Fundamentals of Optimization Techniques with Algorithms

Sukanta Nayak

Fundamentals of Optimization Techniques With Algorithms

This page intentionally left blank

Fundamentals of Optimization Techniques With Algorithms

SUKANTA NAYAK
Department of Mathematics Amrita School of Engineering, Amrita Vishwa Vidyapeetham, Coimbatore, India

ELSEVIER

ACADEMIC PRESS
An imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1650, San Diego, CA 92101, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom

Copyright © 2020 Elsevier Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

Notices
Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

ISBN: 978-0-12-821126-7

For Information on all Academic Press publications
visit our website at https://www.elsevier.com/books-and-journals

Publisher: Matthew Deans
Acquisitions Editor: Glyn Jones
Editorial Project Manager: Fernanda A. Oliveira
Production Project Manager: Nirmala Arumugam
Cover Designer: Greg Harris

Typeset by MPS Limited, Chennai, India

ELSEVIER Book Aid International Working together to grow libraries in developing countries
www.elsevier.com • www.bookaid.org

Dedication

Dedicated to
My Beloved wife
Pallabi

Contents

This page intentionally left blank

Preface

Optimization is the backbone of every system that involves decision-making and optimal strategies. It plays an important role and influences our life directly or indirectly which cannot be avoided or neglected. Optimization is a key concept not only in mathematics, computer science, and operations research, and but also it is essential to the modeling of any system, playing an integral role in the computer-aided design. In recent years, optimization techniques become a considerable part of each system and applicable in a wide spectrum of industries, viz., aerospace, chemical, electrical, electronics, mining, mechanical, information technology, finance, and e-commerce sectors. Therefore, the very need is to easy understanding of the problem and implementation of optimization techniques with advanced computer-added design activity.

As such, the purpose of this textbook is to provide various traditional and advanced optimization techniques along with a variety of example problems, algorithms, and MATLAB codes for the same in a comprehensive manner. This book includes optimization techniques, for linear and nonlinear single variable and multivariable models, as well as multi-objective and advanced optimization techniques. It presents both theoretical and numerical perspectives in a clear and approachable way. To help the reader for applying optimization techniques in practice, the book is devised to detail program codes and computer-aided designs concerning real-world problems. This book exclusively contains both the theoretical and numerical treatment of various optimization techniques. It may serve as the first coursebook for masters, research scholars, and academicians as well as a textbook for undergraduate students of mathematics, science, and various engineering branches, viz., mechanical, electrical, electronics, computer science, and aerospace. Further, the second half of the book will be greatly helpful to the industry people and researchers.

This book comprises ten chapters. Chapter 1 introduces the idea of optimization, its motivation, basic strategy to formulate any optimization problems, and various optimization techniques to implement in different optimization problems. Chapter 2 presents linear programming type optimization problems and various techniques to solve the same. Techniques such as the simplex method, artificial variable technique, and dual simplex method are described with the flow diagram and algorithms to get the first-hand knowledge to write the codes. For easy illustration, many example problems are also demonstrated. In Chapter 3, optimization techniques for single variable nonlinear problems are discussed. Both the classical approach to get the optimum point and the numerical approach to obtain the optimum point of the unimodal function is included. Various bracketing, region elimination, point estimation, and gradient methods are discussed in detail with the algorithms.

Chapters 4 and 5 comprise multivariable nonlinear optimization problems. Chapter 4 deals with unconstrained multivariable optimization problems. Different types of direct and gradient search algorithms, as well as exact methods, are presented to investigate the optimum solution. Algorithms and flowcharts are provided for easy understanding of the mentioned techniques. Generalized MATLAB codes for the same are given to explore the insights of programming. Whereas, Chapter 5 deals with constrained multivariable optimization problems. These problems are of two types, equality and inequality constrained multivariable optimization problems. Various types of penalty methods, viz., interior and exterior penalty methods, are employed to obtain the optimal solution for multivariable constrained nonlinear optimization problems.

Special types of optimization techniques such as geometric, dynamic, and integer programming are presented in Chapters 6, 7, and 8, respectively. Chapter 6 covers the optimization techniques to investigate geometric programming problems. Here, the objective functions and constraints are having designed variables with a real-valued exponent. Therefore, the constraints and objective function involve posynomials that make a difference from the other optimization problem and corresponding techniques as well. Chapter 7 describes the dynamic programming problems and various strategies to solve optimization problems through dynamic programming techniques and algorithms. In the dynamic programming approach, the optimization problems are divided into simpler subproblems and then each sequentially subproblems are solved by simple optimization techniques. For easy understanding, various example problems of dynamic programming are discussed. Chapter 8 includes linear and non-linear programming problems with integer-valued variables. In real-life problems such as capital budgeting, construction scheduling, batch size, and capacity expansion, etc. integer programming techniques are used to determine the field variables.

Chapter 9 presents a multiobjective optimization problem. It is used in multiple criteria decision-making which deals with optimization problems involving two or more objective function to be optimized simultaneously. In this chapter, the reader will go through various techniques, viz., global criterion method, utility function method, inverted utility function method, bounded objective function method, lexicographic method, and goal programming method to solve the multiobjective optimization problem. Finally, in Chapter 10, the reader can explore and get motivation for various nature-inspired optimization techniques, viz., genetic algorithm, neural network-based optimization, ant colony optimization, and particle swarm optimization. A comprehensive description of these methods along with the working algorithms will provide an easy illustration of various real-life problems.

This page intentionally left blank

Acknowledgments

The author would like to thank all the contributors for both the research and developments done and expected to be done in the future in this silver-lining area of optimization. He has been inspired by the researchers' work presented in the Further Reading Section of this book and greatly acknowledges their efforts.

In the first place, the author is very much thankful to his beloved parents. Then, he wants to dedicate this work to his wife Pallabi Sahoo for her continuous love, support, and source of inspiration at all the time during the preparation of this book.

The author is indebted to his friends and colleagues Dr. Arun Kumar Gupta, Mr. Anil Boity, Dr. Prashanth Maroju, and Dr. Anil Kumar Sharma for their help and support during the crucial phase of preparing this manuscript that making it a memorable experience in his life. These extraordinary lighter moments were not only enjoyable but also helped him reinvigorate the academic prowess to start things afresh.

The author like to express sincere gratitude to the readers who will go through this book and enjoy the beauty of optimization in their life. Finally, he would like to thank the Academic Press for enabling him to publish this book and to the team of the publisher who directly or indirectly provided him help and support throughout this project.

This page intentionally left blank

CHAPTER ONE

Introduction to optimization

Optimization algorithms are essential tools in engineering and scientific design activities. These are quite popular and becoming an integral part of various fields, viz., engineering, science, economics, sociology, and many decision-making problems. We may know the importance of optimization from a simple example of a transportation problem where a consignment goes to its destination from the warehouse. It depends on various modes of transportation and time duration to reach, which decides the cost of transportation. Similarly, optimization algorithms are used in aerospace engineering for designing the purpose of aircraft. Here, the objective is to minimize the overall weight of the aircraft. Hence, the tasks or problems involve optimization (either minimization or maximization) of an objective. The basic procedure of optimization for any problem is shown in Fig. 1.1. The problem is treated as a physical problem, which is provided with necessary information called resources or inputs. By using these inputs, the physical problem is modeled. With the objective of the physical problem and depending on constraints, the problem is formulated. Then, by using optimization techniques, the solutions are investigated, which is called the output. Finally, from the output, the optimal set of solution(s) is obtained. Here, we may make a point that

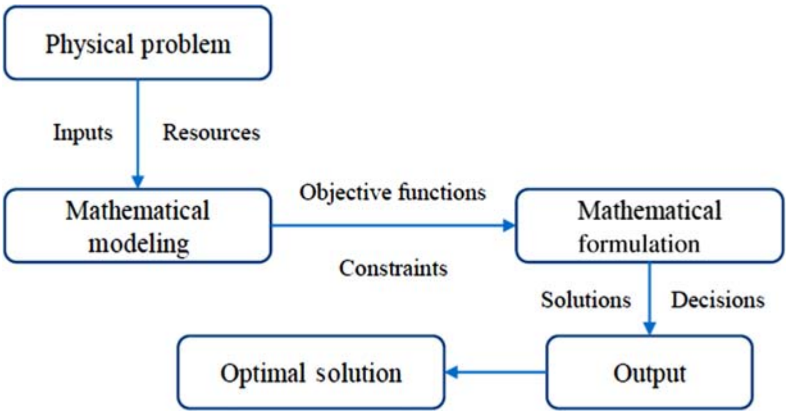


Figure 1.1 Flow chart of modeling physical problem to get an optimal solution.

