given below.

Bulldozer \ Site	A	B	C	D
1	90	75	75	80
2	35	85	55	65
3	125	95	90	105
4	45	110	95	115

How should the bulldozers be moved to the construction sites in order to minimize the total distance traveled?

Step 1. Subtract 75 from Row 1, 35 from Row 2, 90 from Row 3, and 45 from Row 4.

$$\begin{bmatrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{bmatrix} \sim \begin{bmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix}$$

Step 2. Subtract 0 from Column 1, 0 from Colum 2, 0 from Column 3, and 5 from Column 4.

$$\begin{bmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix} \sim \begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix}$$

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix}$$

Step 4. Since the minimal number of lines is less than 4, we have to proceed to Step 5.

Step 5. Note that 5 is the smallest entry not covered by any line. Subtract 5 from each uncovered row.

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix} \sim \begin{bmatrix} 15 & 0 & 0 & 0 \\ -5 & 45 & 15 & 20 \\ 30 & 0 & -5 & 5 \\ -5 & 60 & 45 & 60 \end{bmatrix}$$

Now add 5 to each covered column.

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ -5 & 45 & 15 & 20 \\ 30 & 0 & -5 & 5 \\ -5 & 60 & 45 & 60 \end{bmatrix} \sim \begin{bmatrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix}$$

Now return to Step 3.

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix}$$

Step 4. Since the minimal number of lines is less than 4, we have to return to Step 5.

Step 5. Note that 20 is the smallest entry not covered by a line. Subtract 20 from each uncovered row.

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix} \sim \begin{bmatrix} 20 & 0 & 5 & 0 \\ -20 & 25 & 0 & 0 \\ 35 & 0 & 0 & 5 \\ -20 & 40 & 30 & 40 \end{bmatrix}$$

Then add 20 to each covered column.

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ -20 & 25 & 0 & 0 \\ 35 & 0 & 0 & 5 \\ -20 & 40 & 30 & 40 \end{bmatrix} \sim \begin{bmatrix} 40 & 0 & 5 & 0 \\ 0 & 25 & 0 & 0 \\ 55 & 0 & 0 & 5 \\ 0 & 40 & 30 & 40 \end{bmatrix}$$

Now return to Step 3.

Step 3. Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} \hline 40 & 0 & 5 & 0 \\ \hline 0 & 25 & 0 & 0 \\ \hline 55 & 0 & 0 & 5 \\ \hline 0 & 40 & 30 & 40 \end{bmatrix}$$

Step 4. Since the minimal number of lines is 4, an optimal assignment of zeros is possible and we are finished.

$$\begin{bmatrix} 40 & 0 & 5 & \boxed{0} \\ 0 & 25 & \boxed{0} & 0 \\ 55 & \boxed{0} & 0 & 5 \\ \boxed{0} & 40 & 30 & 40 \end{bmatrix}$$

Since the total cost for this assignment is 0, it must be an optimal assignment.

Here is the same assignment made to the original cost matrix.

$$\begin{bmatrix} 90 & 75 & 75 & \boxed{80} \\ 35 & 85 & \boxed{55} & 65 \\ 125 & \boxed{95} & 90 & 105 \\ \boxed{45} & 110 & 95 & 115 \end{bmatrix}$$

So we should send Bulldozer 1 to Site D, Bulldozer 2 to Site C, Bulldozer 3 to Site B, and Bulldozer 4 to Site A.

Module V

Network Analysis By Linear Programming And Shortest Route, Maximal Flow Problem.

5.1 Network Models

The network models in this chapter include the traditional applications of finding the most efficient way to link a number of locations directly or indirectly, finding the shortest route between two cities, determining the maximum flow in a pipeline network, determining the minimum-cost flow in a network that satisfies supply and demand requirements at different locations, and scheduling the activities of a project.

The minimum-cost capacitated algorithm is a generalized network that subsumes the shortest-route and the maximal-flow models

5.1.1 SCOPE NETWORK MODELS

A multitude of operations research situations can be modeled and solved as networks (nodes connected by branches):

1. Design of an offshore natural-gas pipeline network connecting well heads in the Gulf of Mexico to an inshore delivery point. The objective of the model is to minimize the cost of constructing the pipeline.
2. Determination of the shortest route between two cities in an existing network of roads.
3. Determination of the maximum capacity (in tons per year) of a coal slurry pipeline network joining coal mines in Wyoming with power plants in Houston. (Slurry pipelines transport coal by pumping water through specially designed pipes.)
4. Determination of the time schedule (start and completion dates) for the activities of a construction project.
5. Determination of the minimum-cost flow schedule from oil fields to refineries through a pipeline network.

5.1.2 Network Definitions.

A network consists of a set of nodes linked by arcs (or branches). The notation for describing a network is *(N, A),* where *N* is the set of nodes and *A* is the set of arcs. As an illustration, the network in Figure 5.1 is described as

$$N = \{1, 2, 3, 4, 5\}$$

$$A = \{(1,2), (1,3), (2,3), (2,5), (3,4), (3,5), (4,2), (4,5)\}$$

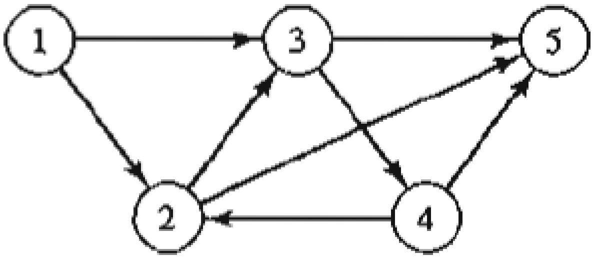


Figure 5.1 Example of *(N,A)* Network

Associated with each network is a flow (e.g., oil products flow in a pipeline and automobile traffic flows in highways). In general, the flow in a network is limited by the capacity of its arcs, which may be finite or infinite.

An arc is said to be **directed** or **oriented** if it allows positive flow in one direction and zero flow in the opposite direction. A **directed network** has all directed arcs.

A **path** is a sequence of distinct arcs that join two nodes through other nodes regardless of the direction of flow in each arc. A path forms a **cycle** or a **loop** if it connects a node to itself through other nodes. For example, in Figure 5.1, the arcs (2,3), (3, 4), and (4,2) form a cycle.

A **connected network** is such that every two distinct nodes are linked by at least one path. The network in Figure 5.1 demonstrates this type of network.

5.2 SHORTEST-ROUTE PROBLEM

The shortest-route problem determines the shortest route between a source and destination in a transportation network.

5.2.1 Shortest-Route Algorithms

Two algorithms for solving both cyclic (i.e., containing loops) and acyclic networks:

1. Dijkstra's algorithm
2. Floyd's algorithm

Dijkstra's algorithm is designed to determine the shortest routes between the source node and every other node in the network. Floyd's algorithm is more general because it allows the determination of the shortest route between *any* two nodes in the network.

5.2.1.1 Dijkstra's algorithm

Let U_i be the shortest distance from source node 1 to node *i,* and define d_{ij} (≥O) as the length of arc *(i, j).* Then the algorithm defines the label for an immediately succeeding node *j* as

$$\mathbf{[U_j, i] = [U_i + d_{ij}, i], d_{ij} \geq 0}$$

The label for the starting node is (0, -], indicating that the node has no predecessor.

Node labels in Dijkstra's algorithm are of two types: *temporary* and *permanent.* A temporary label is modified if a shorter route to a node can be found. If no better route can be found, the status of the temporary label is changed to permanent.

Step o. Label the source node (node 1) with the *permanent* label [0, -]. Set *i* = l.

Step i. (a) Compute the temporary labels [U_i + d_{ij} , i] for each node j that can be reached from node i, provided j

is not permanently labeled. If node j is already labeled with [$U_{j,}$ k] through another node k and if

Ui + dij < Uj, replace [Uj, k] with [U_i + d_{ij} , i].

(b) If all the nodes have permanent labels, stop. Otherwise, select the label [$u_{r,}$ s] having the shortest

distance (= u_r) among all the temporary labels (break ties arbitrarily). Set i = r and repeat step i.

Example:

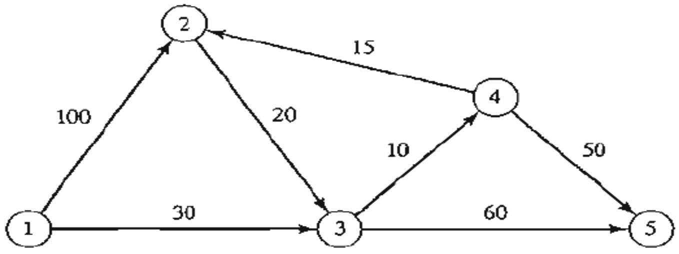


Figure 5.2

The network in Figure 5.2 gives the permissible routes and their lengths in miles between city 1 (node 1) and four other cities (nodes 2 to 5). Determine the shortest routes between city 1 and each of the remaining four cities.

Iteration O: Assign the permanent label [0, -] to node 1.

Iteration 1. Nodes 2 and 3 can be reached from (the last permanently labeled) node 1. Thus, the list of labeled nodes (temporary and permanent) becomes

Node	Label	Status
1	[0,-]	Permanent
2	[0 + 100,1] = [100,1]	Temporary
3	[0+30,1]=[30,1]	Temporary

For the two temporary labels [100, 1] and [30, 1], node 3 yields the smaller distance (u3 = 30). Thus, the status of node 3 is changed to permanent.

Iteration 2. Nodes 4 and 5 can be reached from node 3, and the list of labeled nodes becomes

Node	Label	Status
1	[0, -]	Permanent
2	[lOO, 1]	Temporary
3	[30,1]	Permanent
4	[30+10,3]=[40,3]	Temporary
5	[30 + 60,3J = [90,3]	Temporary

The status of the temporary label [40,3] at node 4 is changed to permanent (U4 = 40).

Iteration 3. Nodes 2 and 5 can be reached from node 4. Thus, the list of labeled nodes is updated as

Node	Label	Status
1	[O, -]	Permanent
2	[40+15, 4]=[55,4]	Temporary
3	[30, 1]	Permanent
4	[40,3]	Permanent
5	[90,3] or [40 + 50,4J = [90,4]	Temporary

Node 2's temporary label [100, 1] obtained in iteration 1 is changed to [55,4) in iteration 3 to indicate that a shorter route has been found through node 4. Also, in iteration 3, node 5 has two alternative labels with the same distance *U5* = 90.

The list for iteration 3 shows that the label for node 2 is now permanent.

Iteration 4. Only node 3 can be reached from node 2. However, node 3 has a permanent label and cannot be relabeled. The new list of labels remains the same as in iteration 3 except that the label at node 2 is now permanent. This leaves node 5 as the only temporary label Because node 5 does not lead to other nodes, its status is converted to permanent, and the process ends.

The shortest route between nodes 1 and any other node in the network is determined by starting at the desired destination node and backtracking through the nodes using the information given by the permanent labels.
For example, shortest route from node 1 to node 2: (2) → [55, 4] → (4) → [40, 3] → (3) → [30,1] → (1)
Thus, the desired route is 1 → 3 → 4 → 2 with a total length of 55 miles.

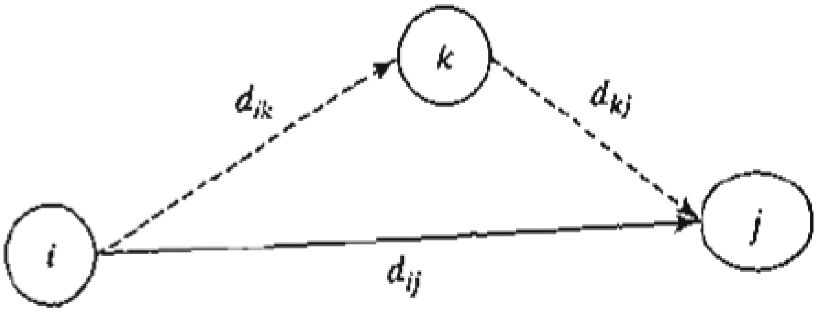
5.2.1.2 Floyd's algorithm

Floyd's algorithm is more general than Dijkstra's because it determines the shortest route between *any* two nodes in the network. The algorithm represents an *n-node* network as a square matrix with *n* rows and *n* columns. Entry (i,j) of the matrix gives the distance *dij* from node *i* to node j, which is finite if *i* is linked directly to j, and infinite otherwise.

The idea of Floyd's algorithm is:

Given three nodes i, j, and *k* in Figure:1 with the connecting distances shown on the three arcs, it is shorter to reach *j* from *i* passing through *k* if

$$d_{ik} + d_{kj} < d_{ij}$$



In this case, it is optimal to replace the direct route from *i →j* with the indirect route *i →k → j.*
This triple operation exchange is applied systematically to the network using the following steps:
Step 0. Define the starting distance matrix *Do* and node sequence matrix So as given below. The diagonal elements are marked with (-) to indicate that they are blocked. Set *k* = 1.

$$D_0 = \begin{array}{c|cccccc|} & 1 & 2 & \cdots & j & \cdots & n \\ \hline 1 & — & d_{12} & \cdots & d_{ij} & \cdots & d_{1n} \\ 2 & d_{21} & — & \cdots & d_{2j} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ i & d_{i1} & d_{i2} & \cdots & d_{ij} & \cdots & d_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n & D_{n1} & d_{n2} & \cdots & d_{nj} & \cdots & — \\ \hline \end{array}$$

$$S_0 = \begin{array}{c|cccccc|} & 1 & 2 & \cdots & j & \cdots & n \\ \hline 1 & — & 2 & \cdots & j & \cdots & n \\ 2 & 1 & — & \cdots & j & \cdots & n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ i & 1 & 2 & \cdots & j & \cdots & n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n & 1 & 2 & \cdots & j & \cdots & — \\ \hline \end{array}$$

General step *k*. Define row *k* and column *k* as *pivot row* and *pivot column.* Apply the *triple operation* to each element *dij* in *Dk - l* , for all *i* and *j*. If the condition

dik + *dkj* < *dij* , (i ≠*k, j*≠*k*, and *i*≠*j)*

is satisfied, make the following hanges:

(a) Create *Dk* by replacing *dij* in *Dk - 1* with *djk* + *dkj*

(b) Create *Sk* by replacing *Sij* in *Sk-l* with *k.* Set *k* = *k* + 1. If *k* = *n* + 1, stop; else repeat step *k.*

Example

For the network in Figure 5.3 , find the shortest routes between every two nodes. The distances (in miles) are given on the arcs. Arc (3,5) is directional, so that no traffic is allowed from node 5 to node 3. All the other arcs allow two-way traffic.

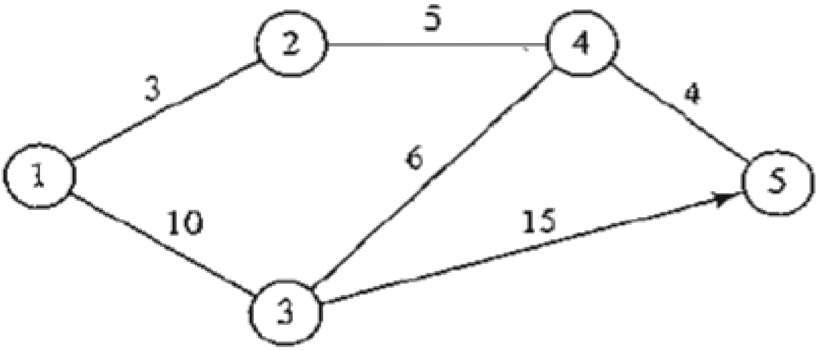
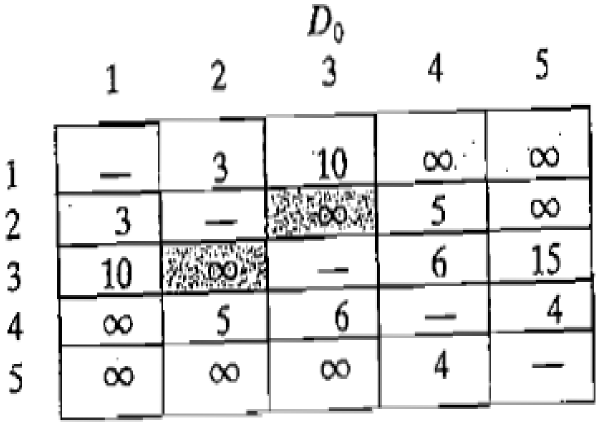


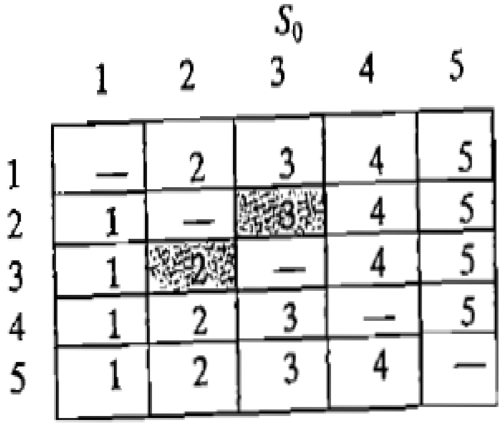
Figure 5.3

Iteration O. The matrices *Do* and So give the initial representation of the network. *Do* is symmetrical, except that *d53* = α because no traffic is allowed from node 5 to node 3.



D_0

	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	∞	5	∞
3	10	∞	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—

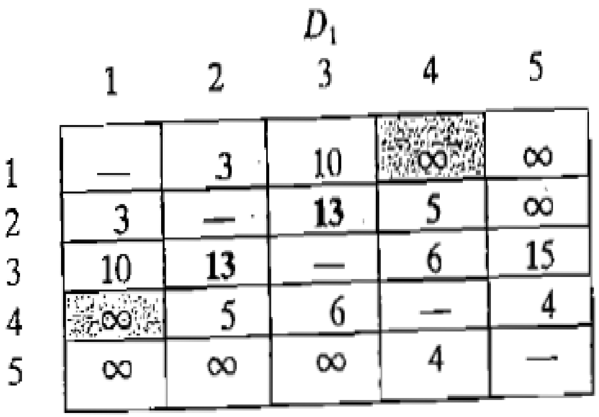


S_0

	1	2	3	4	5
1	—	2	3	4	5
2	1	—	3	4	5
3	1	2	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

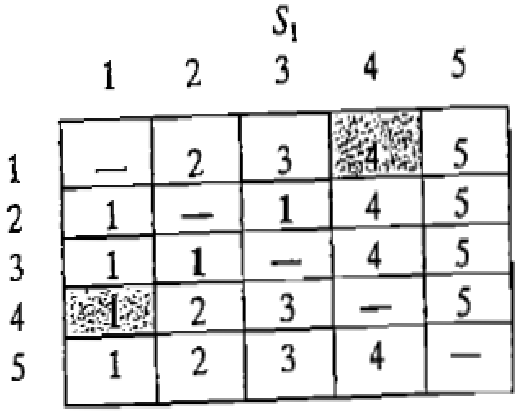
Iteration 1.Set *k* = 1. The pivot row and column are shown by the lightly shaded first row and first column in the Do-matrix. The darker cells, *d13* and *d32*, are the only ones that can be improved by the *triple operation.* Thus, *D1* and SI are obtained from *Do* and So in the following manner:

1. Replace *d23* with *d21* + d13 = 3 + 10 = 13 and set S23 = 1.
2. Replace *d32* with *d31* + *d12* = 10 + 3 = 13 and set S32 = 1.