Fundamentals of Optimization Techniques with Algorithms.
DOI: https://doi.org/10.1016/B978-0-12-821126-7.00001-2
© 2020 Elsevier Inc.
All rights reserved.

optimization is the act of obtaining the best result under given circumstances. In other words, optimization is a process in which we can use the resources in the best possible way.

1.1 Optimal problem formulation

Many industrial and engineering activities involve optimal designs that are achieved by comparing the alternative design solution with previous knowledge. The feasibility of design solutions plays an important role in these activities, and it is given priority. Then, the underlying objective (cost, profit, etc.) estimated for each design solution is computed, and the best design solution is adopted. This is the usual procedure followed due to the limitations of time and resources. But, in many cases, this method is followed simply because of the lack of knowledge of the existing optimization procedures. Each optimization algorithm follows a particular representation of a problem statement and a procedure. As such, a mathematical model of the optimal design problem is discussed through different components in the following sections. Fig. 1.2 shows an outline of the steps involved in an optimal design optimization technique. The first step is to choose the design variables connected with the optimization problem. Then, the formulation of optimization problems involves constraints, objective function, and variable bounds, and so on.

1.1.1 Design variables

The initial step of the optimization problem formulation is choosing design variables, which vary during the optimization process. Generally, the parameters that influence the optimization process are called design variables, and these parameters are highly sensitive to the proper working of the design. However, other design parameters remain fixed or vary concerning the design variables. There are no definite rules to select these parameters, which may be important in a problem, because there may be some cases when one parameter is important to minimize the objective of the optimization problem in some sense, while it may be insignificant for maximization of the problem other sense. Thus the choice of the important parameters in an optimization problem largely depends on the user. However, the important factor to study is to understand the efficiency

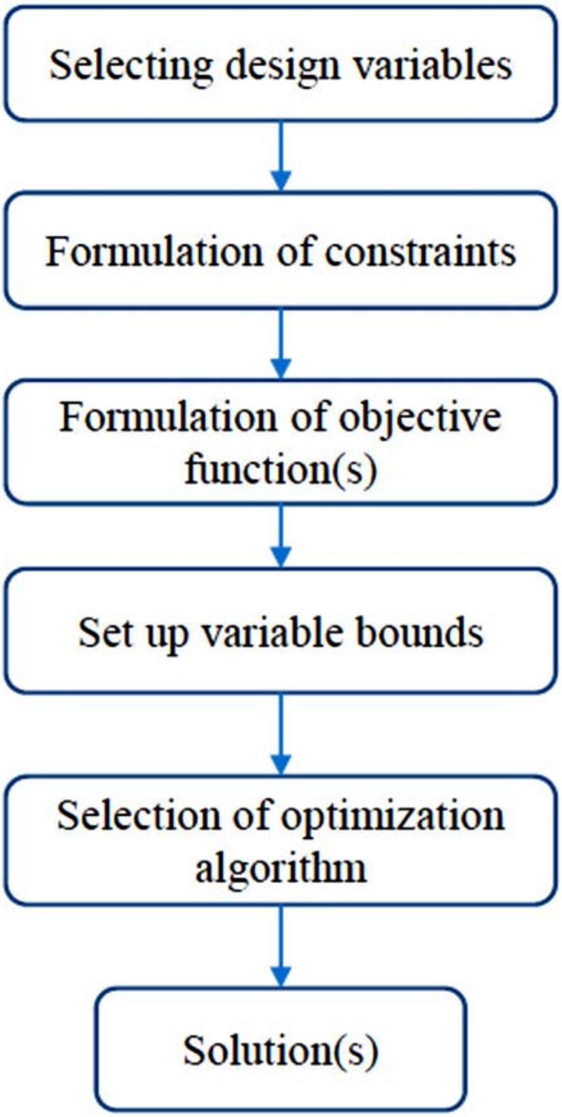

Selecting design variables
Formulation of constraints
Formulation of objective function(s)
Set up variable bounds
Selection of optimization algorithm
Solution(s)

Figure 1.2 Flow chart of optimal design technique.

and the speed of optimization algorithms, which largely depends on the number of selected design variables. In subsequent chapters, we shall discuss the efficiency of various optimization algorithms for the number of design variables.

1.1.2 Constraints

The next important thing after design variables is to identify the constraints of the optimization problem. The constraints reveal the functional relationship among the design variables and other design parameters obeying physical processes with limitations of availability of resources. In some cases, the design may static or dynamic. For example, problems involved in the civil and mechanical engineering usually the constraints are being formulated to manage stress and deflection limitations. But, there is no unique way to formulate the constraints in all the optimization problems. However, the nature and the number of constraints depend on the user. Therefore the user can fix some guidelines to consider the constraints. In this context, we can take an example of any mechanical component design problem, in which the constraint can be formulated to restrain the maximum stress developed. For a regular-shaped component, it is easy to formulate constraints by some existing rules. If the component is irregular shaped, then there may not exist any specific mathematical expression to deal with the maximum stress developed. As such, the numerical method, viz., the finite element method may handle such problems, and through simulation, the maximum stress can be computed. But the simulation procedure and the necessary input to the simulator as well as the output from the simulator must be understood. If we are discussing the type of constraints, then constraints are of two types: (1) equality and (2) inequality. Inequality constraints may have functional relationships among design variables of smaller than ($<$), greater than ($>$), or equal to ($=$) with a resource value.

1.1.3 Objective function

The third important task is to formulate an objective function in terms of design variables and other problem parameters. In many engineering and science problems, the common objective is the minimization of the overall cost of manufactured products or the maximization of profit gained. Different optimization problem has a different nature of the objective. Hence, based on the objective of the optimization problem, the objective function is to be either minimized or maximized. Although most of the objectives can be expressed in a mathematical form, there are few objectives for which expressing in a mathematical form may not be easy. In such cases, usually, an approximating mathematical expression is used. In addition, the real-life optimization problem also involves more than one objective function that the designer may want to optimize simultaneously. These types of problems are investigated by transforming either multiobjective function to single-objective functions or methods that follow any sequential optimization procedure and/or advanced optimization techniques.

1.1.4 Variable bounds

The final work is to set the minimum and the maximum bounds on each design variable. Certain optimization algorithms do not require this information. In these problems, the constraints surround the feasible region. Other problems require this information to confine the search algorithm within these bounds.

1.2 Engineering applications of optimization

Optimization is such an important tool, which very often used in many engineering problems. The following are some of the typical applications from various engineering disciplines.

1. In civil engineering, to design structures such as beams, frames, bridges, foundations, and towers for minimum cost.
2. To investigate the minimum weight design of structures for earthquake and wind.
3. Optimum design of gears, machine tools, and other mechanical components.
4. In aerospace engineering, to design the aircraft and the aerodynamic structure for minimum weight.
5. To design optimal water distribution in water resources systems.
6. To investigate the optimal trajectories in classical mechanics.
7. Optimum design of electrical machinery such as motors, generators, and transformers.
8. Optimum design of electrical networks.
9. Traveling salesman problems.
10. Assignment problems.
11. Transportation problems.
12. The optimal solution for equilibrium problems.
13. Optimal production planning, controlling, and scheduling.