D_1

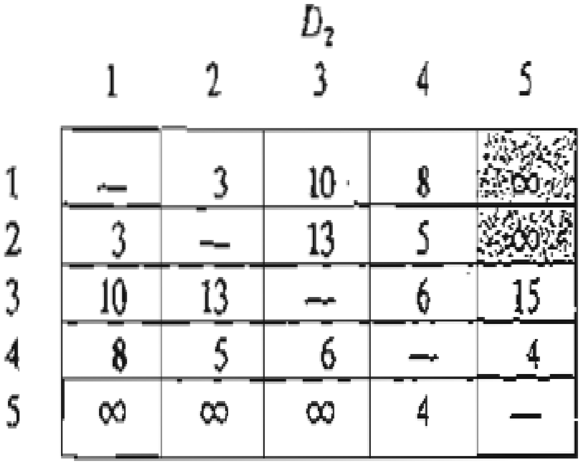
	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	13	5	∞
3	10	13	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—



S_1

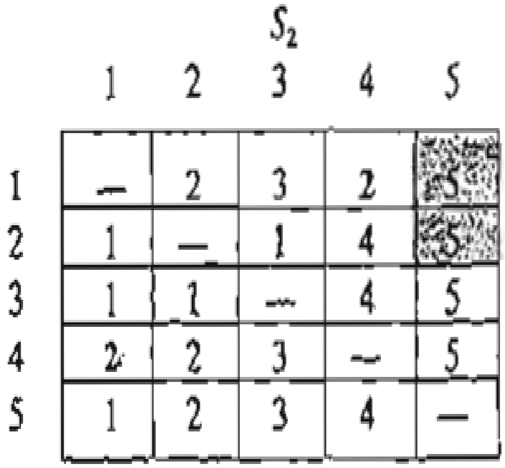
	1	2	3	4	5
1	—	2	3	4	5
2	1	—	1	4	5
3	1	1	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

Iteration 2. Set *k* = 2, as shown by the lightly shaded row and column in *D1*.The *triple operation* is applied to the darker cells in *D1* and *S1*. The resulting changes are shown in bold in *D2* and S2.



D_2

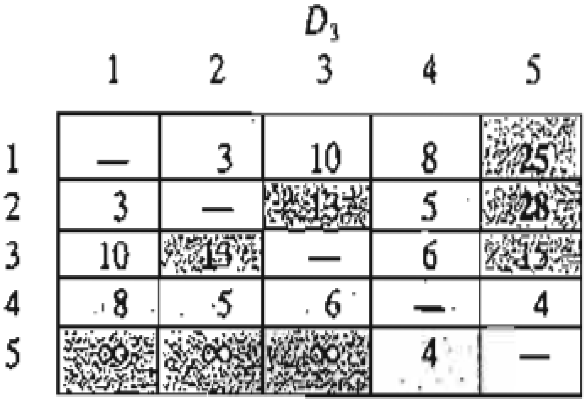
	1	2	3	4	5
1	—	3	10	8	∞
2	3	—	13	5	∞
3	10	13	—	6	15
4	8	5	6	—	4
5	∞	∞	∞	4	—



S_2

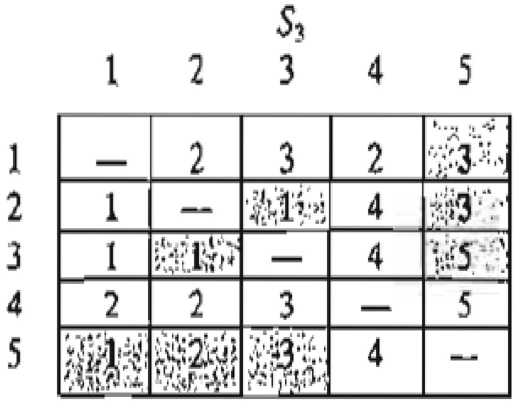
	1	2	3	4	5
1	—	2	3	2	5
2	1	—	1	4	5
3	1	1	—	4	5
4	2	2	3	—	5
5	1	2	3	4	—

Iteration 3. Set *k* = 3, as shown by the shaded row and column in *Dz*. The new matrices are given by *D3* and S3



D_3

	1	2	3	4	5
1	—	3	10	8	25
2	3	—	13	5	28
3	10	13	—	6	15
4	8	5	6	—	4
5	∞	∞	∞	4	—



S_3

	1	2	3	4	5
1	—	2	3	2	3
2	1	—	1	4	3
3	1	1	—	4	5
4	2	2	3	—	5
5	1	2	3	4	—

Iteration 4. Set *k* = 4, as shown by the shaded row and column in *D3*. The new matrices are given by D4 and 54.

D_4

	1	2	3	4	5
1	—	3	10	8	12
2	3	—	11	5	9
3	10	11	—	6	10
4	8	5	6	—	4
5	12	9	10	4	—

S_4

	1	2	3	4	5
1	—	2	3	2	4
2	1	—	4	4	4
3	1	4	—	4	4
4	2	2	3	—	5
5	4	4	4	4	—

Iteration 5. Set *k* = 5, as shown by the shaded row and column in *D4.* No further improvements are possible in this iteration.
The shortest distance from node 1 to node 5 is dl5 = 12 miles.
To determine the associated route, recall that a segment (i, j) represents a direct link only if Sij = j. Otherwise, i and j are linked through at least one other intermediate node. Because s15 = 4 ≠ 5, the route is initially given as 1 → 4 → 5.
1 → 2 → 4 → 5. S12 = 2, S24 = 4, and S45 = 5, no further "dissecting" is needed.

5.2.2 Linear Programming Formulation of the Shortest-Route Problem

Used to find the shortest route between any two nodes in the network. It is equivalent to Floyd's algorithm.
Suppose that the shortest-route network includes *n* nodes and that we desire to determine the shortest route between any two nodes s and *t* in the network. The LP assumes that one unit of flow enters the network at node *s* and leaves at node *t.*
Define

$$x_{ij} = \text{amount of flow in arc } (i, j)$$
$$= \begin{cases} 1, \text{ if arc } (i, j) \text{ is on the shortest route} \\ 0, \text{ otherwise} \end{cases}$$
$$c_{ij} = \text{length of arc } (i, j)$$

Thus, the objective function of the linear program becomes

$$\text{Minimize } z = \sum_{\substack{\text{all defined} \\ \text{arcs } (i, j)}} c_{ij} x_{ij}$$

The constraints represent the *conservation-of-flow equation* at each node:
Total input flow = Total output flow
Mathematically, this translates for node *j* to

$$\begin{pmatrix}\text{External input}\\ \text{into node } j\end{pmatrix} + \sum_{\substack{i\\ \text{all defined}\\ \text{arcs } (i,j)}} x_{ij} = \begin{pmatrix}\text{External output}\\ \text{from node } j\end{pmatrix} + \sum_{\substack{k\\ \text{all defined}\\ \text{arcs } (j,k)}} x_{jk}$$

Example:
Figure 5.4. shows how the unit of flow enters at node 1 and leaves at node 2. Determine the shortest route from node 1 to node 2-that is, s = 1 and *t* = 2.

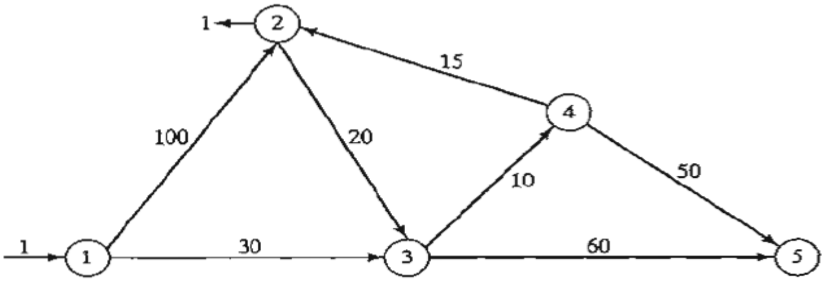


Figure 5.4

We can see from the network that the flow-conservation equation yields

$$\begin{aligned}
&\text{Node 1:} && 1 = x_{12} + x_{13}\\
&\text{Node 2:} && x_{12} + x_{42} = x_{23} + 1\\
&\text{Node 3:} && x_{13} + x_{23} = x_{34} + x_{35}\\
&\text{Node 4:} && x_{34} = x_{42} + x_{45}\\
&\text{Node 5:} && x_{35} + x_{45} = 0
\end{aligned}$$

The complete LP can be expressed as

	x_{12}	x_{13}	x_{23}	x_{34}	x_{35}	x_{42}	x_{45}	
Minimize $z =$	100	30	20	10	60	15	50	
Node 1	1	1						= 1
Node 2	−1		1			−1		= −1
Node 3		−1	−1	1	1			= 0
Node 4				−1		1	1	= 0
Node 5					−1		−1	= 0

Notice that column *Xij* has exactly one "1" entry in row *i* and one "-1" entry in row *j*, a typical property of a network LP.

This solution gives the shortest route from node 1 to node 2 as 1 → 3 → 4 → 2, and the associated
distance is z = 55 (miles).

5.3 MAXIMAL FLOW PROBLEM.

Consider a network of pipelines that transports crude oil from oil wells to refineries. Intermediate booster and pumping stations are installed at appropriate design distances to move the crude in the network. Each pipe segment has a finite maximum discharge rate of crude flow (or capacity).A pipe segment may be uni- or bidirectional, depending on its design. Figure 5.5 demonstrates a typical pipeline network. How can we determine the maximum capacity of the network between the wells and the refineries?

The solution of the proposed problem requires equipping the network with a single source and a single sink by using unidirectional infinite capacity arcs as shown by dashed arcs in Figure 5.5. Given arc *(i, j)* with *i* < *j*, use the notation $(\overline{C}_{ij}, \overline{C}_{ji})$ to represent the flow capacities in the two directions *i* → *j* and *j* → *i*, respectively. To eliminate ambiguity, we place $\overline{C}_{ij}$ on the arc next to node *i* with $\overline{C}_{ji}$ placed next to node *j*, as shown in Figure 5.6

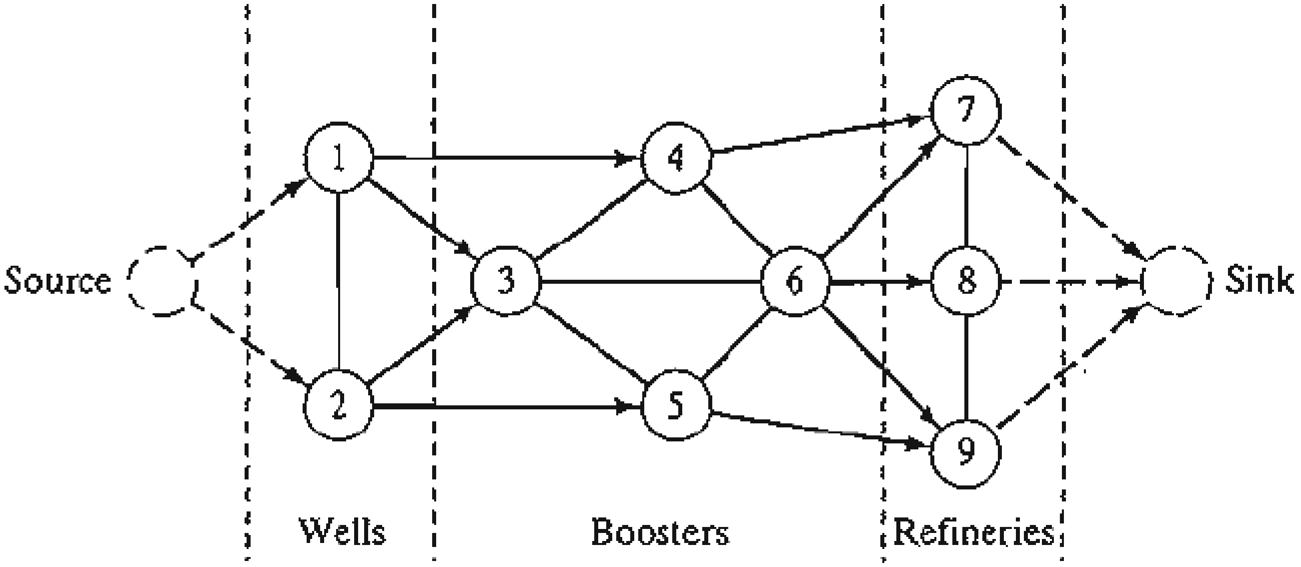


Figure 5.5 Capacitated network connecting wells and refineries through booster stations

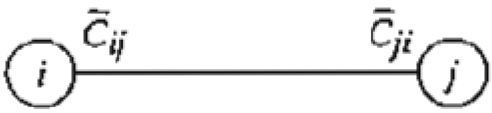


Figure 5.6 Arc flows $\overline{C}_{ij}$ from $i \rightarrow j$ and $\overline{C}_{ji}$ from $j \rightarrow i$

5.3.1 linear Programming Formulation of Maximal Flow Mode

Define *Xij* as the amount of flow in arc *(i,j)* with capacity *Cij* . The objective is to determine *Xij* for all *i* and *j* that will maximize the flow between start node s and terminal node *t* subject to flow restrictions (input flow = output flow) at all but nodes sand *t*.

Example:

In the maximal flow model of Figure 5.7, s = 1 and *t* = 5. The following table summarizes the associated LP with two different, but equivalent, objective functions depending on whether we maximize the output from start node 1 (= z_1) or the input to terminal node 5(=Z_2).

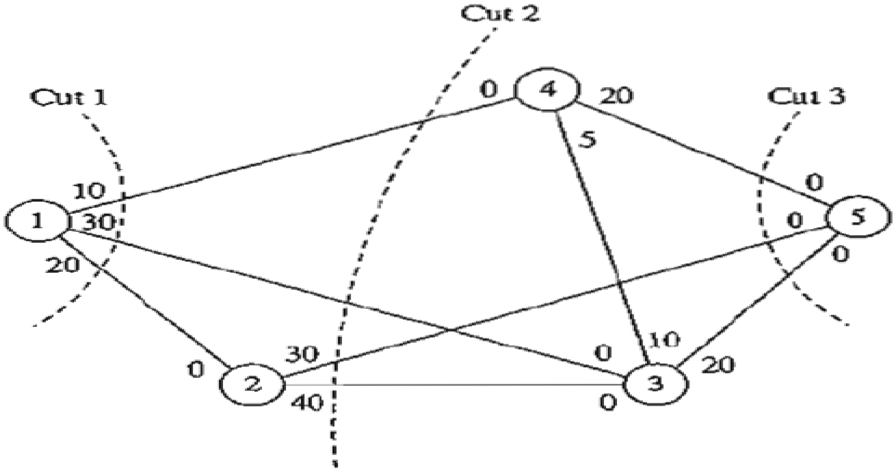


Figure 5.7

	x_{12}	x_{13}	x_{14}	x_{23}	x_{25}	x_{34}	x_{35}	x_{43}	x_{45}	
Maximize z_1 =	1	1	1							
Maximize z_2 =					1		1		1	
Node 2	1			−1	−1					= 0
Node 3		1		1		−1	−1	1		= 0
Node 4			1			1		−1	−1	= 0
Capacity	20	30	10	40	30	10	20	5	20	

The optimal solution using either objective function is

$$x_{12} = 20, x_{13} = 30, x_{14} = 10, x_{25} = 20, x_{34} = 10, x_{35} = 20, x_{45} = 20$$

The associated maximum flow is $Z_1 = Z_2 = 60$.

5.4 Introduction to Non-traditional optimization

In order to survive in today’s dynamic and competitive market; our products are to be cost-effective, compact in size and efficient. To ensure these requirements, one has to follow the principle of optimization, which is nothing but the process of identifying the best solution out of a large number of feasible ones. Therefore, optimization plays a vital role in decision making.

A physical problem can be mathematically expressed as a function of either one or more than one independent variables. If this function carries information of one of the above requirements, it may be termed as an objective function, which is to be either minimized or maximized. Mathematically, this optimum solution (either minimum or maximum) of the objective function can be obtained using the concept of derivative. The derivative of the function becomes equal to zero, corresponding to its optimal solution point. It is to be noted that zero-derivative of a function does not always guarantee an optimum solution, because sometimes a saddle point (also known as an inflection point) may also occur, which is neither a minimum nor maximum point.

In an optimization problem, independent variable(s) used to mathematically define the objective function is/are known as design or decision variable(s). A design variable may take either an integer or a real value. An objective function related to an optimization problem may contain one or more fixed parameter(s), which are known as pre-assigned parameter(s). An optimization problem does not make any sense, unless we specify the range(s) of the design variable(s), these are also known as geometric or side constraint(s). Moreover, an optimization problem may have functional or behavior constraint(s). The presence or absence of the functional constraint(s) decide whether the optimization problem is to be called either a constrained or an un-constrained one, respectively. An optimization problem may be either a linear or a non-linear one. It is known as a linear one, if both its objective function and functional constraint(s) do not carry any non-linear term.

It is to be noted that there may be more than one objective functions in some of the optimization problems, and these are popularly known as multi-objective optimization problems.

5.4.1 Traditional Optimization Tools

Traditional optimization tools generally start from a randomly chosen initial solution and move towards the optimum solution iteratively. Search direction and step length are the two important parameters to be decided by optimization algorithms. There exist a large number of traditional optimization tools, and they are broadly classified into two groups, namely direct search and gradient-based methods.

Direct search methods include random search method, univariate method, pattern search method, and others. Out of these methods, random search method is the most popular one, in which the search direction is decided at each iteration randomly and the step length is specified by the user. Here, the quality of the solution is decided using the objective function value. As the gradient information of the objective function is not required, random search method can be used to handle optimization problem involving discontinuous objective function. However, its search speed may not be high always and consequently, may take a large number of iterations to reach optimal solution.

There exist a few gradient-based methods, namely steepest descent method, conjugate gradient method, quasi-Newton method, and others. Steepest descent method is the most widely used one out of the above methods. In these methods, search direction is decided by the gradient of objective function. The rate of change of an objective function is the maximum along its gradient direction, and consequently, these methods become reasonably fast. As gradient is a local property of an objective function, the chance of the solutions of gradient-based methods for getting trapped into local minima is more. The step length is either assumed to have a fixed value (specified by the user) or determined iteratively using the principle of optimization.

5.4.2 Drawbacks of Traditional Optimization Tools

Traditional optimization tools have the following drawbacks

1. Final solution is dependent on the initially chosen random solution. There is no guarantee that the obtained solution will be a globally optimal one.
2. Optimization problems involving discontinuous objective functions cannot be tackled using the gradient-based methods. Moreover, the solutions of gradient-based methods may get stuck at local optimum points.
3. There exist a variety of optimization problems. A particular traditional optimization method may be suitable for solving only one type of problems. Thus, there is no versatile optimization method, which can be used to solve a variety of problems.

5.4.3 Non-Traditional Optimization Tools

We, human beings, have a natural tendency to follow the way the nature has solved complex optimization problems, whenever we fail to solve them using traditional optimization methods. Some natural processes, such as biological, physical processes etc. are modeled artificially to develop optimization tools for solving the problems.

Non-Traditional methods include
1. Genetic algorithms (GA) ,
2. Genetic programming (GP) ,
3. Evolution strategies (ES) ,
4. Simulated annealing (SA) ,
5. Ant colony optimization (ACO) ,
6. Differential evolution (DE) ,
7. Cultural algorithm (CA) ,
8. Particle swarm optimization (PSO) ,
9. Evolutionary programming (EP) ,
10. Tabu search , and others.

5.5 COMPUTATIONAL COMPLEXITY

The time at which an algorithm produces a successful output is known as time complexity. Problems those can be solvable by a computer is of polynomial time algorithms. The following table shows some polynomial time and exponential time algorithms.

POLYNOMIAL TIME	EXPONENTIAL TIME
Linear Search-n	0/1 knapsack-2^n
Binary Search-log n	Travelling Sales Person-2^n
InsertionSort-n^2	Sum of subsets-2^n
Mergesort-n log n	Graph coloring-2^n
Matrix multiplication-n^3	Hamiltonian cycle-2^n

NP-Hard

Problems those have no polynomial time algorithms to solve are NP-Hard.

Nondeterministic Algorithms

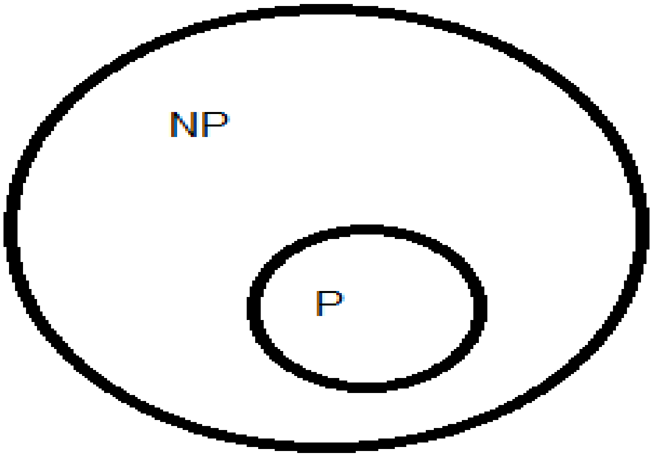
NP-Hard problems can be solved in polynomial time by assuming some part of the algorithms as Nondeterministic polynomial time(can find solution with one unit of time in the future), such problems are known as NP-Complete problems.

- Nondeterministic

```
Algorithm Nsearch(A,n,key)
{
  j=choice();          →1 unit of time
  if (key=A[j])
   {
     write(j);
     success();        →1 unit of time
    }
   write(0);
   Failure();                 →1 unit of time
}
```

P $\underline{c}$ NP(Polynomial time less than or equal Nondeterministic Polynomaial)

Currently those problems are in P were in NP some time before. So those NP problems can be in P in the future. When all the NP problems changed to solvable that is in P, P = NP so P $\underline{c}$ NP.



Reduction

If we know the solution status of one algorithm then we can convert other problems to the known by using the reduction method. The problem for which we know the solution status is known as the base problem. CNF satisfiability problem is considered as the base problem.

CNF:

$X_i=\{x_1,x_2,x_3\}$

CNF= $(x_1Vx_2V\ x_3) \wedge (x_1Vx_2V\ x_3)$

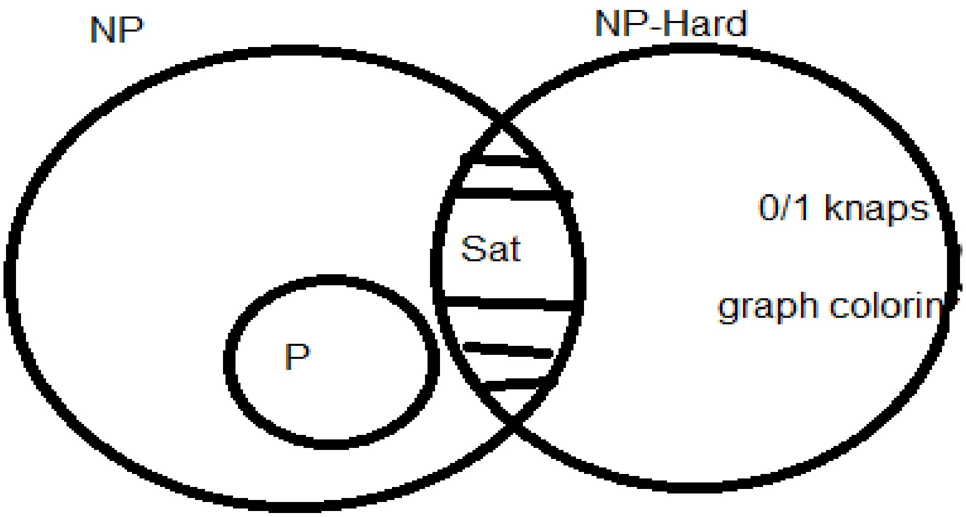
If base problem is NP-Hard then the reduced problem is also NP-Hard

One problem can be converted to another in polynomial time

$$\overbrace{\text{Sat } \alpha \text{ 0/1 knapsack}}^{\text{polynomial time}}$$

If an NP-Hard problem has Non-deterministic solution the that is NP-Complete

CNF satisfiability problem is NP-Hard, NP, NP-complete.



Why using approximation?

We are not able to solve NP-complete problems efficiently, that is, there is no known way to solve them in polynomial time unless P = NP.

Why not looking for an approximate solution?

Optimization Problem, O = (I , SOL, m, type)

I the instance set

SOL(i) the set of feasible solutions for instance i (SOL(i) for i ϵ I)

m(i, s) the meassure of solution s with respect to instance i (positive integer for i ϵ I) and s ϵ SOL(i)

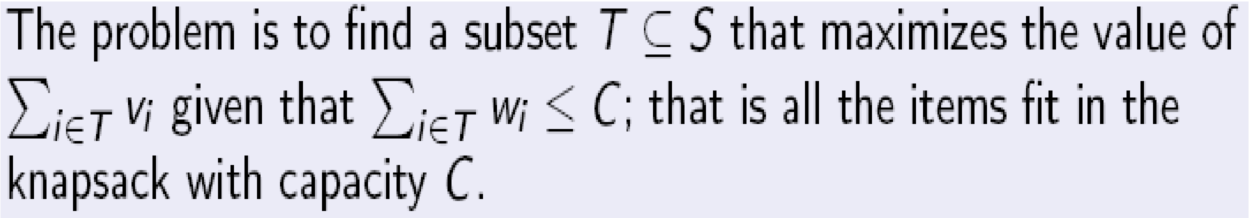
type ϵ {min, max}

$$opt(i) = \underset{s \in SOL(i)}{type} \; m(i, s)$$

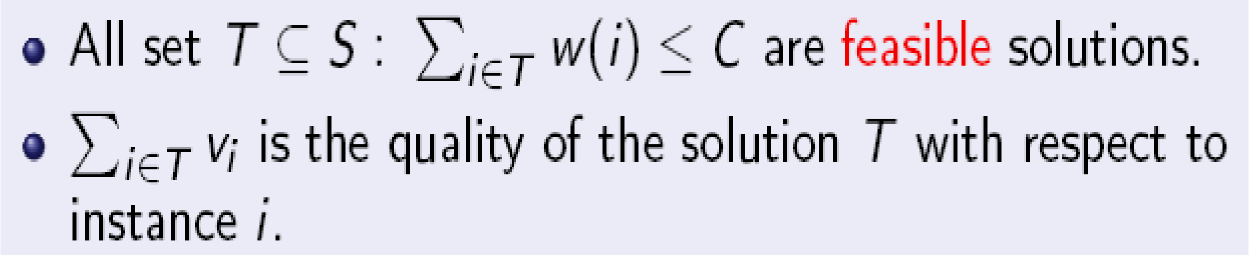
Example for Optimization Problem

Given is a knapsack with capacity C and a set of items S = {1, 2, . . . , n}, where item i has weight w_i and value v_i .

Problem

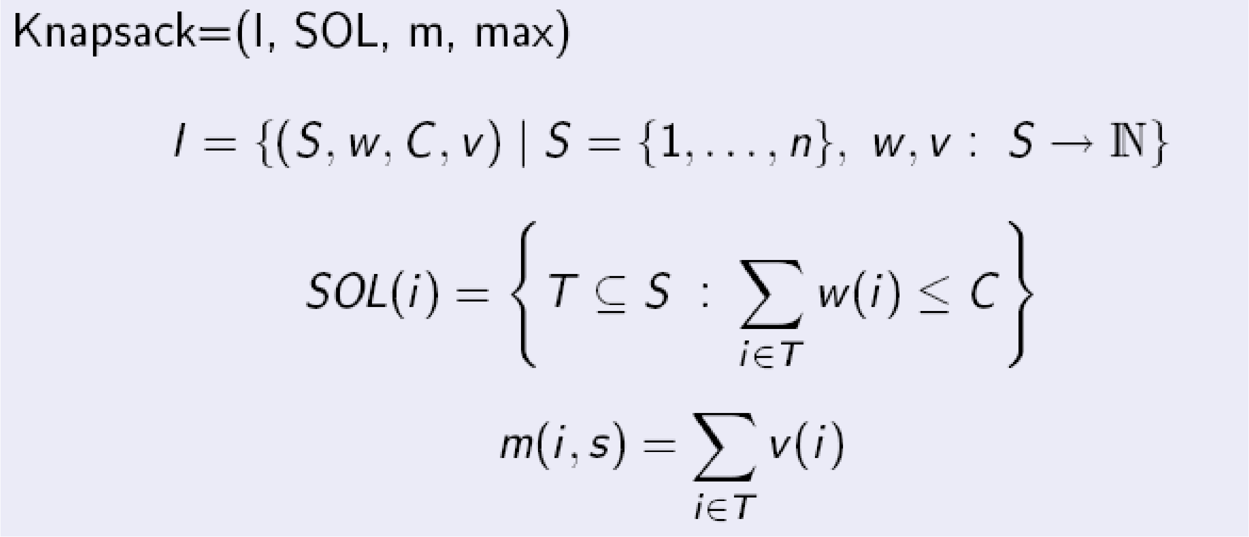


The problem is to find a subset $T \subseteq S$ that maximizes the value of $\sum_{i \in T} v_i$ given that $\sum_{i \in T} w_i \leq C$; that is all the items fit in the knapsack with capacity C.



- All set $T \subseteq S : \sum_{i \in T} w(i) \leq C$ are feasible solutions.
- $\sum_{i \in T} v_i$ is the quality of the solution T with respect to instance i.

Instance



Knapsack=(I, SOL, m, max)

$$I = \{(S, w, C, v) \mid S = \{1, \ldots, n\},\ w, v :\ S \rightarrow \mathbb{N}\}$$

$$SOL(i) = \left\{ T \subseteq S\ :\ \sum_{i \in T} w(i) \leq C \right\}$$

$$m(i, s) = \sum_{i \in T} v(i)$$

5.6 TABU SEARCH

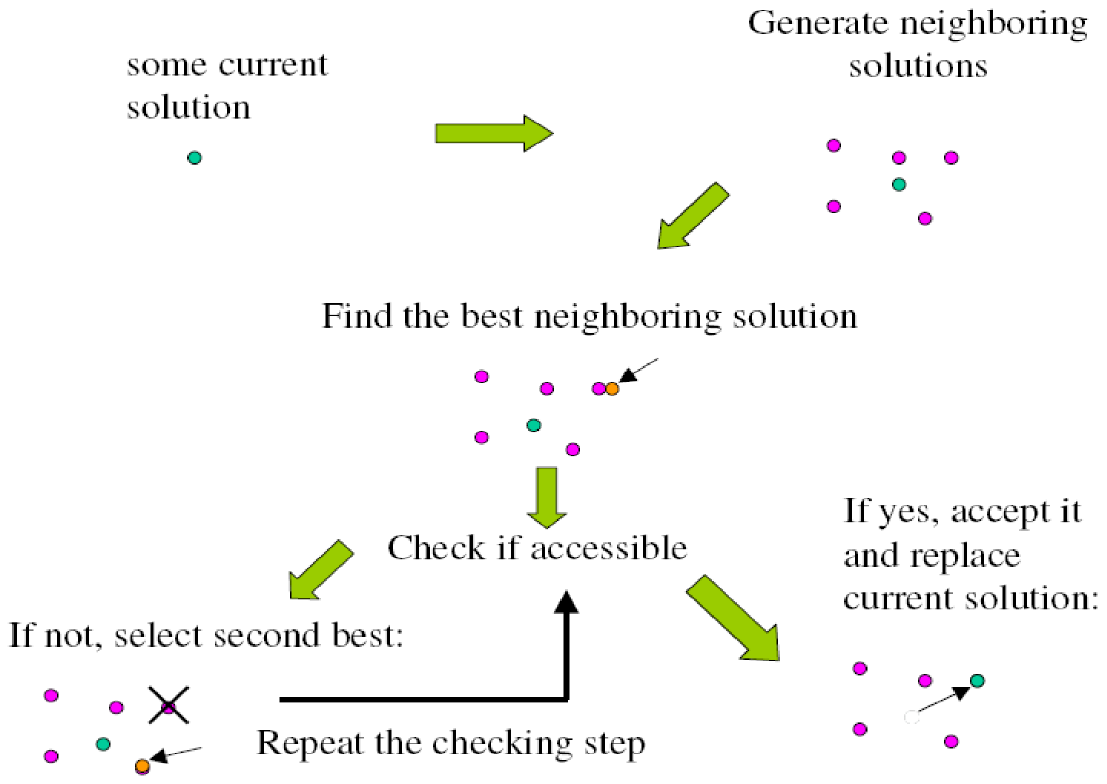
The word tabu (or taboo) comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga island to indicate things that cannot be touched because they are sacred.
“a meta-heuristic superimposed on another heuristic”. By Glover (1986)

The tabu search begins by marching to a local minima. To avoid retracing the steps used, the method records recent moves in one or more tabu lists. The original intent of the list was not to prevent a previous move from being repeated, but rather to insure it was not reversed.

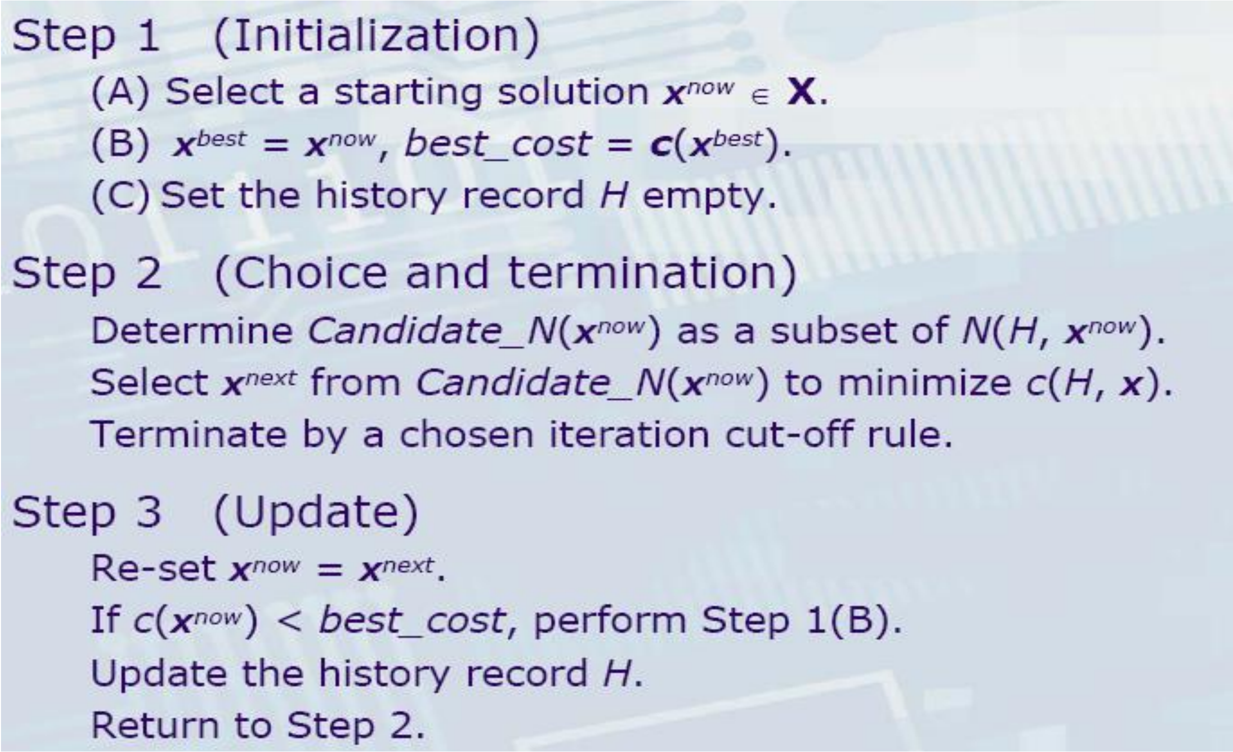
The tabu lists are historical in nature and form the tabu search memory. The role of the memory can change as the algorithm proceeds. At initialization the goal is make a coarse examination of the solution space, known as
“diversification”, but as candidate locations are identified the search is more focused to produce local optimal solutions in a process of “intensification”. In many cases the differences between the various implementations of the tabu
method have to do with the size, variability, and adaptability of the tabu memory to a particular problem domain.

5.6.1 Basics A heuristic search method

Tabu restricts some search of neighboring solutions. Aspiration allows exception of the tabu. Accessible Solutions – The solutions that are not in the tabu list, or in the tabu list but satisfy aspiration conditions of Tabu
Search.


some current solution
Generate neighboring solutions
Find the best neighboring solution
Check if accessible
If yes, accept it and replace current solution:
If not, select second best:
Repeat the checking step

5.6.2 The Basic TS Algorithm



Step 1 (Initialization)

(A) Select a starting solution $\boldsymbol{x}^{now} \in \mathbf{X}$.

(B) $\boldsymbol{x}^{best} = \boldsymbol{x}^{now}$, $best\_cost = \boldsymbol{c}(\boldsymbol{x}^{best})$.

(C) Set the history record H empty.

Step 2 (Choice and termination)

Determine $Candidate\_N(\boldsymbol{x}^{now})$ as a subset of $N(H, \boldsymbol{x}^{now})$.

Select $\boldsymbol{x}^{next}$ from $Candidate\_N(\boldsymbol{x}^{now})$ to minimize $c(H, \boldsymbol{x})$.

Terminate by a chosen iteration cut-off rule.