14. Optimum design of control systems.
15. To design of material handling equipment, viz., conveyors, trucks, and cranes.
16. To design of pumps, turbines, and heat transfer equipment for maximum efficiency.
17. To design of optimum pipeline networks for process industries.
18. Analysis of statistical data and building empirical models from experimental results to obtain the most accurate representation of the physical phenomenon.
19. Allocation of resources or services among several activities to maximize the benefit.
20. Optimum design of chemical processing equipment and plants.

1.3 Optimization techniques

To investigate the aforementioned problems, various optimization techniques are employed depending on the type and the nature of the optimization problems. Fig. 1.3 shows the classification of optimization techniques and various methods to solve optimization problems.

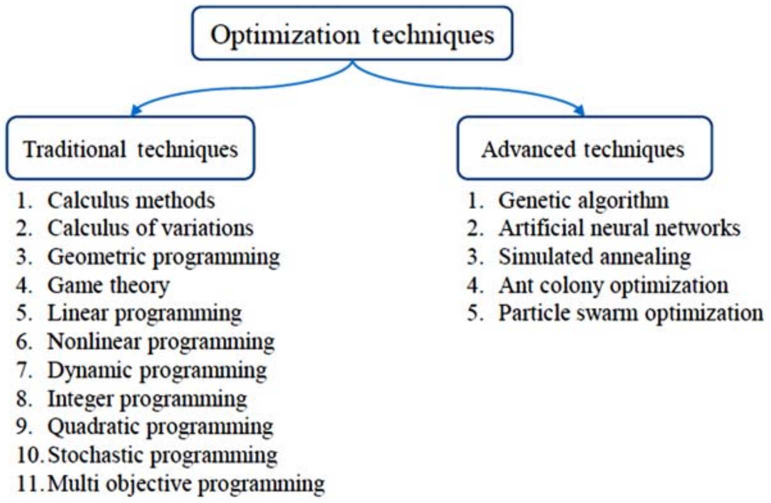


Figure 1.3 Classification of optimization techniques and methods to solve optimization problems.

Further reading

Adrian Bejan, G. T. (1995). *Thermal design and optimization*. New York: Wiley.
Arora, J. (2004). *Introduction to optimum design*. San Diego, CA: Academic Press.
Bhavikatti, S. S. (2010). *Fundamentals of optimum design in engineering*. New Delhi, India: New Age International (P) Limited Publishers.
Deb, K. (2004). *Optimization for engineering design algorithms and examples*. New Delhi, India: Prentice-Hall of India.
Farkas, J. (1984). *Optimum design of metal structures*. Chichester, UK: Ellis Horwood.
Gurdal, R. T. (1992). *Elements of structural optimization*. Dordrecht, The Netherlands: Kluwer Academic.
Gurdal, Z., Haftka, R. T., & Hajela, P. (1998). *Design and optimization of laminated composite*. New York: Wiley.
Jaluria, Y. (2007). *Design and optimization of thermal systems*. Boca Raton, FL: CRC Press.
Johnson, R. C. (1980). *Optimum design of mechanical elements*. New York: Wiley.
Kamat, M. P. (1993). *Structural optimization: Status and promise*. Washington, DC: AIAA.
Kolpakov, A. L. (1997). *Analysis, design and optimization of composite*. New York: Wiley.
Micheli, G. D. (1994). *Synthesis and optimization of digital circuits*. New York: McGraw-Hill.
Nayak, S., & Chakraverty, S. (2018). *Interval finite element method with MATLAB*. San Diego, CA: Academic Press.
Rao, S. S. (1995). *Optimization theory and applications*. New Delhi, India: New Age International (P) Limited Publishers.
Ravindran, A. (2006). *Engineering optimization: Methods*. New York: Wiley.
Venkataraman, P. (2002). *Applied optimization with MATLAB programming*. New York: Wiley.
Wright, J. N. (2006). *Numerical optimization*. New York: Springer.

This page intentionally left blank

CHAPTER TWO

Linear programming

Optimization is one of the essential habits in day-to-day life. Optimization involves the modeling of physical problems to mathematical formulations into objective functions and various strategies to investigate the same. Maximizing or minimizing such linear functions subject to linear constraints are known as linear programming. In other words, linear refers to linear relationships among the variables involved in the model, and programming refers to the mathematical modeling and solving of a problem that includes the use of given resources by choosing particular action to achieve the desired objective. The direct use of linear programming can be seen in oil refineries, chemical industries, cement industries, steel industries, mining, and food processing industries. To investigate the same we need to first study one of the major parts in any linear programming problem (LPP), that is the mathematical formulation and the modeling of any physical problems and then different techniques to handle the same. Accordingly this chapter comprises the formulation of the LPPs, various strategies to solve LPP, namely, graphical method, simplex method, artificial variable techniques, duality principle, and dual simplex method. The prime focus of this chapter is to provide an insight view of various LPPs and strategies to investigate them.

2.1 Formulation of the problem

As the formulation of the problem is one of the important steps for the LPP, it should be done using the following guidelines.

1. *Identify the decision variables*

This step involves the following three sub-steps. (1) Each constraint should be expressed in words. Then we have to see the constraints whether they are in $\geq$, $\leq$ or $=$ form. (2) The objective function should be expressed verbally. (3) Identify the decision variables verbally using (1) and (2).

© 2020 Elsevier Inc.
All rights reserved.

2. *Identify the problem data*

To formulate the linear programming we have to identify the problem data in terms of parameters and constants with the decision variables. As such the variables can be controlled but not the data set.

3. *Formulate the constraints*

Here we have to transform the verbal expressions of the constraints in terms of the required resources and its availability of each resource. Then each of the constraints is linear equality or inequality in terms of the decision variables. The values of decision variables must satisfy the constraints to produce an acceptable or feasible solution.

4. *Formulate the objective function*

Finally identify the objective function to be maximized or minimized. Then express the same in linear functions of decision variables.

The above steps are demonstrated in the following example and formulate the respective LPP.

Example 2.1. A tool and machine manufacturer produces two different types of products such as P_1 and P_2. Each P_1 product needs 4 hours of grinding and 2 hours of polishing; whereas each P_2 product requires 2 hours of grinding and 5 hours of polishing. The manufacturer has only 2 grinders and 3 polishers. Each grinder works for 40 hours a week and each polisher works for 60 hours a week. Profit on a P_1 product is ₹30 and on a P_2 model is ₹40. Whatever produced in a week is sold in the market. How should the manufacturer allocate its production capacity to the two types of products so that maximum profit in a week can be obtained?

Solution:

Decision variables

Let x_1 be the number of P_1 products, and x_2 be the number of P_2 products produced per week.

Objective function

The weekly profit (in ₹) is

$$Z = 30x_1 + 40x_2 \tag{2.1}$$

Problem data

To produce these number of models, the total number of grinding hours needed per week is

$$4x_1 + 2x_2 \tag{2.2}$$

and the total number of polishing hours required per week is

$$2x_1 + 5x_2. \tag{2.3}$$

Formulation of constraints

Since the number of grinding hours available is not more than 80 and the number of polishing hours is not more than 180, therefore,

$$4x_1 + 2x_2 \leq 80 \tag{2.4}$$

$$2x_1 + 5x_2 \leq 180. \tag{2.5}$$

Also since the negative number of products are not produced we must have

$$x_1 \geq 0 \text{ and } x_2 \geq 0. \tag{2.6}$$

Finally if we summarize all the steps the LPP statement for this problem is

Maximize $Z = 30x_1 + 40x_2$

subject to

$$4x_1 + 2x_2 \leq 80,$$

$$2x_1 + 5x_2 \leq 180,$$

$$x_1, x_2 \geq 0.$$

It is noted that the variables incorporated in the problem are called decision variables. Whereas Eq. (2.1) show the relationship of maximum profit and the decision variables, which is called the objective function. Finally the inequalities from Eqs. (2.4) to (2.6) are called constraints. Since the objective function and constraints are linear, this problem is named as LPP.