Step 3 (Update)

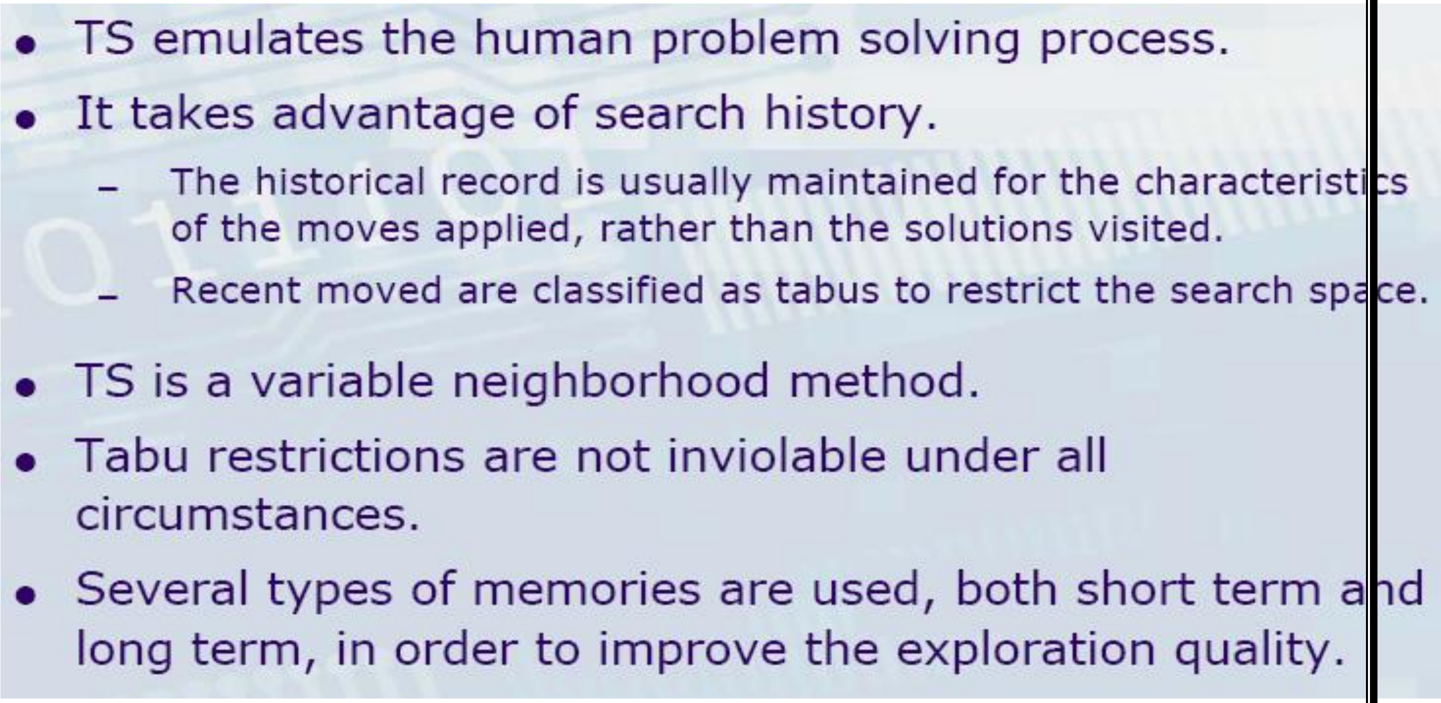
Re-set $\boldsymbol{x}^{now} = \boldsymbol{x}^{next}$.

If $c(\boldsymbol{x}^{now}) < best\_cost$, perform Step 1(B).

Update the history record H.

Return to Step 2.

5.6.3 Main Features



- TS emulates the human problem solving process.
- It takes advantage of search history.
  - The historical record is usually maintained for the characteristics of the moves applied, rather than the solutions visited.
  - Recent moved are classified as tabus to restrict the search space.
- TS is a variable neighborhood method.
- Tabu restrictions are not inviolable under all circumstances.
- Several types of memories are used, both short term and long term, in order to improve the exploration quality.

The tabu search begins by marching to a local minima. To avoid retracing the steps used, the method records recent moves in one or more tabu lists.

The original intent of the list was not to prevent a previous move from being repeated, but rather to insure it was not reversed.

5.6.4 Tabu Search

1. Tabu search is a local search strategy with a flexible memory structure.
2. Adaptive memory
3. Responsive exploration strategies
Always move to the best available neighborhood solution point, even if it is worse than the current solution point.
4. Tabu list:
Maintain a list of solution points that must be avoided (not allowed) or a list of move attributes that are not allowed. This is referred to as the tabu list.
Update this list based on some memory structure (short-term memory).
5. Aspiration Criteria
Allow for exceptions from the tabu list, if such moves lead to promising solutions.
6. Intensification and diversification
Diversification: Search the unexplored area of the solution space by Increase tabu tenure, Change tabu restriction, etc
7. Long Term Memory
Frequency based memory/Recency based memory. Adaptive Memory Programming (AMP)

First Level Tabu Search

In a first level tabu search approach, the following issues should be considered:

1. Solution representation and evaluation
2. Neighborhood structure/Move mechanism
3. Move Attribute (used for tabu classification)
4. Tabu status and duration (tenure)
5. Aspiration criteria
6. Stopping criteria
7. Initial Solution

Tabu Tenure

How to decide tabu duration (tenure)?
1. Effective tabu tenure depends on the instance (size, etc.)
2. An effective range for tabu tenure can be determined experimentally
3. Static versus Dynamic tabu tenure

Aspiration Criteria
Choices for aspiration criteria
1. Better than the best found so far
2. Aspiration-by-default
Once a first level tabu search algorithm is designed and implemented, we can incorporate other features to enhance the algorithm.

TABU SEARCH SHORT-TERM MEMORY COMPONENT.

Memory Aspects
1.recency (short term)
how recently was I here?
2.frequency (long term)
how often have I been here?
3.quality (aspiration)
how good is being here?
4.influence (aspiration)
how far away am I from where I have just been?

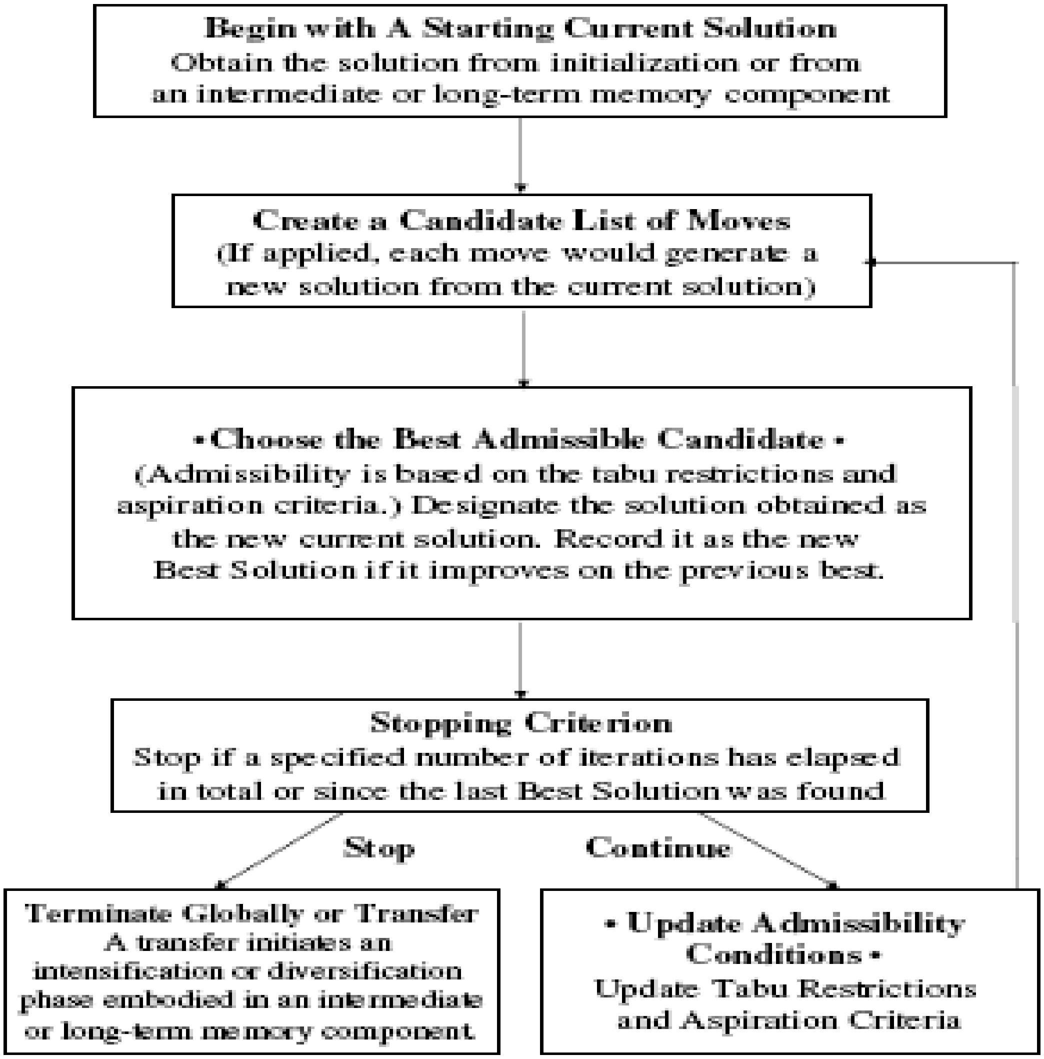

Begin with A Starting Current Solution
Obtain the solution from initialization or from an intermediate or long-term memory component
Create a Candidate List of Moves
(If applied, each move would generate a new solution from the current solution)
• Choose the Best Admissible Candidate •
(Admissibility is based on the tabu restrictions and aspiration criteria.) Designate the solution obtained as the new current solution. Record it as the new Best Solution if it improves on the previous best.
Stopping Criterion
Stop if a specified number of iterations has elapsed in total or since the last Best Solution was found
Stop
Continue
Terminate Globally or Transfer
A transfer initiates an intensification or diversification phase embodied in an intermediate or long-term memory component
• Update Admissibility Conditions •
Update Tabu Restrictions and Aspiration Criteria

Figure 5.1 Tabu search short-term memory component.

Example:

Tabu Search for TSP

Tabu Search is a heuristic that, if used effectively, can promise an efficient near-optimal solution to the TSP.

1. **Solution Representation**

A feasible solution is represented as a sequence of nodes, each node appearing only once and in the order it is visited. The first and the last visited nodes are fixed to 1. The starting node is not specified in the solution representation and is always understood to be node 1.

3	5	2	4	7	6	8	1

Figure 1: Solution Representation

2. **Initial Solution**: A good feasible, yet not-optimal, solution to the TSP can be found quickly using a greedy approach. Starting with the first node in the tour, find the nearest node. Each time find the nearest unvisited node from the current node until all the nodes are visited.

3. Neighborhood: A neighborhood to a given solution is defined as any other solution that is obtained by a pair wise exchange of any two nodes in the solution. This always guarantees that any neighborhood to a feasible solution is always a feasible solution (i.e, does not form any sub-tour).

At each iteration, the neighborhood with the best objective value (minimum distance) is selected.

4.

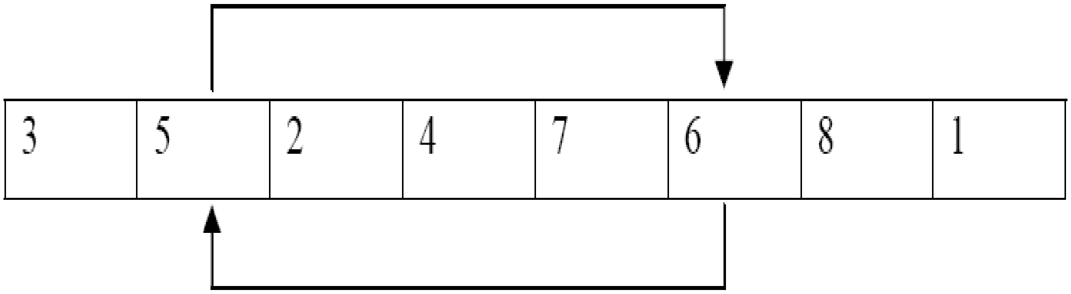
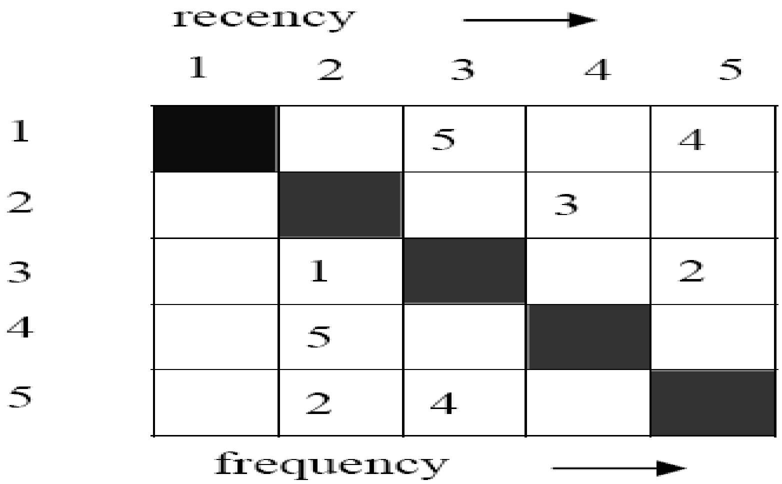


Figure 2: Neighborhood solution obtained by swapping the order of visit of cities 5 and 6

5. **Tabu List**: To prevent the process from cycling in a small set of solutions, some attribute of recently visited solutions is stored in a Tabu List, which prevents their occurrence for a limited period. For our problem, the attribute used is a pair of nodes that have been exchanged recently. A Tabu structure stores the number of iterations for which a given pair of nodes is prohibited from exchange as illustrated in Figure 3.



recency →

	1	2	3	4	5
1	■		5		4
2		■		3	
3		1	■		2
4		5		■	
5		2	4		■

frequency →

6. **Aspiration criterion**: Tabus may sometimes be too powerful: they may prohibit attractive moves, even when there is no danger of cycling, or they may lead to an overall stagnation of the searching process . It may, therefore, become necessary to revoke tabus at times. The criterion used for this to happen in the present problem of TSP is to allow a move, even if it is tabu, if it results in a solution with an objective value better than that of the current best-known solution.

7. **Diversification**: Quite often, the process may get trapped in a space of local optimum. To allow the process to search other parts of the solution space (to look for the global optimum), it is required to diversify the search process, driving it into new regions. This is implemented in the current problem using “frequency based memory”.

8. Termination criteria: The algorithm terminates if a pre-specified number of iterations is reached

Module VI

6.1 GENETIC ALGORITHMS

1. Genetic algorithms are based on the principles of **natural genetics and natural selection**.
2. The basic elements of natural genetics used in the genetic search procedure are
   a) **reproduction**,
   b) **crossover**, and
   c) **mutation**

Outline of the Basic Genetic Algorithm

1. [Start] Generate random population of *n* chromosomes (suitable solutions for the problem)

2. [Fitness] Evaluate the fitness *f(x)* of each chromosome *x* in the population

3. [New population] Create a new population by repeating following steps until the new population is complete

A. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

B. **[Crossover]** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.

C. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).

D. **[Accepting]** Place new offspring in the new population

4. [Replace] Use new generated population for a further run of the algorithm

5. [Test] If the end condition is satisfied, **stop**, and return the best solution in current population

6. [Loop] Go to step **2**

Difference between Gas and traditional method of optimization

1. A population of points (trial design vectors) is used for starting the procedure instead of a single design point.
   i) n design variables
   ii) size of the population- *2n* to 4n

Since several points are used as candidate solutions, GAs are less likely to get trapped at a local optimum.

2. GAs use only the values of the objective function. The derivatives are not used in the search procedure.
3. In GAs the design variables are represented as strings of binary variables that correspond to the chromosomes in natural genetics.
i) Search method - solving discrete and integer programming problems.
ii) continuous design variables, the string length can be varied to achieve any desired resolution.

4. The objective function value corresponding to a design vector plays the role of fitness in natural genetics.
5. In every new generation, a new set of strings is produced by using randomized parents selection and crossover from the old generation.

Limitations of Gas
1. GAs are not suited for all problems, especially problems which are simple and for which derivative information is available.
2. Fitness value is calculated repeatedly which might be computationally expensive for some problems.
3. Being stochastic, there are no guarantees on the optimality or the quality of the solution.
4. If not implemented properly, the GA may not converge to the optimal solution.

6.1.1 Basic concepts

1. Population − It is a subset of all the possible (encoded) solutions to the given problem.
2. Chromosomes − A chromosome is one such solution to the given problem.
3. Gene − A gene is one element position of a chromosome.
4. Allele − It is the value a gene takes for a particular chromosome.

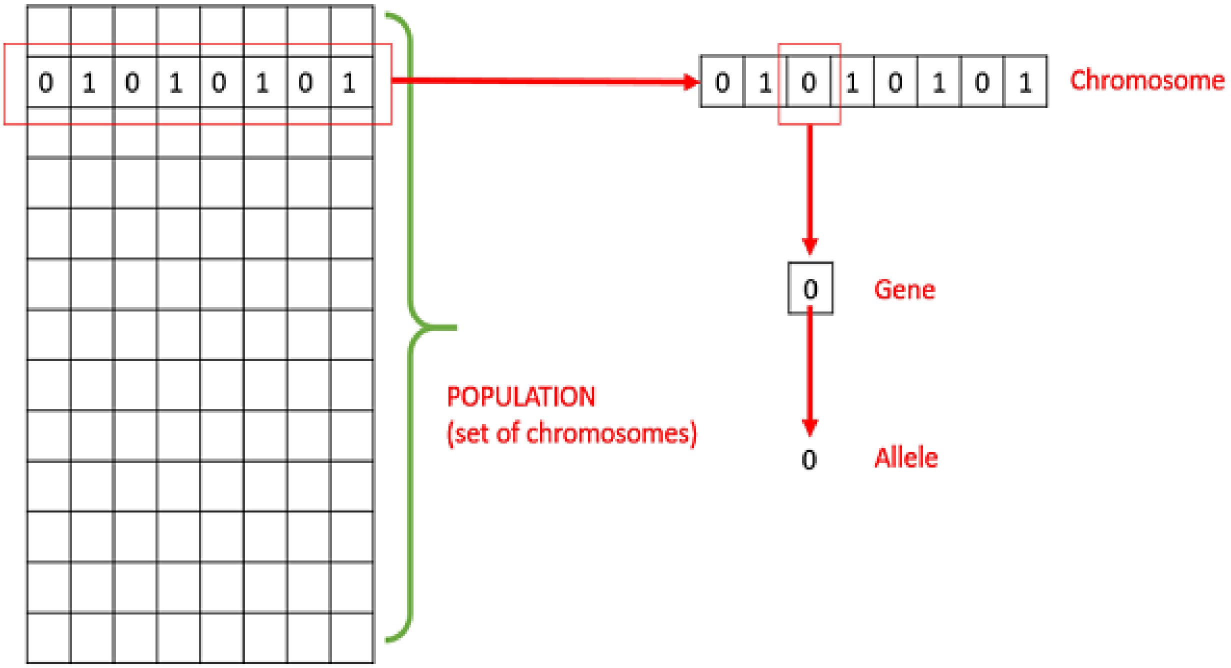


Figure 1 Basic concepts

5. Genotype − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.

6. Phenotype − Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.

7. Decoding and Encoding − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

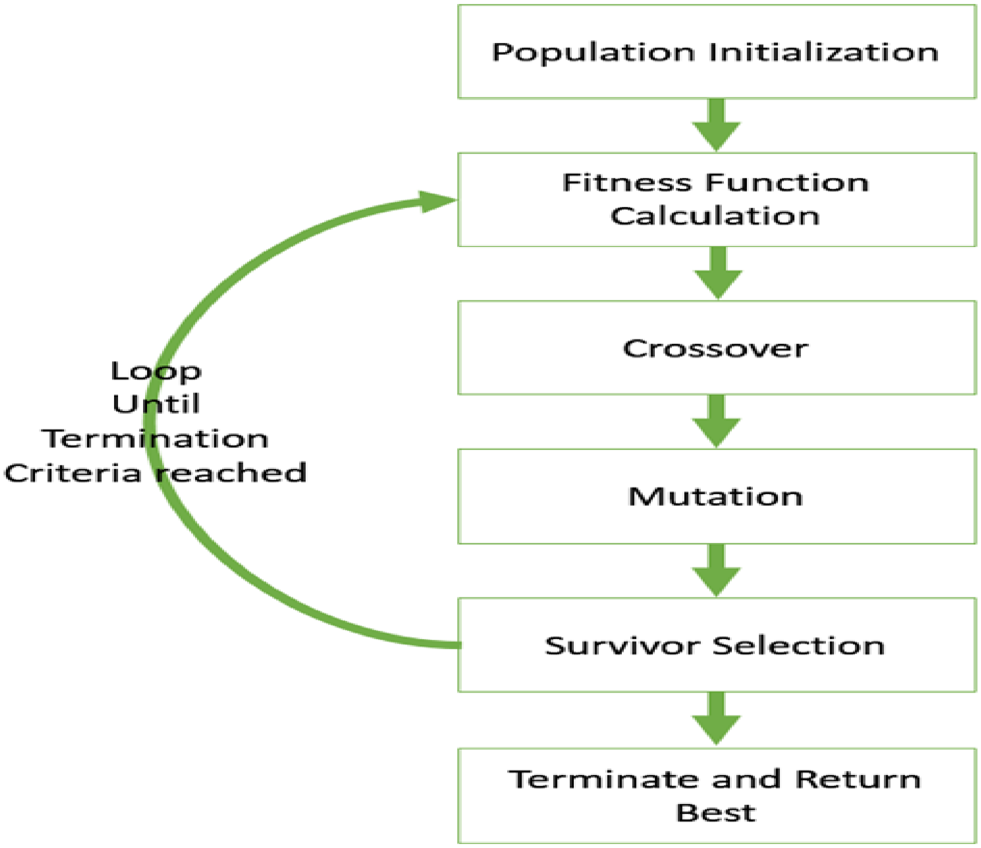

Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached

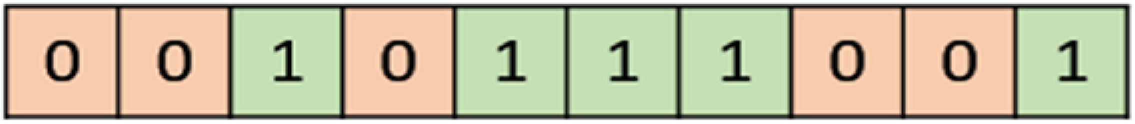
Figure 2 Basic Structure

6.1.2 Encoding (Representation of Design Variables)

Each chromosome is represented by a binary string. Each bit in the string can represent some characteristics of the solution. There are many other ways of encoding. The encoding depends mainly on the solved problem.

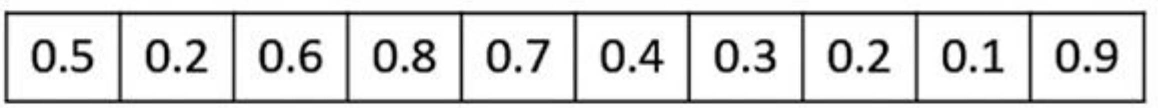
For example, one can encode directly integer or real numbers, sometimes it is useful to encode some permutations and so on.

1. Binary Representation



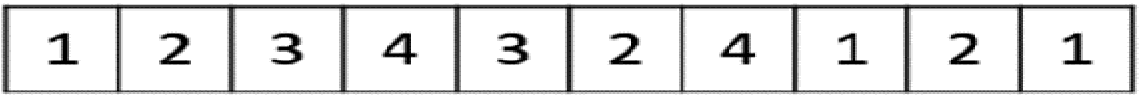
0	0	1	0	1	1	1	0	0	1

2. Real Valued Representation



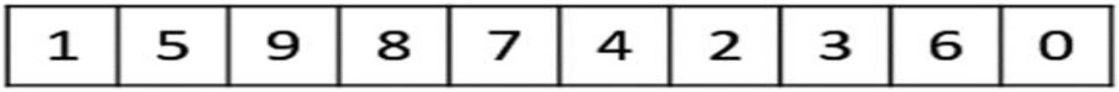
0.5	0.2	0.6	0.8	0.7	0.4	0.3	0.2	0.1	0.9

3. Integer Representation



1	2	3	4	3	2	4	1	2	1

4. Permutation Representation



1	5	9	8	7	4	2	3	6	0

A chromosome should in some way contain information about solution that it represents. The most used way of encoding is a binary string. A chromosome then could look like this:

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

Population Models

There are two population models widely in use -

1. Steady State

In steady state GA, we generate one or two off-springs in each iteration and they replace one or two individuals from the population. A steady state GA is also known as **Incremental GA**.

2. Generational

In a generational model, we generate ‘n’ off-springs, where n is the population size, and the entire population is replaced by the new one at the end of the iteration.

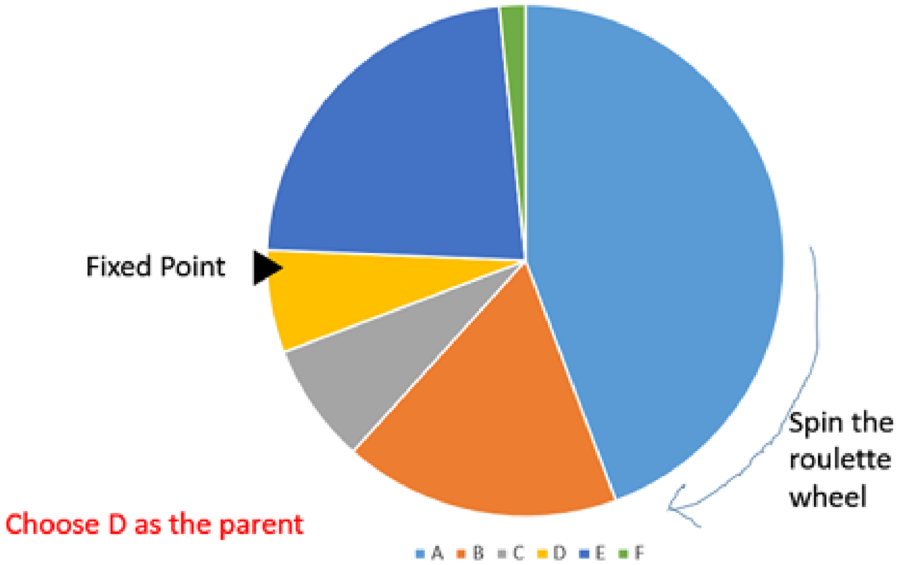
6.1.3 Selection

Fitness Proportionate Selection is one of the most popular ways of parent selection. Every individual can become a parent with a probability which is proportional to its fitness. Fitter individuals have a higher chance of mating and propagating their features to the next generation.

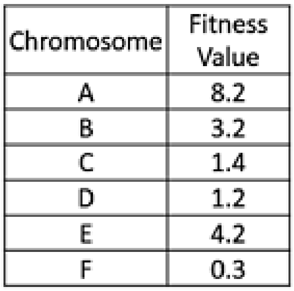
1. Roulette Wheel Selection

In a roulette wheel selection, the circular wheel is divided as described before. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated.

It is clear that a fitter individual has a greater pie on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated. Therefore, the probability of choosing an individual depends directly on its fitness.



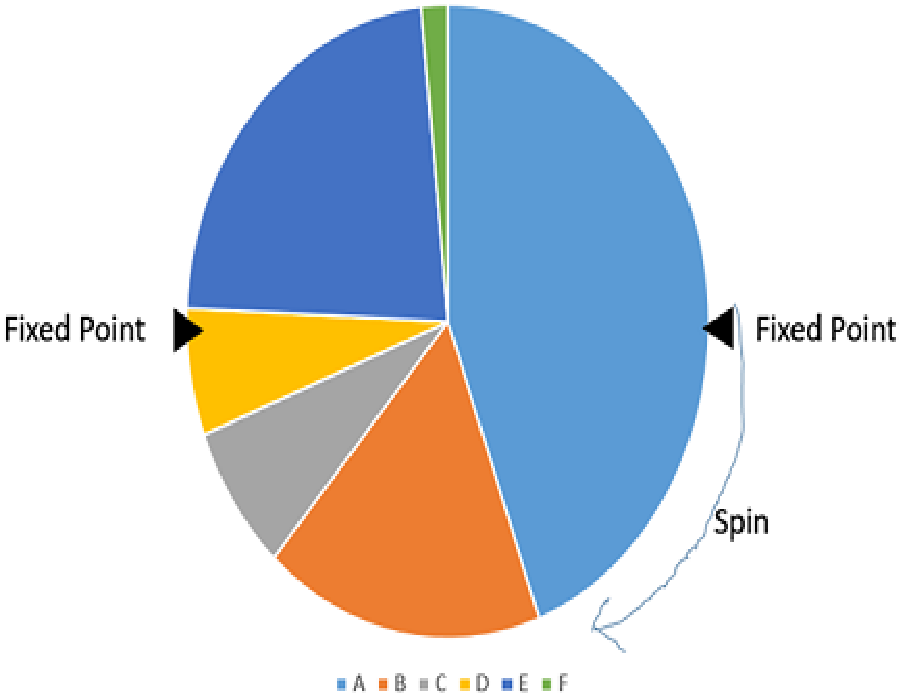
Choose D as the parent



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

2. **Stochastic Universal Sampling (SUS)**

Stochastic Universal Sampling is quite similar to Roulette wheel selection, however instead of having just one fixed point, we have multiple fixed points as shown in the following image. Therefore, all the parents are chosen in just one spin of the wheel. Also, such a setup encourages the highly fit individuals to be chosen at least once.

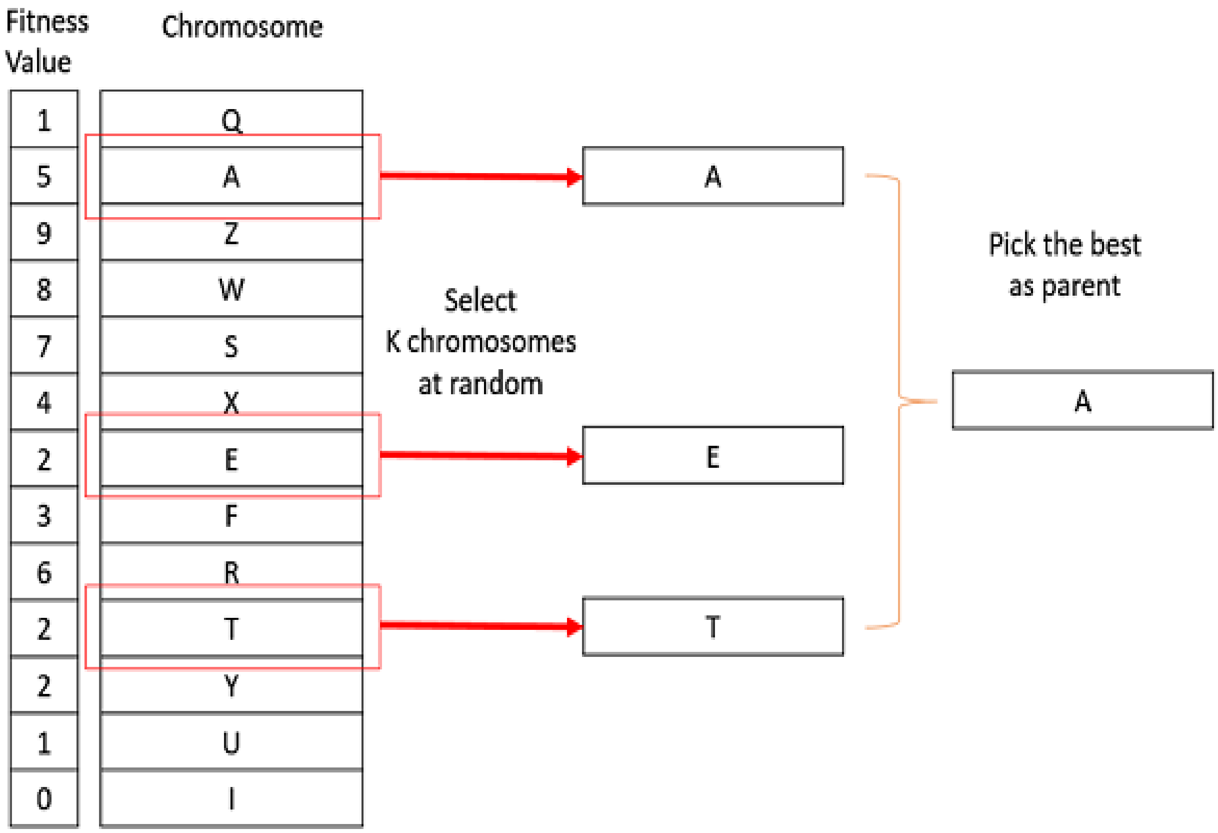


Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

3. Tournament Selection

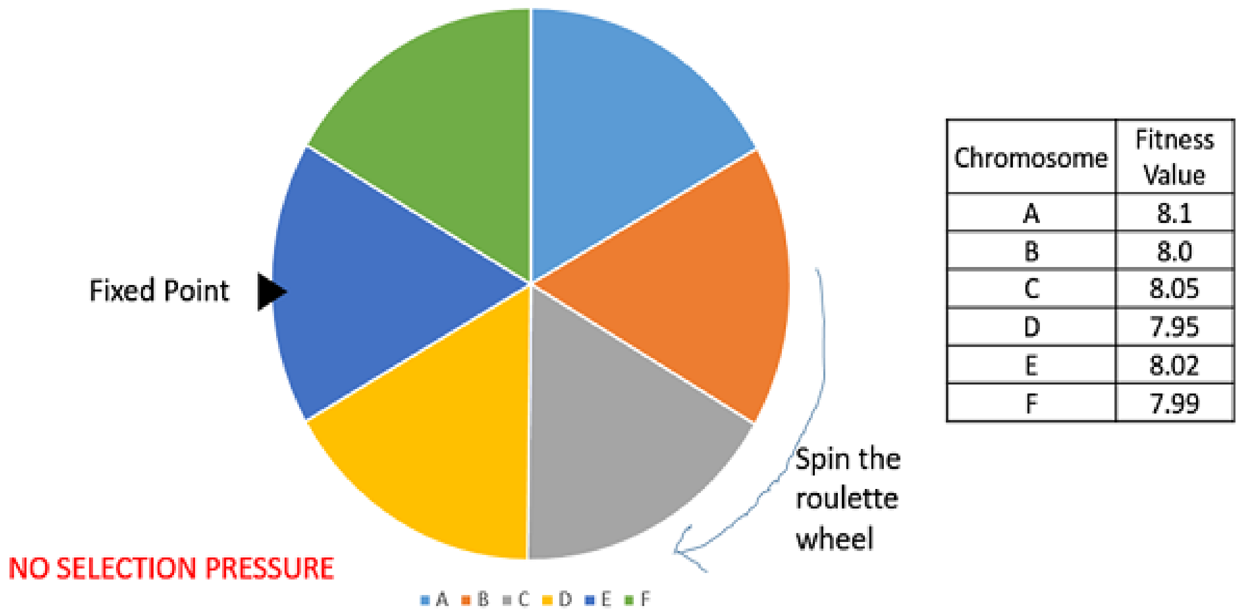
In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent.

Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.



4. Rank Selection

Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run). This leads to each individual having an almost equal share of the pie (like in case of fitness proportionate selection) as shown in the following image and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent. This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.



Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

5. Random Selection

In this strategy we randomly select parents from the existing population. There is no selection pressure towards fitter individuals and therefore this strategy is usually avoided.

6.1.4 Crossover

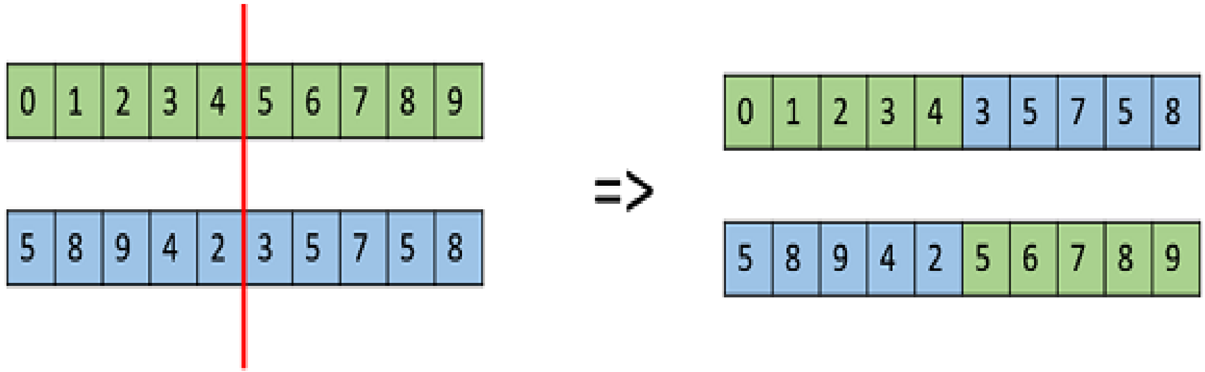
Crossover operates on selected genes from parent chromosomes and creates new offspring. The simplest way how to do that is to choose randomly some crossover point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent.

Crossover can be illustrated as follows: (| is the crossover point):

Chromosome 1	**11011 | 00100110110**
Chromosome 2	**11011 | 11000011110**
Offspring 1	**11011 | 11000011110**
Offspring 2	**11011 | 00100110110**

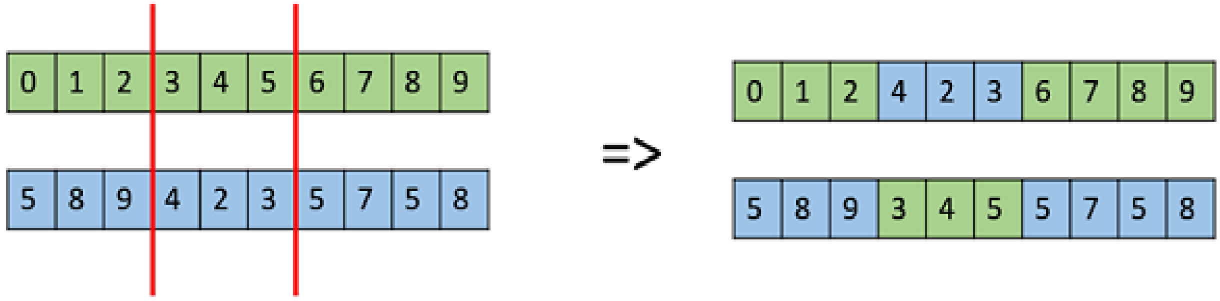
1. One Point Crossover

In this one-point crossover, a random crossover point is selected and the tails of its two parents



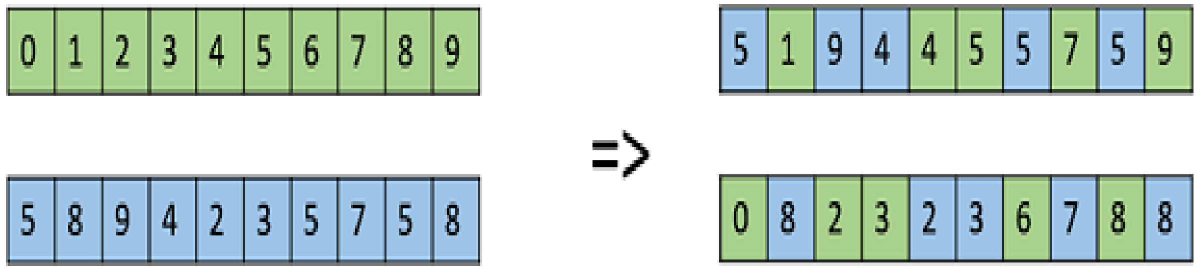
2. Multi Point Crossover

Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



3. Uniform Crossover

Treat each gene separately. Flip a coin for each chromosome to decide whether or not it'll be included in the off-spring.