Example 2.2. A music company compose models M_1, M_2, and M_3 that have profit contributions per unit of ₹200, ₹450, and ₹650 respective. The weekly minimum production requirements are 50 units for model M_1, 150 units for model M_2, and 100 units for model M_3. Each type of model requires a certain amount of time for the manufacturing of parts for assembling and for parking. Specifically a dozen units of the model M_1 require 4 hours for manufacturing 3 hours for assembling, and 2 hours of packaging. Similarly for a dozen units of the model M_2 need 3 hours, 4 hours, and 2 hours, and model M_2 need 6 hours, 9 hours, and 5 hours of manufacturing, assembling, and packaging, respectively. During the forthcoming week, the company has available 150 hours of manufacturing, 180 hours of assembling, and 55 hours of packaging time. Formulate this problem as a linear programming model to maximize the total profit of the company.

Solution:

The data of the problem are summarized in Table 2.1.

Decision variables

Let x_1, x_2, and x_3 units of model M_1, M_2, and M_3 to be produced per week.

Objective function

$$\text{Maximize(total profit)} z = 200x_1 + 450x_2 + 650x_3 \tag{2.7}$$

Subject to the constraints

$$x_1 \geq 50, x_2 \geq 150, x_3 \geq 100 \tag{2.8}$$

$$\frac{1}{12}(4x_1 + 3x_2 + 6x_3) \leq 130 \text{ (Manufacturing time)} \tag{2.9}$$

$$\frac{1}{12}(3x_1 + 4x_2 + 9x_3) \leq 180 \text{ (Assembling time)} \tag{2.10}$$

$$\frac{1}{12}(2x_1 + 2x_2 + 5x_3) \leq 55 \text{ (Packaging time)} \tag{2.11}$$

and

$$x_1, x_2, x_3 \geq 0.$$

Table 2.1 Required summarized data for Example 2.2.

Resources	M_1	M_2	M_3	**Total availability**
Production requirements (per units)	50	150	100	
Manufacturing time (per dozen)	4	3	6	150
Assembling time (per dozen)	3	4	9	55
Packaging time (per dozen)	2	2	5	180
Contribution (per unit ₹)	200	450	650	

Table 2.2 Specification of carbon and silicon.

Materials	Minimum (%)	Maximum (%)
Carbon	3.20	3.40
Silicon	2.25	2.35

Table 2.3 Specifications and costs of steel scrap, cast iron scrap, and remelt from the foundry.

Material	Carbon %	Silicon %	Cost (in ₹)
Steel scrap	0.4	0.15	1030/tonne
Cast iron scrap	3.80	2.40	1100/tonne
Remelt from foundry	3.50	2.30	850/tonne

Example 2.3. ABC farm makes casting where they used electric furnace to melt iron with the following specifications (Table 2.2).

Specifications and costs of various raw materials used for this purpose are given in Table 2.3

If the total charge of iron metal required is 4 tonnes, find the weight in kilogram of each raw material that must be used in the optimal mix at minimum cost.

Solution: Let x_1, x_2, and x_3 be the amounts (in kg) of these raw materials. The objective is to minimize the cost that is

$$\text{Minimize } Z = \frac{1030}{1000}x_1 + \frac{1100}{1000}x_2 + \frac{850}{1000}x_3 \tag{2.12}$$

For iron melt to have a minimum of 3.2% carbon,

$$0.4x_1 + 3.8x_2 + 3.5x_3 \geq 3.2 \times 4000 \tag{2.13}$$

For iron melt to have a maximum of 3.4% carbon,

$$0.4x_1 + 3.8x_2 + 3.5x_3 \leq 3.4 \times 4000 \tag{2.14}$$

For iron melt to have a maximum of 2.25% silicon,

$$0.15x_1 + 2.41x_2 + 2.35x_3 \geq 2.25 \times 4000 \tag{2.15}$$

For iron melt to have a maximum of 2.35% silicon,

$$0.15x_1 + 2.41x_2 + 2.35x_3 \leq 2.35 \times 4000 \tag{2.16}$$

Also, since the materials added up must be equal to the full charge weight of 4 tonnes.

$$x_1 + x_2 + x_3 = 4000 \tag{2.17}$$

Finally since the amounts of raw materials cannot be negative

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$$

Thus the LPP is to find x_1, x_2 and x_3 which
Minimize $Z = 0.85x_1 + 0.9x_2 + 0.5x_3$
Subject to

$$\begin{aligned}0.4x_1 + 3.8x_2 + 3.5x_3 &\geq 12800\\ 0.4x_1 + 3.8x_2 + 3.5x_3 &\leq 13600\\ 0.15x_1 + 2.41x_2 + 2.35x_3 &\geq 9000\\ 0.15x_1 + 2.41x_2 + 2.35x_3 &\leq 9400\end{aligned}$$

$$x_1 + x_2 + x_3 = 4000$$

$$x_1, x_2, x_3 \geq 0.$$

Practice set 2.1

1. An electronic company produces three types of parts for automatic washing machines. It purchases casting of the parts from a local foundry and then finishes the part on drilling, shaping, and polishing machines.

The selling prices of parts A, B, and C are ₹10, ₹15, and ₹18, respectively. All the parts made can be sold. Casting for parts A, B, and C costs ₹8, ₹11, and ₹13, respectively. The shop possesses only one of each type of casting machine. Costs per hour to run each of the three machines are ₹25 for drilling, ₹38 for shaping, and ₹35 for polishing. The capacities (parts per hour) for each part on each machine are shown in the following table.

Machines	**Capacity per hour**		
	Part A	**Part B**	**Part C**
Drilling	25	40	25
Shaping	25	20	20
Polishing	40	30	40

The management wants to know how many parts of each type it should produce per hour to maximize profit for an hour's run.

Formulate the problem as a linear programming model to maximize total profit to the company.

2. A businessman is opening a new restaurant and has budgeted ₹9,00,000 for advertisement for the coming month. He is considering four types of advertising.

a. 30-second television commercials
b. 30-second radio commercials
c. Half-page advertisement in a newspaper
d. 30-second commercials in social media

The owner wishes to reach families (1) with income over ₹50,000 and (2) with income under ₹50,000. The amount of exposure of each media to families of types (1) and (2), the cost of each media is given in the following table.

Media	**Cost of advertisement (in ₹)**	**Exposure to families with annual income over ₹50,000**	**Exposure to families with annual income over ₹50,000**
Television	45,000	3,00,000	3,00,000
Radio	15,000	4,00,000	6,00,000
Newspaper	20,000	3,00,000	2,00,000
Social media	10,000	2,50,000	1,50,000

To have a balanced campaign, the owner has determined the following four restrictions.

a. There should be no more than four television advertisements.
b. There should be no more than four advertisements on social media.
c. There should be no more than 60 percent of all advertisements in the newspaper and social media put together.
d. There must be at least 45,00,000 exposures to families with an annual income of over ₹50,000.

Formulate this problem as a linear programming model to determine the number of each type of advertisement to be given to maximize the total number of exposures.

3. An investor has three investment opportunities available at the beginning of each year for the next 5 years. He has a total of ₹5,00,000 available for investment at the beginning of the first year. A summary of the financial characteristics of the three investment alternatives is presented in the following table.

Investment alternatives	Allowable size of the initialinvestment (in ₹)	Return (in %)	Timing of return	Immediate reinvestment possible
1	1,00,000	19	1 year later	Yes
2	Unlimited	16	2 years later	Yes
3	50,000	20	3 years later	Yes

The investor wishes to determine the investment plan that will maximize the amount of money that can be accumulated by the beginning of the 6th year in the future. Formulate this problem as a linear programming model to maximize total return.

4. Vitamins A and B are found in foods F and G. One unit of food F contains 3 units of Vitamin A and 4 units of Vitamin B. One unit of food G contains 6 units of vitamin A and 3 units of vitamin B, one unit of foods F and G cost ₹4 and ₹5, respectively. The minimum daily requirement (for a person) of Vitamins A and B is 80 and 100 units, respectively. Assuming that anything above the daily requirement of A and B is not harmful. Formulate this problem as a linear programming model to find out the optimum mixture of food F and G at the minimum cost which meets the daily minimum requirement of Vitamins A and B.

5. A company manufactures two models of garden rollers, namely, X and Y. When preparing the 2018 budget, it was found that the limitations on capacity were represented by the following weekly production maxima

Model	Foundry	Machine shop	Contribution per model (in ₹)
X	150	250	150
Y	250	200	120

Besides, the material required for model X was in short supply and sufficient only for 150 units per week, guaranteed for the year. Formulate the problem as a linear programming model to determine the optimal combination of output.

2.2 Graphical method

In general the linear programming model gives a set of solutions based on several possible values of decision variables, namely, $x_1, x_2, \cdots, x_n$, which satisfies the given constraints and also avails an optimal (maximum or minimum) value for the objective function. In this context, a linear programming model involving only two variables can be effectively solved by a graphical technique that provides a pictorial representation of the solution and gets insight into the basic concepts. Whereas most of the real-world problems have more than two variables and for such cases the graphical method is not suitable. But the graphical method provides an understanding to visualize the optimal solutions from the solution region that can be helpful to analyze real-world problems. The following are terminologies that will be used in the graphical method.

1. Solution: The set of decision variables $x_i, i = 1, 2, \cdots, n$, which satisfies the constraints of a LPP is said to be the solution.
2. Feasible solution: The set of decision variables $x_i, i = 1, 2, \cdots, n$, which satisfies the constraints as well as the nonnegativity conditions ($x_i \geq 0$) of a LPP is called a feasible solution.
3. Infeasible solution: The set of decision variables $x_i, i = 1, 2, \cdots, n$, which do not satisfy all constraints as well as nonnegativity conditions ($x_i \geq 0$) of a LPP is called the infeasible solution.
4. Basic solution: Consider the set of m simultaneous equations with n variables ($n > m$) in a LPP. The solution obtained by taking $(n - m)$ variables equal to zero and solving remaining m simultaneous equations with m variables is called basic solutions of the same. The variables that contribute to the basic solution are called basic variables, whereas the $(n - m)$ variables that do not contribute to the basic solution is called nonbasic variables.
5. Basic feasible solution: A feasible solution of a LPP which is a basic solution is called a basic feasible solution. In other words we may say that the basic feasible solution satisfies the nonnegativity condition. These solutions are of two types
   a. Degenerate: If at least one basic variable is zero.
   b. Nondegenerate: If all the m basic variables are nonzero and positive.

6. Optimum basic feasible solution: A basic feasible solution that optimizes (minimizes or maximizes) the objective function is said to be an optimum basic feasible solution.
7. Unbounded solution: A solution that decreases or increases the objective function value is called an unbounded solution.

2.2.1 Working procedure

To solve a LPP graphically we need the following step-by-step approach.

STEP 1: Formulation of the given problem as a LPP.
STEP 2: Plot the given constraints as equalities on x_1 and x_2 coordinate plane and then determine the convex region formed by the equalities.
STEP 3: Find the vertices of the convex region and investigate the value of the objective function at each vertex. The vertex that gives the optimal (maximum or minimum) value of the objective function gives the desired optimal solution to the problem.

The following are the important points of the graphical method.

1. The collection of all feasible solutions to a LPP forms a convex set whose extreme points correspond to the basic solutions.
2. A convex set is a polygon in which if any two points are selected arbitrarily and a straight line segment is drawn then that line segment lies completely within the polygon.
3. There are a finite number of basic feasible solutions within the feasible solution space.
4. If the convex set of the feasible solution which constitutes of equality simultaneous equations and nonnegative conditions is a convex polyhedron, then at least one of the extreme points gives an optimal solution.
5. If more than one extreme points give optimal solutions then the value of the objective function will be the same for all the convex combinations of these extreme points.

Example 2.4. Solve the following LPP graphically.

$$\text{Maximize } Z = 3x_1 + 4x_2 \tag{2.18}$$

subject to

$$4x_1 + 2x_2 \leq 80, \tag{2.19}$$

$$2x_1 + 5x_2 \leq 180, \tag{2.20}$$

$$x_1, x_2 \geq 0. \tag{2.21}$$

Solution: Plot the $x_1 - x_2$ coordinate system as shown in Fig. 2.1. Here the nonnegativity conditions Eq. (2.21) suggests that the values of x_1 and x_2 must lie in the first quadrant only.

Then plot the equalities (lines) $4x_1 + 2x_2 = 80$ and $2x_1 + 5x_2 = 180$. Any point on or below $4x_1 + 2x_2 = 80$ and any point on or below $2x_1 + 5x_2 = 180$ satisfies the restrictions Eqs. (2.19) and (2.20), respectively. This shows that the desired point (x_1, x_2) must be in the shaded convex region $OABC$. Hence the shaded region $OABC$ is called the feasible solution region for the given problem. The vertices of feasible solution region are $O(0,0), A(20,0), B(2.5,35)$, and $C(0,36)$, respectively.

The values of the objective function Eq. (2.18) at these points are

$Z_O = 0, Z_A = 60, Z_B = 147.5, Z_C = 144$ at points O, A, B, and C, respectively.

Thus the maximum value of Z is 147.5 and it occurs at B. Hence the optimal solution of this LPP is

$x_1 = 2.5, x_2 = 35$, and $Z_{\max} = 147.5$.

Example 2.5. Find the maximum value of

$$Z = 2x + 3y, \tag{2.22}$$

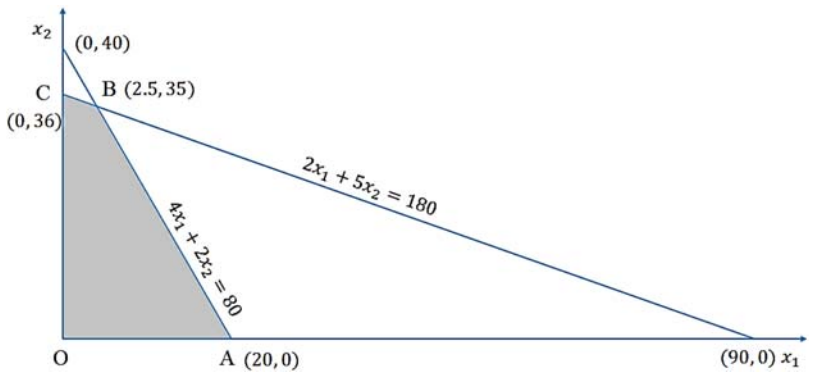


Figure 2.1 Graphical solution of the LPP given in Example 2.4.

subject to the constraints:

$$x + y \leq 30, \tag{2.23}$$

$$y \geq 3, \tag{2.24}$$

$$x - y \geq 0, \tag{2.25}$$

$$0 \leq y \leq 12 \tag{2.26}$$

and

$$0 \leq x \leq 20. \tag{2.27}$$

Solution: Any arbitrary point (x, y) that satisfies the nonnegative conditions $x \geq 0, y \geq 0$ lies in the first quadrant only. Since we have the inequalities $x + y \leq 30$, $y \geq 3$, $y \leq 12$, $x \geq y$, and $x \leq 20$, the desired point (x, y) will lie within the convex region $ABCDE$ (shown as shaded region in Fig. 2.2). The vertices of shaded region $ABCDE$ are $A(3, 3)$, $B(20, 3)$, $C(20, 10)$, $D(18, 12)$, and $E(12, 12)$. Accordingly the values of Z at these five vertices A, B, C, D, and E are $Z_A = 15, Z_B = 49, Z_C = 70, Z_D = 72$, and $Z_E = 60$, respectively. Hence the maximum value of Z is 72, which occurs at the vertex D. As such, the solution to the LPP is $x = 18, y = 12$, and maximum $Z = 72$.