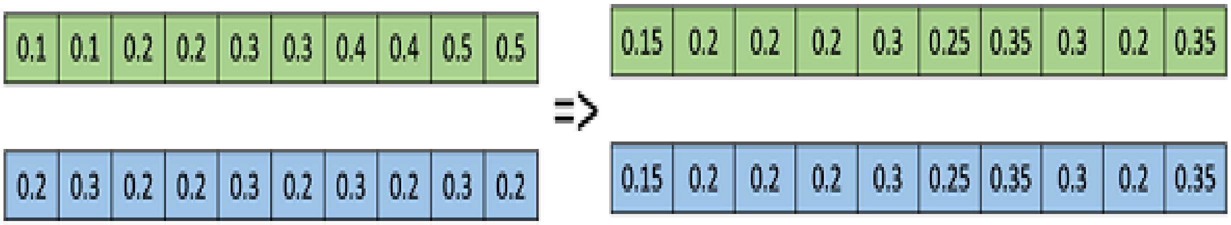


4. Whole Arithmetic Recombination

This is commonly used for integer representations and works by taking the weighted average of the two parents by using the following formulae −

Child1 = α.x + (1-α).yChild2 = α.x + (1-α).y

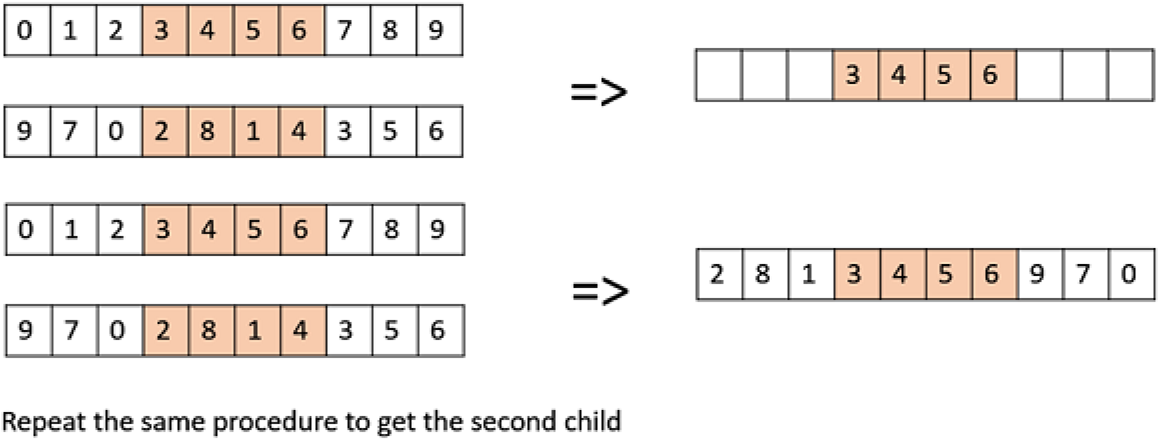
Obviously, if α = 0.5, then both the children will be identical as shown in the following image.



5. Davis' Order Crossover (OX1)

OX1 is used for permutation based crossovers with the intention of transmitting information about relative ordering to the off-springs. It works as follows −

1. Create two random crossover points in the parent and copy the segment between them from the first parent to the first offspring.
2. Now, starting from the second crossover point in the second parent, copy the remaining unused numbers from the second parent to the first child, wrapping around the list.
3. Repeat for the second child with the parent's role reversed.



6.1.5 Mutation

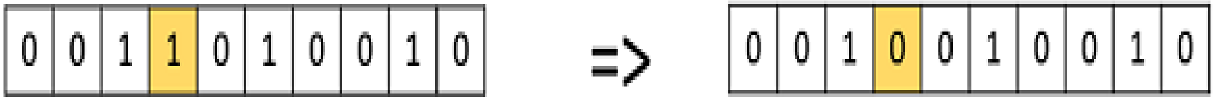
1. After a crossover is performed, mutation takes place.
2. Mutation is intended to prevent falling of all solutions in the population into a local optimum of the solved problem.
3. Mutation operation randomly changes the offspring resulted from crossover.
4. In case of binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	110**0**111000011110
Mutated offspring 2	110110**1**1001101**1**0

Mutation Operators

1. **Bit Flip Mutation**

In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.



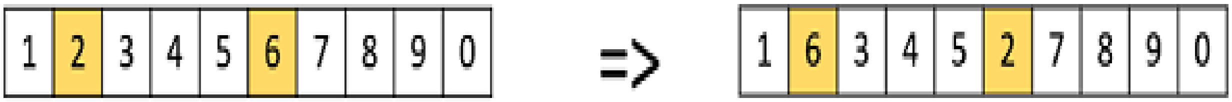
0 0 1 1 0 1 0 0 1 0 => 0 0 1 0 0 1 0 0 1 0

2. **Random Resetting**

Random Resetting is an extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

3. **Swap Mutation**

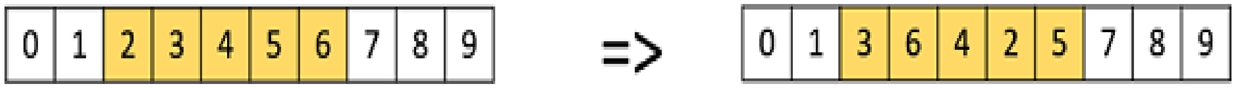
In swap mutation, we select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings.



1 2 3 4 5 6 7 8 9 0 => 1 6 3 4 5 2 7 8 9 0

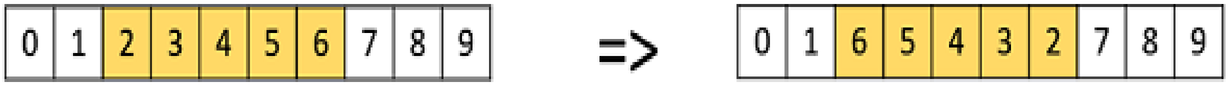
4. **Scramble Mutation**

Scramble mutation is also popular with permutation representations. In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.



5. Inversion Mutation

In inversion mutation, we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.



6.2 SIMULATED ANNEALING

Simulated Annealing (SA) is a method to solve an optimization problem by simulating a stochastic thermal dynamics of a metal cooling process.

Stochastic means having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

SA obtains an optimal solution by simulating a physical fact that liquid metal transmutes to be crystal (which has the smallest internal thermal energy) if it is cooled satisfactory slowly from a high temperature state (with large internal thermal energy).

Transmute means to change something completely, especially into something different and better.

Functional minimization corresponds to minimization of internal thermal energy of metal in a melting pot.
-The idea to solve the optimization problem by using this fact is proposed by Kirkpatric. He demonstrated that the method is successfully applicable to so-called combinatorial optimization problem.

Combinatorial optimization problem- finding an optimal object from a finite set of objects.
Examples

1. the travelling salesman problem ("TSP") ,
2. the minimum spanning tree problem ("MST").
3. physical design of computers

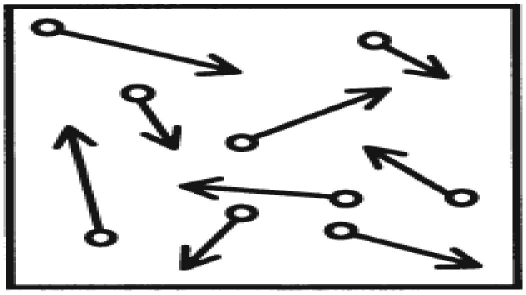
6.2.1 ANNEALING PROCESS

If a metal is cooled so fast, final state of a metal is amorphous (glass-like).
If a metal is cooled very slowly, the final state of a metal is a ground state and makes crystal

The process to heat up a metal and to cool it down slowly IS referred to as "annealing“, "quenching" or "hardening".

1. When metal is heated up in a melting pot, its internal thermal energy increases, and its phase turn to be a liquid phase.

State or internal energy of a metal is determined stochastically, according to the molecule's behavior.
Internal energy is large when its temperature is satisfactory high.



(a) High temperature

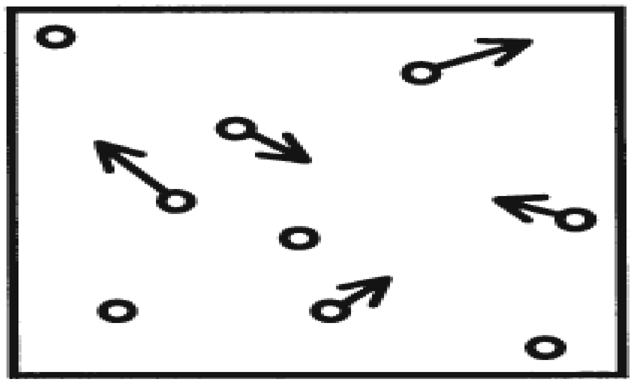
2. From this high temperature state, if temperature of a metal is cooled down slowly, its internal thermal energy decreases slowly although it sometimes increases ruled by the Boltzmann's probability,

$$\exp(-\Delta E / k_B T)$$

where, ΔE : internal energy difference between two different states,

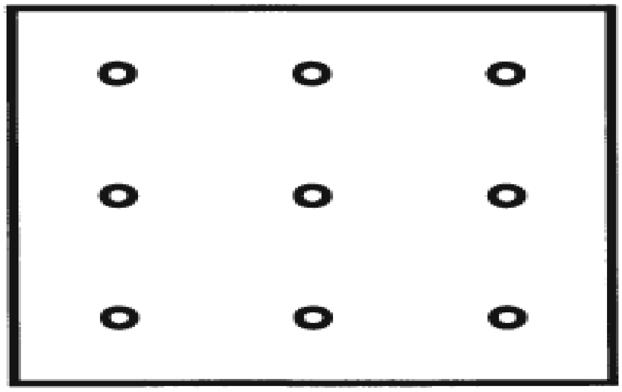
T : absolute temperature

k_B :Boltzmann constant



(b) Low temperature

3. Since a cooling process of metal is ruled by stochastic thermal dynamics, the final state (positions) of molecules is determined randomly according to the behavior of the molecules or its cooling speed.



(c) Frozen

6.2.2 DIGITAL SIMULATION OF ANNEALING PROCESS

Annealing process can be simulated by a digital computer if we consider that the process is a behavior of a set of many discrete particles. By using this concept, Metoropolis has succeeded to develop an algorithm to simulate the stochastic process which can realize the thermal equilibrium at some constant temperature *T*. To develop the algorithm, he substitutes a set of molecule's movement by a series of a small deviation of each molecule. The process is simulated by a Markov chain of states determined by deviations of discrete particles.

6.2.3 Simulated Annealing Algorithm

Kirkpatrik showed that Metoropolis's algorithm can be applied to the functional optimization by corresponding parameters of the annealing process to parameters of the optimization problem as shown in Table I .

Table 1 Relationship between thermal dynamics and functional optimization

Thermal dynamics	***Functional optimization***
State	Solution
Energy	Cost or Objective function
State transition	Neighborhood solution
Temperature (temperature)	Control parameter
Freezing point	Heuristic solution

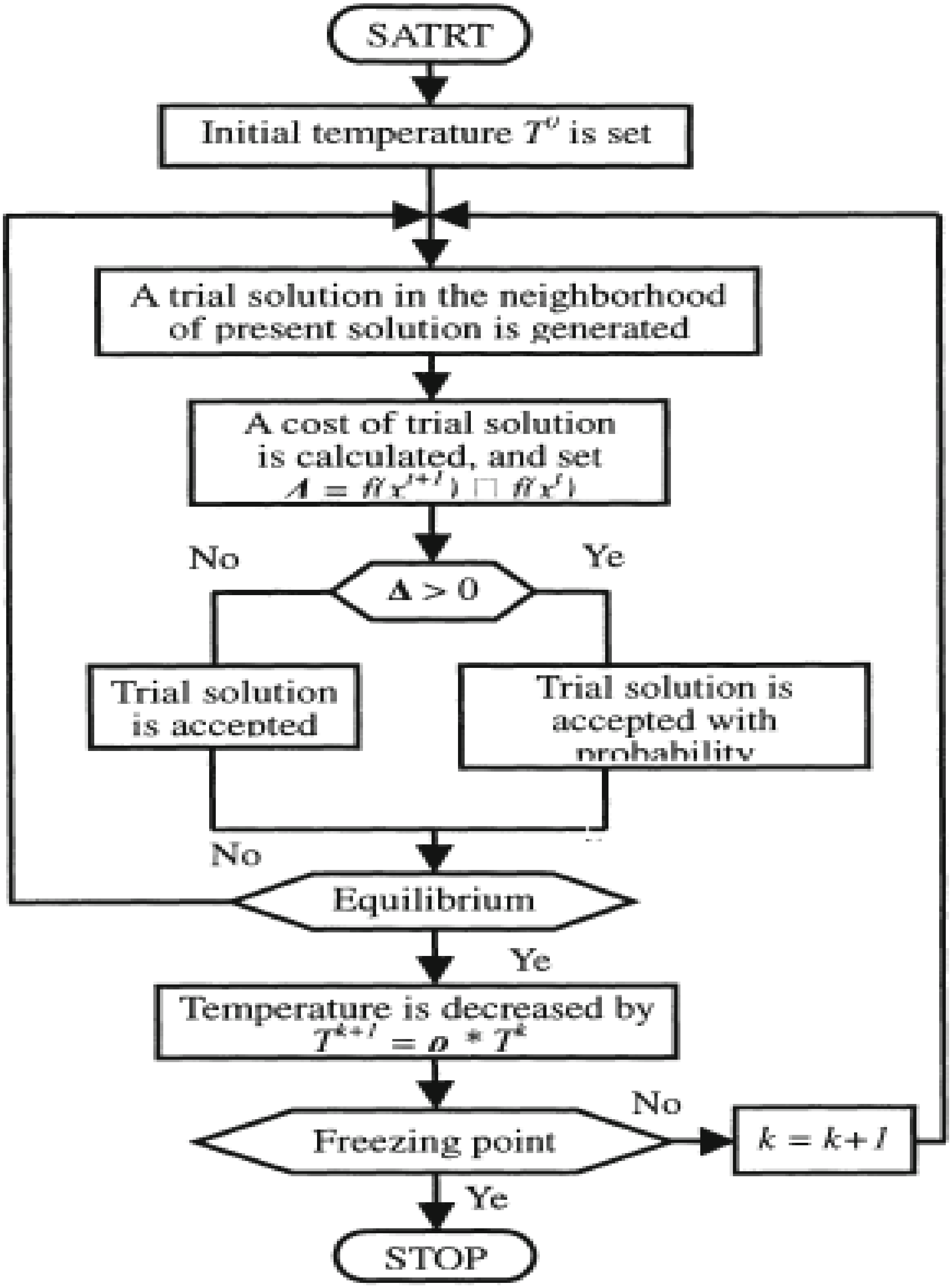

SATRT
Initial temperature T^0 is set
A trial solution in the neighborhood of present solution is generated
A cost of trial solution is calculated, and set
$\Delta = f(x^{j+1}) \square f(x^j)$
No
Ye
$\Delta > 0$
Trial solution is accepted
Trial solution is accepted with probability
No
Equilibrium
Ye
Temperature is decreased by
$T^{k+1} = \alpha * T^k$
No
Freezing point
$k = k+1$
Ye
STOP

Fig. 2 Algorithm of simulated annealing

1. firstly, initial solution and initial temperature are selected.
2. Secondly, a trial solution is generated in the neighborhood of the current solution.
3. Then, a difference of the objective functions between current(t) and trial(t+l) solutions

$$\Delta = f(X^{t+1}) - f(X^{t})$$

is calculated.
4. If Δ <0, then trial solution is accepted. If Δ >0, then trial solution is accepted with probability exp(- Δ *I* T^k)
exp(- Δ *I* T^k) is known as the **acceptance probability function**
5. In a real computer implementation, the solution is accepted when **exp(- Δ *I* T^k) >r** holds. Where, *r* is a random number in the range 0 to 1.
6. Repeat step 2 to 5 until the thermal equilibrium of the current temperature reaches.
7. In a real implementation, thermal equilibrium is estimated when a predetermined large number (several hundreds or thousands) of trial solutions are evaluated.
When equilibrium reaches, the temperature is decreased by the following equation
Where, *p* is taken as 0.80 to 0.99.
8. **Repeat steps from 2 to 7** with the new temperature until freezing point reaches.
9. If the freezing point reaches, the algorithm stops.

6.2.3 Cooling Schedule

The cooling schedule severely affects the solution efficiency and the quality of the solution. In applying a simulated annealing algorithm to an optimization problem, it is important to decrease a temperature satisfactory slowly to obtain a qualified solution.

1.

$$T > c / \log k$$

Where, k means an iteration number of temperature decrease, and c is a constant independent to temperature T.

Takes an infinitive computation burden.

2.

$$T^{k+1} = T^{k} /(1 + \beta T^{k})$$

Where, β is a very smalI constant. In this cooling schedule, temperature decrease is so slow, but a trial solution in one temperature is limited to one time.

6.3 SA IN SCHEDULING

Scheduling problems are combinatorial problems and as such are well suited for SA applications. SA has been applied to single-machine, flow-shop and job-shop problems.

6.3.1 *Single machine scheduling*

In a single machine weighted tardiness problem, the objective is to

$$\min \sum_{i=1}^{n} w_i T_i$$

where T_i is the tardiness of job i and w_i is the weight assigned to job i; there are no constraints for this problem.

The temperature, Z for the k + lth iteration was computed by

$$Z_{k+1} = \frac{Z_k}{(1 + \beta Z_k)}$$

The initial and final Z values are set to 1.000 and 0.334, respectively. The value of β depends on the number of iterations (10,000 for n = 50 and 20,000 for n = 100).

The starting job sequence is chosen at random and the neighborhood solutions are generated by interchanging
two randomly selected jobs in the sequence.

6.3.2 *Flowshop scheduling*

In the flowshop scheduling problem, there are n jobs to be processed on m machines; all jobs must visit the machines in the same predetermined order 1, 2 . . . . . m.

The objective is to find a sequence of jobs that minimizes the makespan (maximum completion time) under the constraint that a job can start on any machine only when that machine is available and the job has finished processing on the previous machine.