Example 2.6. A company making cold drinks has two bottling plants located at towns T_1 and T_2. Each plant produces three drinks A, B, and C, and its production capacity per day is shown in Table 2.4.

The marketing department of the company forecasts demands for 80,000 bottles of A, 22,000 bottles of B, and 40,000 bottles of C during June. The operating costs per day of plants at T_1 and T_2 are ₹6000 and

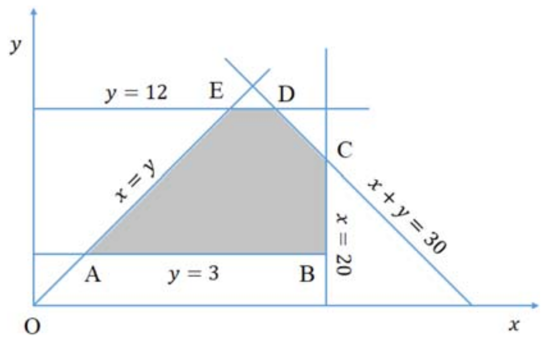


Figure 2.2 Graphical solution of the LPP given in Example 2.5.

Table 2.4 Production capacity of cold drinks A, B, and C.

Cold drinks	**Plant**	
	T_1	T_2
A	6000	2000
B	1000	2500
C	3000	3000

₹4000, respectively. Find (graphically) the number of days for which each plant must be run in June to minimize the operating costs while meeting the market demand.

Solution: Let the plants at T_1 and T_2 be run for x_1 and x_2 days. Then the objective is to minimize the operation cost that is

$$\text{Minimize } Z = 6000x_1 + 4000x_2 \tag{2.28}$$

Constraints on the demand for the three cold drinks are:
For cold drink A,

$$6000x_1 + 2000x_2 \geq 80,000 \text{ or } 3x_1 + x_2 \geq 40. \tag{2.29}$$

For cold drink B,

$$1000x_1 + 2500x_2 \geq 22,000 \text{ or } x_1 + 2.5x_2 \geq 22. \tag{2.30}$$

For cold drink C,

$$3000x_1 + 3000x_2 \geq 40,000 \text{ or } x_1 + x_2 \geq \frac{40}{3}. \tag{2.31}$$

$$x_1, x_2 \geq 0. \tag{2.32}$$

Thus the LPP is to minimize Eq. (2.28) subject to constraints Eqs. (2.29) to (2.32).

The solution space satisfying the constraints Eqs. (2.29) to (2.32) is shown shaded in Fig. 2.3. It is seen that the solution space is unbounded. The constraints Eq. (2.31) is dominated by the constraints Eqs. (2.29) and (2.30) and hence does not affect the solution space. Such a constraint as Eq. (2.31) is called the redundant constraint.

The vertices of the convex region ABC are $A(22,0), B(12,4)$, and $C(0,40)$. Values of the objective function Eq. (2.28) at these vertices are $Z_A = 132,000, Z_B = 88,000$, and $Z_C = 160,000$.

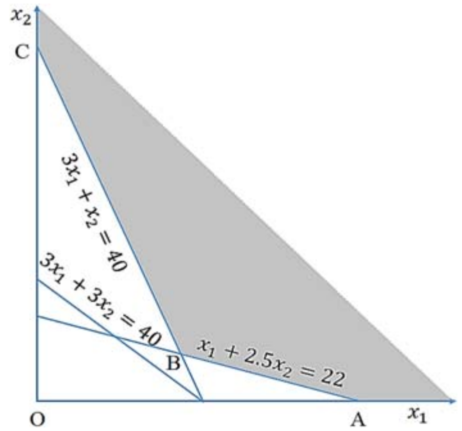


Figure 2.3 Graphical solution of the LPP given in Example 2.6.

Thus the minimum value of Z is Rs. 88,000 and it occurs at B. Hence the solution to the problem is $x_1 = 12$ days, $x_2 = 4$ days, and $Z_{\min} = \text{Rs. } 88,000$.

The constraints in general give the region of feasible solution which may be bounded or unbounded. From the above-mentioned problems we observed that LPP is having a finite solution and the optimal solution existed at a vertex of the feasible region as well as the optimal solution was unique. But it is not always so. LPP may have (1) a unique optimal solution, (2) an infinite number of optimal solutions, (3) an unbounded solution, or (4) no solution. In the following, we will illustrate the cases (2) to (4) with example LPP.

Example 2.7. Unix Hardware Company uses milling machines, grinding machines, and lathes to produce two motor parts. The machining times required for each part available on different machines and the profit on each motor part are given in Table 2.5.

Determine the number of parts I and II to be manufactured per week to maximize the profit.

Solution: Let x_1, and x_2 be the number of parts I and II manufactured per week, respectively.

The objective function is

$$\text{maximize } Z = 100x_1 + 40x_2. \tag{2.33}$$

Constraints are based on the availability of time for each machine.

Table 2.5 Required machining times for each part and the profit on each motor part.

Types of machines	**Machine time required for the motor part (in min)**		**Maximum time available per week (in min)**
	I	**II**	
Milling	10	4	2000
Grinding	3	2	900
Lathes	6	12	3000
Profit/unit (in ₹)	100	40	

For milling machines,

$$10x_1 + 4x_2 \leq 2000 \tag{2.34}$$

For grinding machines,

$$3x_1 + 2x_2 \leq 900 \tag{2.35}$$

For lathes,

$$6x_1 + 12x_2 \leq 3000 \tag{2.36}$$

$$x_1, x_2 \geq 0. \tag{2.37}$$

The solution space satisfying Eqs. (2.34), (2.35), (2.36), and the non-negative restrictions Eq. (2.37) is shown in Fig. 2.4. Note that Eq. (2.35) is a redundant constraint as it does not affect the solution space. The vertices of the convex region $OABC$ are $O(0,0)$, $A(200,0)$, $B(125,187.5)$, and $C(0,250)$, respectively. The values of the objective function at different points O, A, B, and C are $Z_O = 0$, $Z_A = 20,000$, $Z_B = 20,000$, and $Z_C = 10,000$, respectively. It is observed that the maximum value of Z occurs at two vertices, namely, A and B. Also it may be pointed out that the line joining A and B will give the same maximum value of Z. Hence for this case there is an infinite number of feasible solutions that provide the same maximum value of Z. So there is no unique optimal solution to this LPP and any point on the line AB can be taken to give the profit of ₹20,000.

Example 2.8. Using the graphical method, solve the following LPP.

$$\text{Maximize } Z = 2x_1 + 3x_2 \tag{2.38}$$

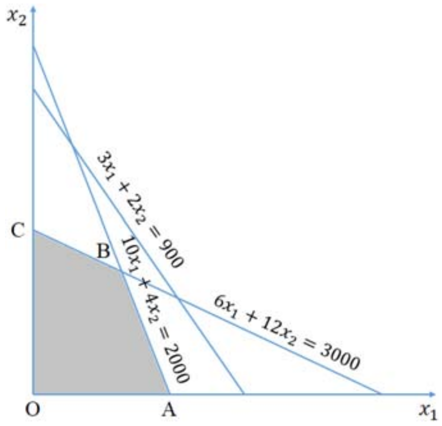


Figure 2.4 Graphical solution of the LPP given in Example 2.7.

Subjected to

$$x_1 - x_2 \leq 2 \tag{2.39}$$

$$x_1 + x_2 \geq 4 \tag{2.40}$$

$$x_1, x_2 \geq 0. \tag{2.41}$$

Solution: The solution space satisfying the constraints Eqs. (2.39) and (2.41) is shown in the convex shaded region in Fig. 2.5. Here we get the unbounded solution space and the vertices of the feasible region are $A(3, 1)$ and $B(0, 4)$. Then the values of objective function Z at A and B are $Z_A = 9$ and $Z_B = 12$, respectively. But we may find infinitely many points in the convex region where Z possesses much higher values. For example, at point $(5, 6)$, the value of Z is 28. Hence this LPP has an unbounded solution.

Example 2.9. Solve graphically the following LPP.

$$\text{Maximize } Z = 4x_1 + 3x_2 \tag{2.42}$$

Subject to

$$x_1 - x_2 \leq -1 \tag{2.43}$$

$$-x_1 + x_2 \leq 0 \tag{2.44}$$

$$x_1, x_2 \geq 0. \tag{2.45}$$

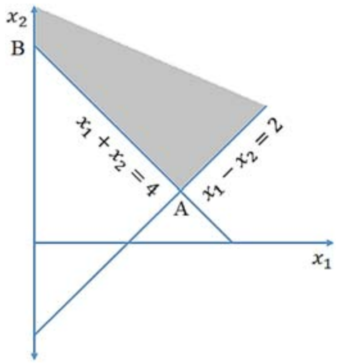


Figure 2.5 Graphical solution of the LPP given in Example 2.8.

Solution: The two solution spaces satisfying Eqs. (2.43) to (2.45) are shown in Fig. 2.6. It is shown that there are no common point (x_1, x_2) in both the shaded regions and hence the problem cannot be solved. Since the constraints are inconsistent, the solution does not exist for LPP.

Practice set 2.2

1. Use the graphical method to solve the following LPP.
   Maximize $Z = 2x_1 + x_2$
   Subject to the constraints

$$x_1 + 2x_2 \le 10$$

$$x_1 - x_2 \le 2$$

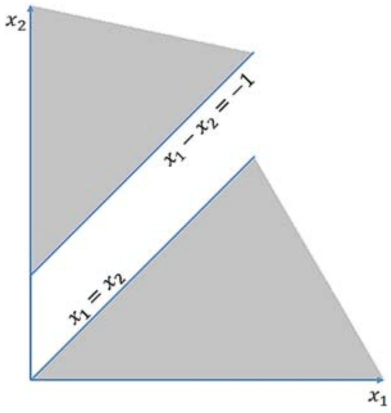


Figure 2.6 Graphical solution of the LPP given in Example 2.9.

$$x_1 + x_2 \leq 6$$

$$x_1 - 2x_2 \leq 1$$

and

$$x_1, x_2 \geq 0.$$

2. Use the graphical method to solve the following LPP
   Maximize $Z = -x_1 + 2x_2$
   Subject to the constraints

$$-x_1 + 3x_2 \leq 10$$

$$x_1 - x_2 \leq 2$$

$$x_1 + x_2 \leq 6$$

and

$$x_1, x_2 \geq 0.$$

3. Use the graphical method to solve the following LPP
   Maximize $Z = 2x_1 + 3x_2$
   Subject to the constraints

$$x_1 - x_2 \geq 0$$

$$x_1 + x_2 \leq 30$$

$$x_2 \geq 3$$

$$0 \leq x_2 \leq 12$$
$$0 \leq x_1 \leq 20$$

and

$$x_1, x_2 \geq 0.$$

4. Use the graphical method to solve the following LPP
   Maximize $Z = 4x_1 + 2x_2$
   Subject to the constraints

$$-x_1 + 2x_2 \leq 6$$

$$-x_1 + x_2 \leq 2$$

and

$$x_1, x_2 \geq 0.$$

5. Use the graphical method to solve the following LPP

   Maximize $Z = 6x_1 + 12x_2$

   Subject to the constraints

$$x_1 + x_2 \leq 15$$
$$x_1 + 2x_2 \leq 10$$
$$2x_1 - 5x_2 \leq 20$$

   and

$$x_1, x_2 \geq 0.$$

6. Use the graphical method to solve the following LPP

   Maximize $Z = 1.75x_1 + 1.5x_2$

   Subject to the constraints

$$4x_1 + 5x_2 \leq 20$$
$$8x_1 + 5x_2 \leq 320$$
$$x_1 \geq 15, x_2 \geq 10$$

   and

$$x_1, x_2 \geq 0.$$

7. The manager of an oil refinery must decide on the optimal mix of two possible blending process of which the inputs and outputs per production run as follows.

Process (units)	**Input (units)**		**Output (units)**	
	Grade A	**Grade B**	**Gasoline X**	**Gasoline Y**
1	5	3	5	8
2	4	5	4	4

   The maximum amounts available for crudes A and B are 200 units and 150 units, respectively. Market requirements show that at least 100 units of gasoline X and 80 units of gasoline Y must be produced. The profits per production run for processes 1 and 2 are ₹300 and ₹400, respectively. Formulate this problem as a linear programming model and solve it by using the graphical method to evaluate the production run for processes 1 and 2.

8. Two products A and B are to be manufactured. One single unit of product A requires 2.4 minutes of punch press time and 5 minutes of assembly time. The profit for product A is ₹0.60 per unit. One single

unit of product B requires 3 minutes of punch press time and 2.5 minutes of welding time. The profit for product B is ₹0.70 per unit. The capacity of the punch press department available for these products is 1200 minutes per week. The welding department has an idle capacity of 600 minutes per week and the assembly department has a capacity of 1500 minutes per week. Formulate this problem as a linear programming model to determine the quantities of products A and B that would yield the maximum.

9. A pineapple firm produces two products, namely, canned pineapple and canned juice. The specific amounts of material, labor, and equipment required to produce each product and the availability of each of these resources are presented in the following table.

	Canned juice	**Pineapple**	**Available resources**
Labor (man hours)	3	2.0	12
Equipment (per hours)	1	2.3	6.9
Material (unit)	1	1.4	4.9

Assuming one unit of each canned juice and canned pineapple has profit margins of ₹5 and ₹3, respectively. Formulate this LPP and solve it graphically.

10. A small-scale manufacturer has production facilities for producing two different products. Each of the products requires three different operations, namely, grinding, polishing, and testing. Product A requires 15, 20, and 10 minutes to grind, polish, and test, respectively. On the other hand product B requires 7.5, 40, and 45 minutes for grinding, polishing, and testing, respectively. The production run calls for at least 4.5 hours of grinding time, 20 hours of polishing, and 15 hours of testing time. If manufacturing product A costs ₹60 and product B costs ₹90, then determine the number of units of each product the firm should produce to minimize the cost operation.

2.3 General LPP

This section describes LPPs with two or more variables along with its canonical and standard forms. In general any LPP with two or more variables can be expressed as follows.

Find the unknown (variables) $x_1, x_2, \ldots, x_n$ that maximize (or minimize) the objective function

$$Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{2.46}$$

Subject to the constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &\leq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &\leq b_m \end{aligned} \tag{2.47}$$

and the nonnegativity restrictions

$$x_1, x_2, \ldots, x_n \geq 0, \tag{2.48}$$

where, $c_j, b_i, a_{ij}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$ are constants.

In this LPP all the constraints are in less than equal inequalities $(\leq)$ type, but sometimes the constraints may also contain few equalities, some others may be inequalities of $(\leq)$ type and remaining inequalities of $(\geq)$ type. These inequalities can be changed to equalities by adding (or subtracting) some nonnegative variables to (or from) the left-hand side of such constraints. The conversion of inequalities into equalities may be done by using the following slack and surplus variables.

Case 1. If the constraints of a general LPP are of the following type

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, i = 1, 2, 3, \ldots, m \tag{2.49}$$

then the nonnegative variables s_i which satisfy

$$\sum_{j=1}^{n} a_{ij}x_j + s_i = b_i, i = 1, 2, 3, \ldots, m \tag{2.50}$$

are called the slack variables.

Case 2. If the constraints of a general LPP are of the following type

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i, i = 1, 2, 3, \ldots, m \tag{2.51}$$

then the nonnegative variables s_i which satisfy

$$\sum_{j=1}^{n} a_{ij}x_j - s_i = b_i, i = 1, 2, 3, \ldots, m \tag{2.52}$$

are called the surplus variables.

2.3.1 Canonical and standard forms of LPP

The next step after the formulation of LPP is investigation of the solution. But before using any method to solve the LPP, we need to convert the formulated LPP into a suitable form and these are discussed as follows.