The initial value of Z(temperature) is computed by

$$Z_1 = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{p_{ij}}{(5mn)}$$

where p_{ij} is the processing time for job i on machine j, m is the number of machines and n is the number of jobs. The final temperature for the k th iteration is

$$Z_k = 1$$

The neighborhoods considered (interchange or shift neighborhood) and on the search technique (random or ordered) used.

In the interchange method, two randomly selected jobs in the sequence are interchanged.

In the shift neighborhood method, two adjacent jobs are interchanged.

6.3.3 *Jobshop scheduling*

A set of jobs J and a set of machines M are given. Each job consists of a chain of operations O, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. Each machine can perform at most one operation at a time. A schedule is an allocation of the operations to time intervals on the machines. The problem is to find the minimum length schedule.

The neighborhood solutions are generated by *transitions*. A transition is generated, by choosing two successive operations on some machine k, and switching their processing order on the machine. The probability of accepting an increase is given by $exp(-(C(j)-C(i))/c$, where *C(i)* is the cost associated with the ith configuration and c is the control parameter which is gradually decreased.

6.4 SA in solving Travelling salesman problem.

Definitions:

State (s): a particular tour through the set of given cities or points

Neighbor State (′): state obtained by randomly switching the order of two cities

Cost Function (c): determines the total cost of a state

Relative Change in Cost (δ): the relative change in cost c between s and s′

Cooling Constant (β): the rate at which the Temperature is lowered each time a new solution is found

Acceptance Probability Function (P): determines the probability of moving to a more costly state

n = number of cities or points

T_0 = Initial Temperature

T_k = the Temperature at the k^{th} instance of accepting a new solution state

$T^{k+1} = \beta T^k$, where β is some constant between 0 and 1

$$P(\delta, T_k) = \begin{cases} e^{\left(\frac{-\delta}{T_k}\right)}, & \delta > 0 \\ 1, & \delta \leq 0 \end{cases} \quad for\ T_k > 0.$$

1) Choose a random state s and define T_0 and β
2) Create a new state s' by randomly swapping two cities in s
3) Compute $\delta = \frac{C(s') - C(s)}{C(s)}$
    a. If $\delta \leq 0$, then $s = s'$
    b. If $\delta > 0$, then assign $s = s'$ with probability $P(\delta, T_k)$
        i. Compute $T_{k+1} = \beta T_k$ and increment k
4) Repeat steps 2 and 3 keeping track of the best solution until stopping conditions are met

Stopping Conditions: for simulated annealing in TSP

If the algorithm is stopped too soon, the approximation won’t be as close to the global optimum. If it isn’t stopped soon enough, wasted time and calculations are spent with little to no benefit.

6.5 Application of GA in solving sequencing and scheduling problems

Figure 1 shows the framework of the genetic algorithm that is employed for the job scheduling problem. Parent schedules with high evaluations are selected and remaining chromosomes are discarded by the reproduction operator. These parent schedules then generate new offspring chromosomes via heuristic crossover and local improvement. Finally, the evaluation process computes the fitness values of newly generated
chromosomes for the reproduction in the next generation. This procedure is repeated and the population of schedules evolves generation by generation.

Algorithm

Step 0 (Selection of the first gene)
Let *i=l* and compare the first genes of the two parent chromosomes. Choose the one with the longer processing time, i.e. higher job number, as the first gene al of the child.

Step 1 (Stopping, Order of Ascending/Descending)
If i = n+1, then stop. The crossover is complete. If i ≥ k for which a_k = 1, then go to Step 3. Otherwise, go to Step 2.

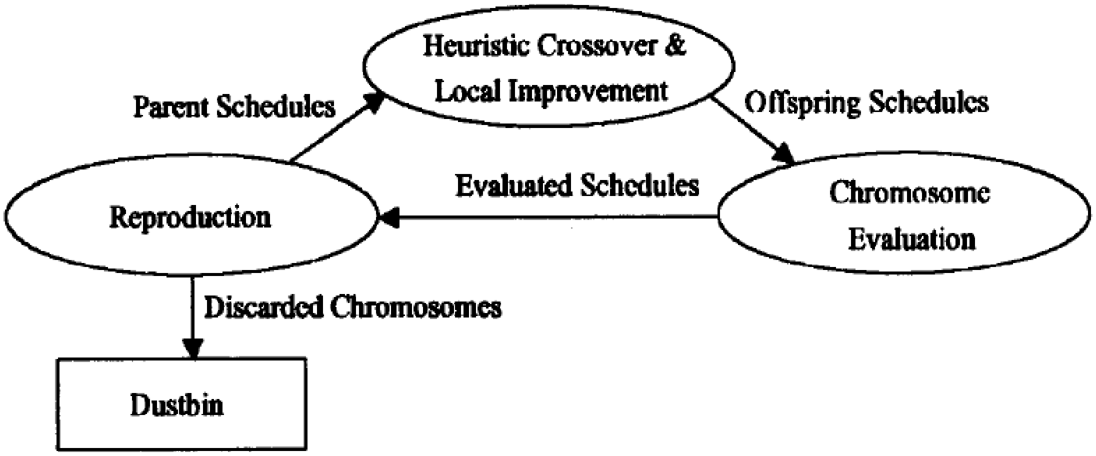


Figure 1. The Framework of Genetic Algorithm for the Job Sequencing Problem.

Step 2 (Descending Order)

Let i = *i+1.* From each parent select the very next gene to a_{i-l} of the child chromosome. If a_{i-l} is the last gene of a parent, then select the first gene of the parent. Let a_i be the gene determined as follows and go to Step 1.

2.1) If both of the genes already exist in the new chromosome, then choose the highest gene which is lower than

a_{i-l}.

2.2) If one constructs a cycle and the other does not, then choose the latter as a_i.

2.3) If neither of the genes constructs a cycle and there exist genes lower than ai-l, then choose the higher one among them as *ai.*

2.4) If neither of the genes constructs a cycle and neither of them is lower than ai-l, then choose the lower one as *ai*.

Step 3 (Ascending order)

Let i = i+1. From each parent select the very next gene to ai-l of the child chromosome. If ai-l is the last gene of a parent, then select the first gene of the parent. Let ai be the gene determined as follows and go to Step 1.

3.1) If both of the genes construct cycles, then among the remaining genes, choose the lower gene which is higher than ai-l' If there are no such genes, then select the lowest as ai.

3.2) If one constructs a cycle and the other does not, then choose the latter as ai.

3.3) If neither of the genes constructs a cycle and there exist genes higher than ai-l, then choose the lower one among them as ai.

3.4) If neither of the genes constructs a cycle and neither of them is higher than ai-l, then choose the lower one as ai.

Example 1 The following example illustrates the algorithm in generating a child chromosome.

Chromosome 1:	9 8 4 5 6 7 1 3 2 10
Chromosome 2:	8 6 1 2 3 10 5 9 4 7
Child :	9 8 6 1 2 3 10 5 7 4

The first gene of a child is determined by the first job with longer processing time of the two parents. Thus, job 9 is selected as the first gene. Then job 8 of chromosome 1 and job 4 of chromosome 2 are compared and the job with longer processing time is selected to follow job 9 by Step 2.3. Thus, job 8 follows job 9. The procedure then continues by Step 2.3 and job 6 and job 1 of chromosome 2 follow. Since job 1 with the shortest processing time is selected, the algorithm now proceeds in ascending order. Jobs 2, 3 and 10 of chromosome 2 follow job 1 in the child chromosome by Step 3.3. Then by Step 3.2 Job 10 is followed by job 5 of chromosome 2, since job 9 of chromosome 1 constructs a cycle. Job 7 is then selected to follow job 5 by Step 3.1 since both job 6 and 9 of the two parents construct cycles. Finally, job 4 constitutes the last gene of the child chromosome.

ALGORITHM FOR SOLVING JOB SHOP SCHEDULING PROBLEMS

Scheduling is one of the critical issues when planning and managing manufacturing processes and represents a significant problem for many companies . One of the most difficult problems in this area is the job shop scheduling problem (JSSP), which is well known as one of the most difficult NP-hard problems.

The problem formulation

The flexible job-shop scheduling problem can be formulated as follows: There is a set of NP products *P* = {*P*1, *P*2,…, *Pi*,…, *PN*}. To complete each of these products we need to complete corresponding job from a set of *K* jobs *J* = {*J*1, *J*2,…, *Jj*,…, *JK*}. Each job *Jj* consists of a predetermined sequence of operations. Each operation requires one machine *M* selected from a set of available machines *M* = {*M*1, *M*2,…, *Mk*,…, *MM*}.

The aim is to optimize production by using a genetic algorithm or in other words to find a schedule of the operations on the machines with the shortest makespan taking into account the precedence statements:

1. All machines are available at time 0

2. All jobs are released at time 0

3. Every job is a chain of operations and order of operations has to be maintained

4. All operations of a given job have to be processed in a given order.

5. Two operations of a job cannot be processed at the same time;

6. Each machine can process at most one operation at a time

7. Once processing starts on a given machine, it must complete on that particular machine without any interruption.

8. Each operation has a fixed duration.

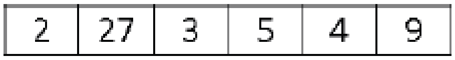
9. Each machine cannot process more than one operation at a time

10. All machines are available at zero time and machine efficiency is 100 %

11. There is no break time.

1 Chromosome representation and decoding

Chromosome representation has three components:
Product Code Component (PCC),
Series Size Component (SSC) and
Machine Code Selection Component (MCSC).



2	27	3	5	4	9

Figure 1 Chromosome representation and decoding (example)

As can be seen from example in Fig. 1 the first element in array (PCC) represents product code (product code value is "2"). Value of the second element (SSC) is 27 – that means that there are 27 products in the series. Next four elements in the array are used for description of operations and machines (MCSC).

2 Population initialization

The initial population in algorithm is produced randomly.

3 Selection and crossover

4. Mutation

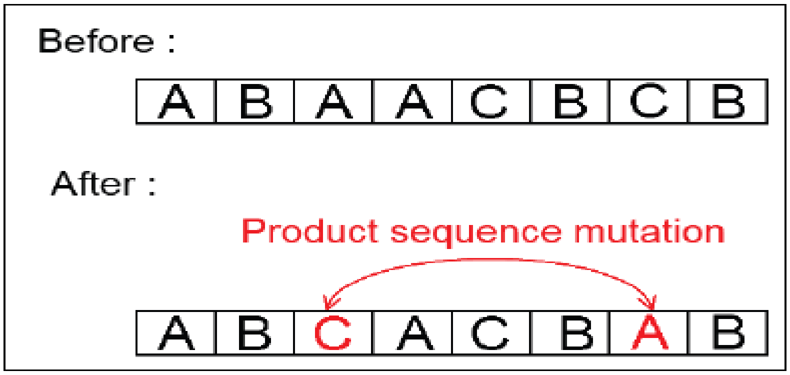


Figure 3 Product sequence mutation (example)

Since in some operations we can use more than one machine in this type of mutation we alter genes that represent machines, or in other words machine for a certain operation is altered. Newly assigned machine must be included in the machine set of the corresponding operation. All individuals in the population have the same chance for mutation except the best individual which is protected from alteration of its genes. Machine sequence mutation is described as follows:

1. Select one individual form population
2. Select one product
3. Select one operation
4. Assign new machine for selected operation.

5. Fitness evaluation

The scheduling problem is an optimization problem where the ultimate objective is to reduce the total cost of production by reducing the time of production**.**

6.6 Application of GA in solving Travelling salesman problem

Genetic algorithm is used to solve Travelling Salesman Problem. Genetic algorithm is a technique used for

estimating computer models based on methods adapted from the field of genetics in biology. To use this technique, one encodes possible model behaviors into "genes". After each generation, the current models are rated and allowed to mate and breed based on their fitness. In the process of mating, the genes are exchanged, crossovers and mutations can occur. The current population is discarded and its offspring forms the next
generation.

Algorithm

1. Initialization: Generate N random candidate routes and calculate fitness value for each route.

2. Repeat following steps Number of iteration times:

a) Selection: Select two best candidate routes.

b) Reproduction: Reproduce two routes from the best routes.

c) Generate new population: Replace the two worst routes with the new routes.

3. Return the best result

APPENDIX 1

CONTENT BEYOND THE SYLLABUS

1. Why do we need better optimization Algorithms?

To train a neural network model, we must define a loss function in order to measure the difference between our model predictions and the label that we want to predict. What we are looking for is a certain set of weights, with which the neural network can make an accurate prediction, which automatically leads to a lower value of the loss function.

I think you must know by now, that the mathematical method behind it is called gradient descent.

$$\theta_j \leftarrow \theta_j - \epsilon \nabla_{\theta_j} \mathcal{L}(\theta)$$

Eq. 1 Gradient Descent for parameters θ with loss function L.

In this technique (Eq.1), we must calculate the gradient of the loss function L with respect to the weights (or parameters θ) that we want to improve. Subsequently, the weights/parameters are updated in the direction of the negative direction of the gradient.

By periodically applying the gradient descent to the weights, we will eventually arrive at the optimal weights that minimize the loss function and allow the neural network to make better predictions.

So far the theory.

Do not get me wrong, gradient descent is still a powerful technique. In practice, however, this technique may encounter certain problems during training that can slow down the learning process or, in the worst case, even prevent the algorithm from finding the optimal weights

These problems were on the one hand saddle points and local minima of the loss function, where the loss function becomes flat and the gradient goes to zero:

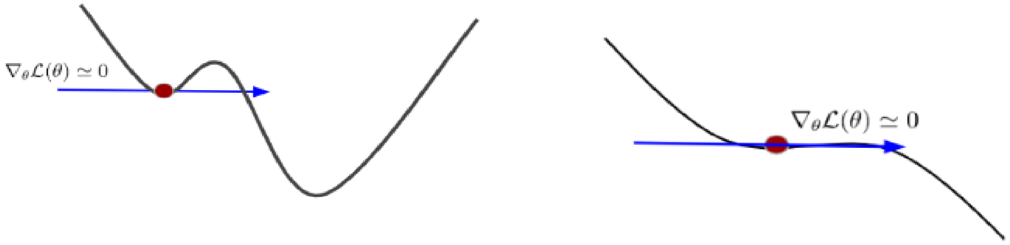

$\nabla_\theta \mathcal{L}(\theta) \simeq 0$
$\nabla_\theta \mathcal{L}(\theta) \simeq 0$

Fig. 1 Saddle Points and Local Minima

A gradient near zero does not improve the weight parameters and prevents the entire learning process.

On the other hand, even if we have gradients that are not close to zero, the values of these gradients calculated for different data samples from the training set may vary in value and direction. We say that the gradients are noisy or have a lot of variances. This leads to a zigzag movement towards the optimal weights and can make learning much slower:

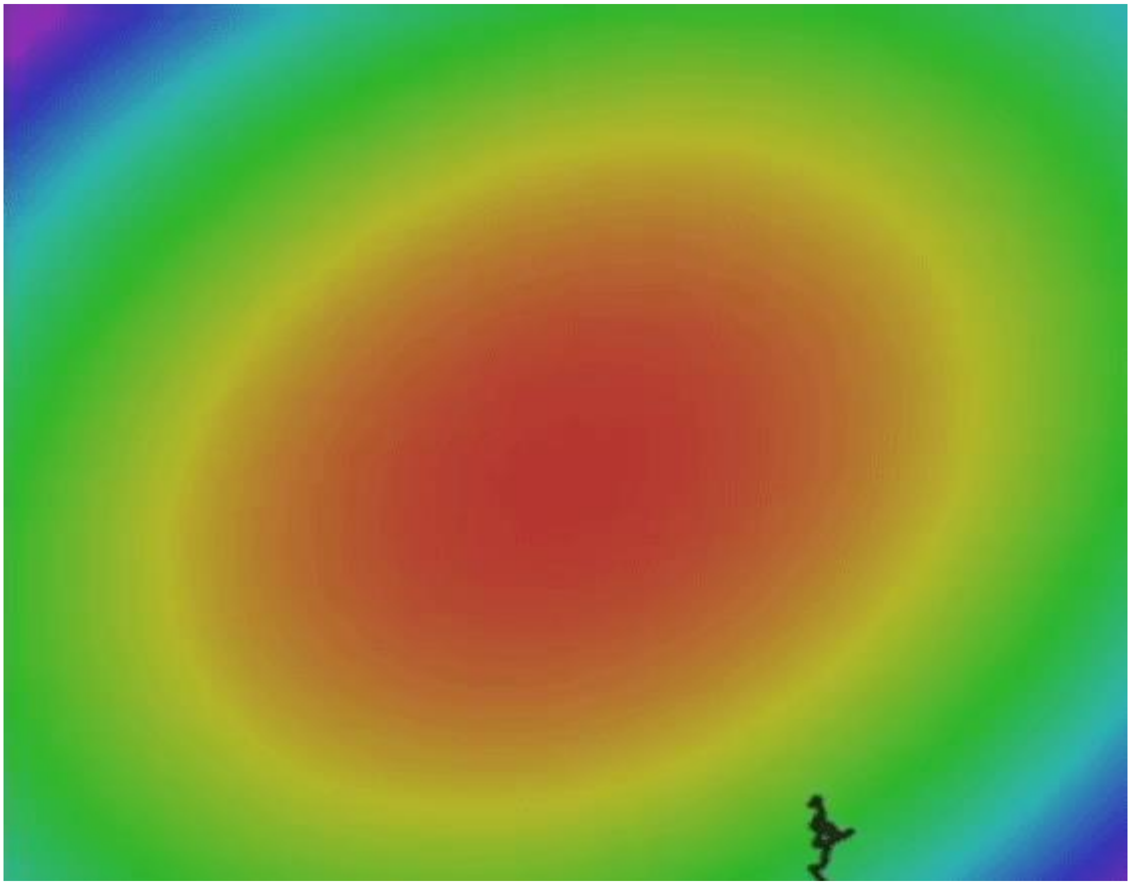
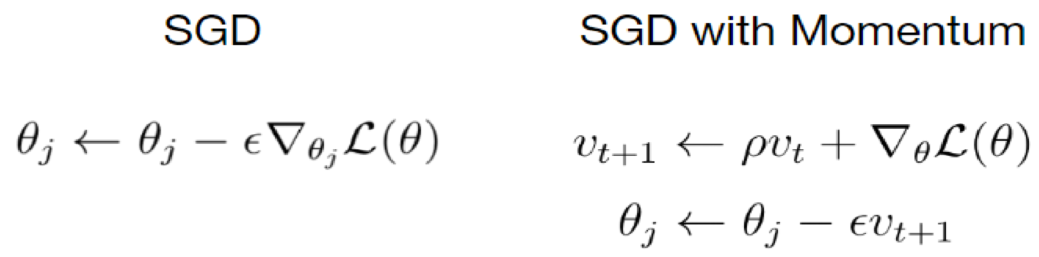


Fig. 3 Example of zig-zag movements of noisy gradients.

In the following article, we are going to learn about more sophisticated gradient descent algorithms. All of these algorithms are based on the regular gradient descent optimization that we have come to know so far. But we can extend this regular approach for the weight optimization by some mathematical tricks to build even more effective optimization algorithms that allow our neural network to adequately handle these problems, thereby learning faster and to achieve a better performance

2. Stochastic Gradient Descent with Momentum

The first of the sophisticated algorithms I want to present you is called stochastic gradient descent with momentum.



SGD

$$\theta_j \leftarrow \theta_j - \epsilon \nabla_{\theta_j} \mathcal{L}(\theta)$$

SGD with Momentum

$$v_{t+1} \leftarrow \rho v_t + \nabla_{\theta} \mathcal{L}(\theta)$$

$$\theta_j \leftarrow \theta_j - \epsilon v_{t+1}$$

Eq. 2 Equations for stochastic gradient descent with momentum.

On the left side in Eq. 2, you can see the equation for the weight updates according to the regular stochastic stochastic gradient descent. The equation on the right shows the rule for the weight updates according to the SGD with momentum. The momentum appears as an additional term ρ times v that is added to the regular update rule.

Intuitively speaking, by adding this momentum term we let our gradient to build up a kind of velocity v during training. The velocity is the running sum of gradients weighted by ρ.

ρ can be considered as friction that slows down the velocity a little bit. In general, you can see that the velocity builds up over time. By using the momentum term saddle points and local minima become less dangerous for the gradient. Because step sizes towards the global minimum now don’t depend only on the gradient of the loss function at the current point, but also on the velocity that has built up over time.

In other words, we are moving more towards the direction of velocity than towards the gradient at a certain point.

If you want to have a physical representation of the stochastic gradient descent with momentum think about a ball that rolls down a hill and builds up velocity over time. If this ball reaches some obstacles on its way, such as a hole or a flat ground with no downward slope, the velocity v would give the ball enough power to roll over these obstacles. In this case, the flat ground and the hole represent saddle points or local minima of a loss function.

In the following video (Fig. 4), I want to show you a direct comparison of regular stochastic gradient descent and stochastic gradient descent with momentum term. Both algorithms are trying to reach the global minimum of the loss function which lives in a 3D space. Please note how the momentum term makes the gradients to have less variance and fewer zig-zags movements.

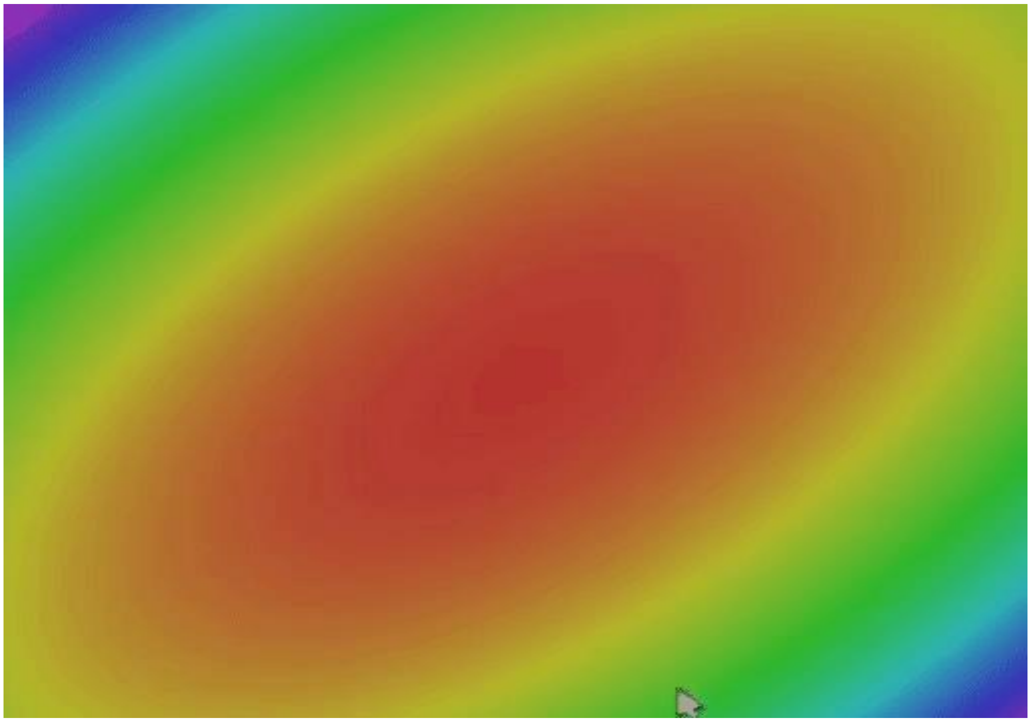


Fig. 4 SGD vs. SGD with Momentum

In general, the momentum term makes converges towards optimal weights more stable and faster.

3. AdaGrad

Another optimization strategy is called AdaGrad. The idea is that you keep the running sum of squared gradients during optimization. In this case, we have no momentum term, but an expression g that is the sum of the squared gradients.

SGD with Momentum

$$v_{t+1} \leftarrow \rho v_t + \nabla_\theta \mathcal{L}(\theta)$$

$$\theta_j \leftarrow \theta_j - \epsilon v_{t+1}$$

AdaGrad

$$g_0 = 0$$

$$g_{t+1} \leftarrow g_t + \nabla_\theta \mathcal{L}(\theta)^2$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1e^{-5}}$$

Eq. 3 Parameter update rule for AdaGrad.

When we update a weight parameter, we divide the current gradient by the root of that term g. To explain the intuition behind AdaGrad, imagine a loss function in a two-dimensional space in which the gradient of the loss function in one direction is very small and very high in the other direction.

Summing up the gradients along the axis where the gradients are small causes the squared sum of these gradients to become even smaller. If during the update step, we divide the current gradient by a very small sum of squared gradients g, the result of that division becomes very high and vice versa for the other axis with high gradient values.

As a result, we force the algorithm to make updates in any direction with the same proportions.

This means that we accelerate the update process along the axis with small gradients by increasing the gradient along that axis. On the other hand, the updates along the axis with the large gradient slow down a bit.

However, there is a problem with this optimization algorithm. Imagine what would happen to the sum of the squared gradients when training takes a long time. Over time, this term would get bigger. If the current gradient is divided by this large number, the update step for the weights becomes very small. It is as if we were using very low learning that becomes even lower the longer the training goes. In the worst case, we would get stuck with AdaGrad and the training would go on forever.

4. RMSProp

There is a slight variation of AdaGrad called RMSProp that addresses the problem that AdaGrad has. With RMSProp we still keep the running sum of squared gradients but instead of letting that sum grow continuously over the period of training we let that sum actually decay.

AdaGrad

$$g_0 = 0$$

$$g_{t+1} \leftarrow g_t + \nabla_\theta \mathcal{L}(\theta)^2$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1e^{-5}}$$

RMS Prop

$$g_0 = 0\,, \alpha \simeq 0.9$$

$$g_{t+1} \leftarrow \alpha \cdot g_t + (1-\alpha) \nabla_\theta \mathcal{L}(\theta)^2$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1e^{-5}}$$

Eq. 4 Update rule for RMS Prop.

In RMSProp we multiply the sum of squared gradients by a decay rate α and add the current gradient weighted by (1- α). The update step in the case of RMSProp looks exactly the same as in AdaGrad where we divide the current gradient by the sum of squared gradients to have this nice property of accelerating the movement along the one dimension and slowing down the movement along the other dimension.

Let's see how RMSProp is doing in comparison with SGD and SGD with momentum in finding the optimal weights.

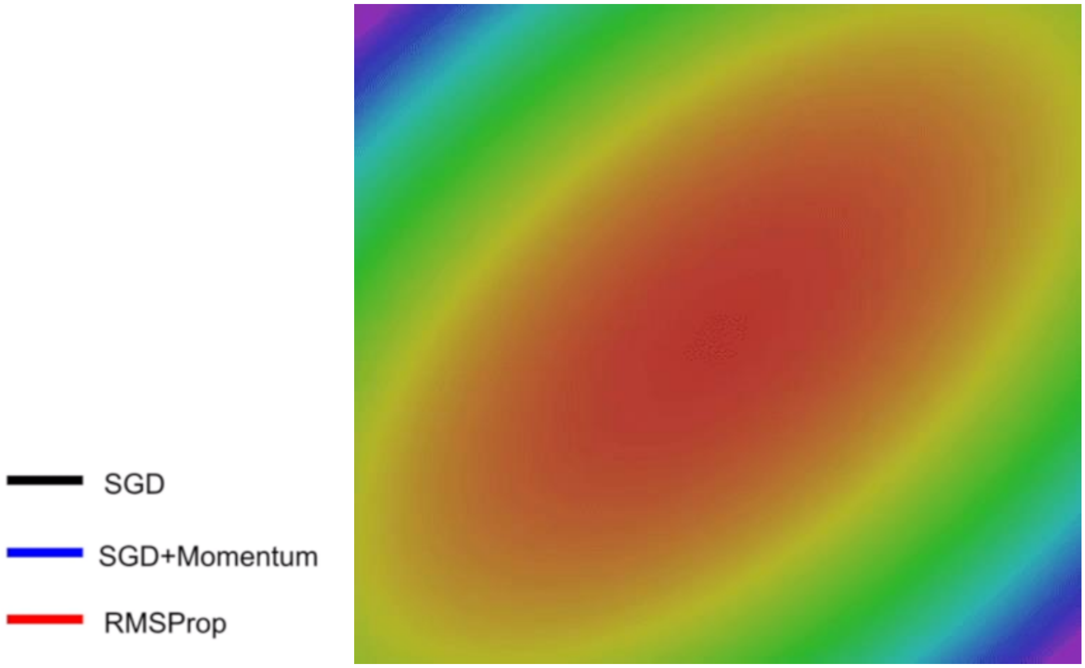


Fig. 5 SGD vs. SGD with Momentum vs. RMS Prop

Although SGD with momentum is able to find the global minimum faster, this algorithm takes a much longer path, that could be dangerous. Because a longer path means more possible saddle points and local minima. RMSProp, on the other hand, goes straight towards the global minimum of the loss function without taking a detour.

5. Adam Optimizer

So far we have used the moment term to build up the velocity of the gradient to update the weight parameter towards the direction of that velocity. In the case of AdaGrad and RMSProp, we used the sum of the squared gradients to scale the current gradient, so we could do weight updates with the same ratio in each dimension.

These two methods seemed pretty good ideas. Why do not we just take the best of both worlds and combine these ideas into a single algorithm?

This is the exact concept behind the final optimization algorithm called Adam, which I would like to introduce to you.

The main part of the algorithm consists of the following three equations. These equations may seem overwhelming at first, but if you look closely, you’ll see some familiarity with previous optimization algorithms.

$$m_0 = 0, v_0 = 0$$

$$\boxed{m_{t+1} \leftarrow \beta_1 m_t + (1 - \beta_1) \nabla_\theta \mathcal{L}(\theta)}$$

Momentum

$$\boxed{v_{t+1} \leftarrow \beta_2 v_t + (1 - \beta_2) \nabla_\theta \mathcal{L}(\theta)^2}$$

RMS Prop

$$\boxed{\theta_j \leftarrow \theta_j - \frac{\epsilon}{\sqrt{v_{t+1} + 1e^{-5}}} m_{t+1}}$$

RMS Prop + Momentum

Eq. 5 Parameter update rule for Adam Optimizer

The first equation looks a bit like the SGD with momentum. In the case, the term would be the velocity and the friction term. In the case of Adam, we call the first momentum and is just a hyperparameter.

The difference to SGD with momentum, however, is the factor (1- β1), which is multiplied with the current gradient.

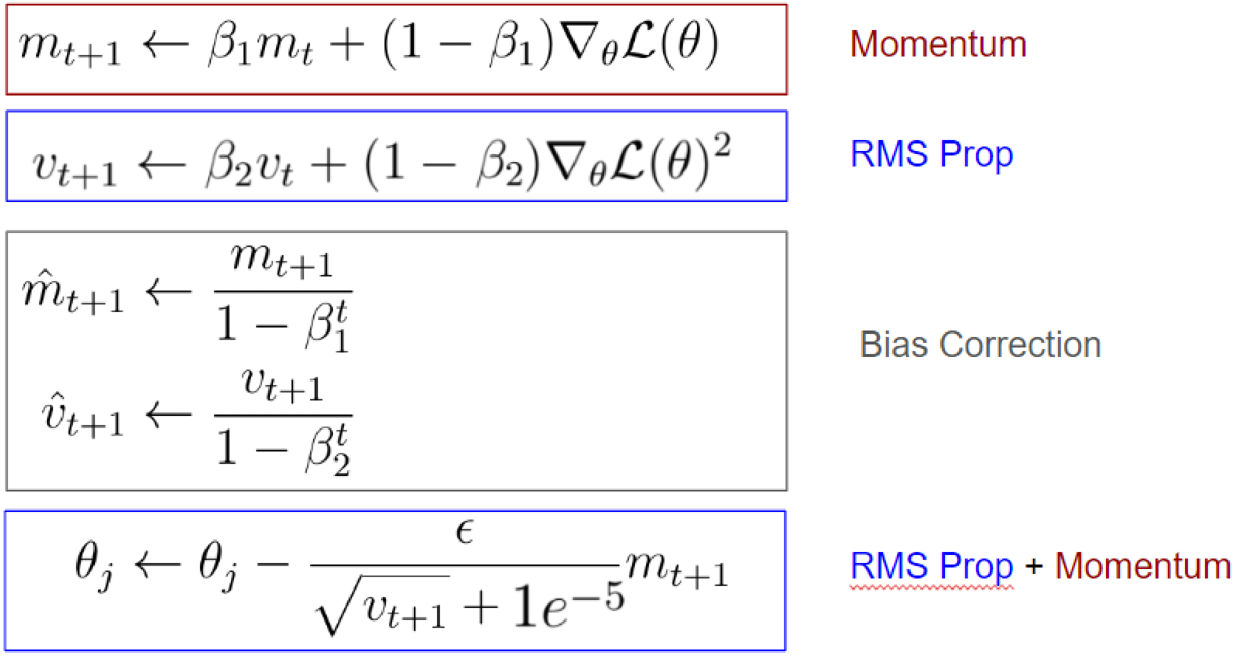
The second part of the equations, on the other hand, can be regarded as RMSProp, in which we are keeping the running sum of squared gradients. Also, in this case, there is the factor (1-β2) which is multiplied with the squared gradient.

The term in the equation is called the second momentum and is also just a hyperparameter. The final update equation can be seen as a combination of RMSProp and SGD with Momentum.

So far, Adam has integrated the nice features of the two previous optimization algorithms, but here's a little problem, and that's the question of what happens in the beginning.

At the very first time step, the first and second momentum terms are initialized to zero. After the first update of the second momentum, this term is still very close to zero. When we update the weight parameters in the last equation, we divide by a very small second momentum term v, resulting in a very large first update step.

This first very large update step is not the result of the geometry of the problem, but it is an artifact of the fact that we have initialized the first and second momentum to zero. To solve the problems of large first update steps, Adam includes a correction clause:



$$m_{t+1} \leftarrow \beta_1 m_t + (1 - \beta_1)\nabla_\theta \mathcal{L}(\theta)$$

Momentum

$$v_{t+1} \leftarrow \beta_2 v_t + (1 - \beta_2)\nabla_\theta \mathcal{L}(\theta)^2$$

RMS Prop

$$\hat{m}_{t+1} \leftarrow \frac{m_{t+1}}{1 - \beta_1^t}$$

$$\hat{v}_{t+1} \leftarrow \frac{v_{t+1}}{1 - \beta_2^t}$$

Bias Correction

$$\theta_j \leftarrow \theta_j - \frac{\epsilon}{\sqrt{v_{t+1}} + 1e^{-5}} m_{t+1}$$

RMS Prop + Momentum

Eq. 6 Bias Correction for Adam Optimizer

You can see that after the first update of the first and second momentum and we make an unbiased estimate of these momentums by taking into account the current time step. These correction terms make the values of the first and second momentum to be higher in the beginning than in the case without the bias correction.

As a result, the first update step of the neural network parameters does not get that large and we don’t mess up our training in the beginning. The additional bias corrections give us the full form of Adam Optimizer.

Now, let us compare all algorithms with each other in terms of finding the global minimum of the loss function:

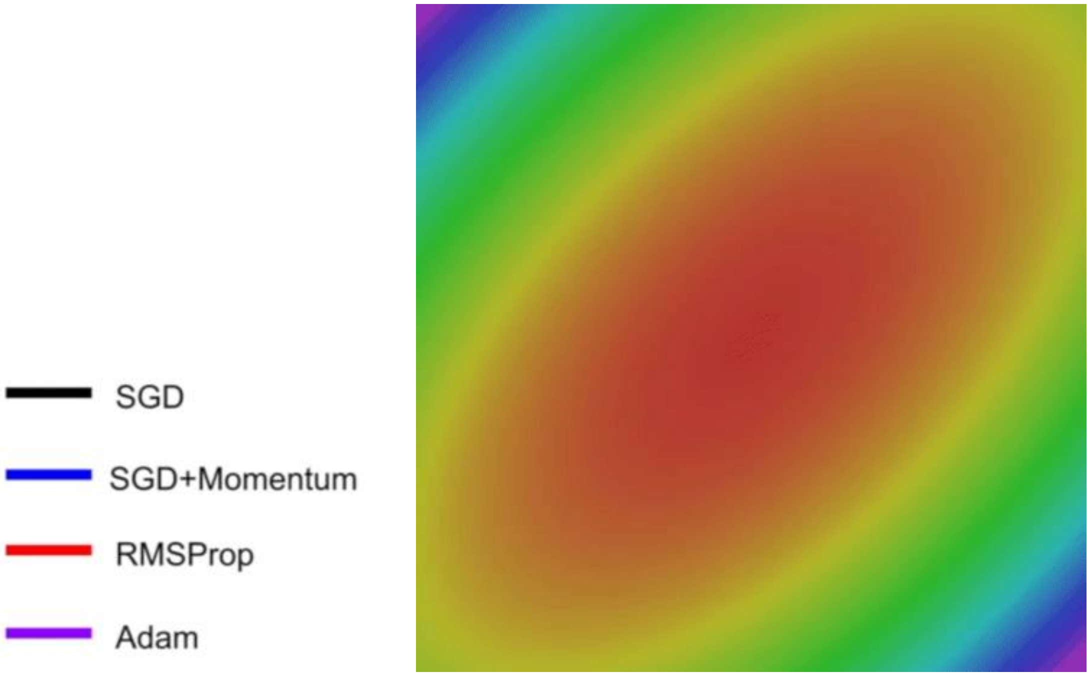


Fig. 6 Comparison of all optimization algorithms.

6. What is the best Optimization Algorithm for Deep Learning?

Finally, we can discuss the question of what the best gradient descent algorithm is.

In general, a normal gradient descent algorithm is more than adequate for simpler tasks. If you are not satisfied with the accuracy of your model you can try out RMSprop or add a momentum term to your gradient descent algorithms.

But in my experience the best optimization algorithm for neural networks out there is Adam. This optimization algorithm works very well for almost any deep learning problem you will ever encounter. Especially if you set the hyperparameters to the following values:

- β1=0.9
- β2=0.999
- Learning rate = 0.001–0.0001

… this would be a very good starting point for any problem and virtually every type of neural network architecture I’ve ever worked with.

That’s why Adam Optimizer is my default optimization algorithm for every problem I want to solve. Only in very few cases do I switch to other optimization algorithms that I introduced earlier.

In this sense, I recommend that you always start with the Adam Optimizer, regardless of the architecture of the neural network of the problem domain you are dealing with.