1. *Canonical form*: The general LPP is very often expressed in the following form

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{2.53}$$

subject to the constraints

$$a_{i1}x_1 + a_{i2}x_2 + \ \ldots \ + a_{in}x_n \leq b_i;\ i = 1, 2, \ldots, m \tag{2.54}$$

$$x_1, x_2, \ldots, x_n \geq 0, \tag{2.55}$$

using some elementary transformations.

The current form of the LPP is known as canonical form and it has the following characteristics.

- **a.** The objective function is of a maximization type,
- **b.** All constraints are of $(\leq)$ type, and
- **c.** All variables x_i are nonnegative.

2. *Standard form*: The general LPP can also be written in the following form

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{2.56}$$

subject to the constraints

$$a_{i1}x_1 + a_{i2}x_2 + \ \ldots \ + a_{in}x_n = b_i;\ i = 1, 2, \ldots, m \tag{2.57}$$

$$x_1, x_2, \ldots, x_n \geq 0. \tag{2.58}$$

The current form of the LPP is known as standard form and it has the following characteristics.

- **a.** The objective function is of a maximization type,
- **b.** All the constraints are expressed as equations (equality),
- **c.** The right-hand side of each constraint is nonnegative, and
- **d.** All variables are nonnegative.

Example 2.10. Convert the following LPP to the standard form

$$\text{Maximize } Z = 2x_1 + 3x_2 + 5x_3,$$

subject to

$$7x_1 - 4x_2 \leq 5,$$
$$5x_1 + 2x_2 + 3x_3 \geq 13,$$
$$6x_1 + 4x_3 \leq 3,$$
$$x_1, x_2 \geq 0.$$

Solution: Given LPP the variable x_3 is unrestricted. So we may take $x_3 = x_3^{'} - x_3^{''}$, where $x_3^{'}, x_3^{''} \geq 0$. Now we may write the given constraints in the following form

$$7x_1 - 4x_2 \leq 5,$$
$$5x_1 + 2x_2 + 3x_3^{'} - 3x_3^{''} \geq 13,$$
$$6x_1 + 4x_3^{'} - 4x_3^{''} \leq 3,$$
$$x_1, x_2, x_3^{'}, x_3^{''} \geq 0.$$

The current problem can be converted into standard form by introducing the slack/surplus variables that results

$$\text{Maximize } Z = 2x_1 + 3x_2 + 5x_3^{'} - 5x_3^{''},$$

subject to

$$7x_1 - 4x_2 + s_1 = 5,$$
$$5x_1 + 2x_2 + 3x_3^{'} - 3x_3^{''} - s_2 = 13,$$
$$6x_1 + 4x_3^{'} - 4x_3^{''} + s_3 = 3,$$
$$x_1, x_2, x_3^{'}, x_3^{''}, s_1, s_2, s_3 \geq 0.$$

Example 2.11. Express the following problem in standard form.

$$\text{Minimize } Z = 4x_1 + 5x_2$$

subject to

$$2x_1 - x_2 - 3x_3 = -4,$$

$$3x_1 + 4x_2 + x_4 = 10,$$

$$x_1 - 5x_2 = 15,$$

$$x_1, x_3, x_4 \geq 0.$$

Solution: The current problem, as x_3 and x_4 are the slack/surplus variables. Whereas x_1 and x_2 are the decision variables. Also the variable x_2 is unrestricted. So we may consider $x_2 = x_2' - x_2''$ where $x_2', x_2'' \geq 0$.

We can write the problem in the standard form as given below.

$$\text{Maximize } Z' = -Z = -4x_1 - 5x_2' + 5x_2''$$

subject to

$$-2x_1 + x_2' - x_2'' + 3x_3 = 4$$

$$3x_1 + 4x_2' - 4x_2'' + x_4 = 10$$

$$x_1 - 5x_2' + 5x_2'' = 15$$

$$x_1,\ x_2',\ x_2'',\ x_3,\ x_4 \geq 0.$$

Practice set 2.3

1. Convert the following LPP to standard form

$$\text{Maximize } Z = 3x_1 + 5x_2 + 8x_3$$

subject to

$$2x_1 - 5x_2 \leq 6,$$

$$3x_1 + 2x_2 + x_3 \geq 5,$$

$$3x_1 + 4x_3 \leq 3,$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0.$$

2. Reduce the following LPP in the standard form

$$\text{Minimize } Z = 3x_1 + 5x_2 + 7x_3$$

subject to

$$-3x_1 + 2x_2 \le 5,$$

$$2x_1 + 3x_2 + 4x_3 \ge 5,$$

$$2x_1 + 5x_3 \le 7,$$

$$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0.$$

3. Express the following LPP to standard form

$$\text{Maximize } Z = 2x_1 - x_2 + 3x_3$$

subject to

$$x_1 + 2x_2 + x_3 \le 8,$$

$$2x_1 - x_2 + x_3 - 2 \ge 0,$$
$$4x_1 - 2x_2 - 3x_3 = -6,$$

$$x_1 \ge 0, x_2 \ge 0.$$

4. Convert the following LPP to the standard form.
Maximize $Z = 3x_1 + 5x_2 + 7x_3$, subject to the constraints $6x_1 - 4x_2 \le 5$, $3x_1 + 2x_2 + 5x_3 \ge 11$, $4x_1 + 3x_3 \le 2$, $x_1, x_2 \ge 0$.

5. Express the following problem in the standard form.
Minimize $Z = 3x_1 + 4x_2$, subject to the constraints $2x_1 - x_2 - 3x_3 = -4$, $3x_1 + 5x_2 + x_4 = 10$, $x_1 - 4x_2 = 12$, $x_1, x_3, x_4 \ge 0$.

2.4 Simplex method

From the previous section, which is a graphical solution of LPP, it is observed that the region of the feasible solution was found to be convex, bounded by vertices, and edges joining them. The optimal solution obtained at some vertex. But, if the optimal solution was not unique, the total optimal points were on an edge. These observations also hold for the general LPP containing two or more decision variables. Essentially the problem is that of finding the particular vertex of the convex region that corresponds to the optimal solution. In this context the most commonly used method for locating the optimal solution is the simplex method. This method works on step-by-step moving of one vertex to the adjacent vertex where the objective function gives a better solution than the previous. As we get a finite number of vertices in this method, the optimal vertex and the corresponding solution are obtained in a finite number of steps.

Let us consider an LPP consists of n decision variables and m $(m \leq n)$ constraints. Then the simplex method can reduce an infinite number of solutions to a finite number of promising solutions by using the following steps.

STEP 1: After converting the given LPP to standard form, we get $(m + n)$ variables. Then the initial solution can be found by setting n variables equals to zero and the remaining m equations are solved to get the solution. The obtained solution exists and is also unique. Here the m variables are called basic variables and they form a basic solution whereas n zero variables are known as nonbasic variables.

STEP 2: The variables must satisfy the nonnegative restrictions, that is, variables must be nonnegative. But some of the basic solutions possess negative valued variables and those solutions are called infeasible solution, which is discarded. As such the process starts with a basic solution that contains nonnegative variables. Then the next basic solution also followed the same nonnegative restriction that ensures the feasibility condition. Such a solution is called a basic feasible solution. If all the variables in the basic feasible solution are positive, then it is called a nondegenerate solution and if some of the variables are zero, it is called a degenerate solution.

STEP 3: A new basic feasible solution may be obtained from the previous one by equating one of the basic variables to zero and replacing it by a new nonbasic variable. The eliminated variable is called the outgoing variable while the new variable is known as the incoming variable.

It is noted that the incoming variable must improve the value of the objective function, which should obey the optimality condition. These steps are repeated until no further improvement is possible. Finally the resulting solution is called the optimal basic feasible solution. In other words we may say that the simplex method is based on the feasibility and optimality conditions.

Common terminologies used in the above steps are discussed in detail by taking a general LPP in the following standard form.

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{2.59}$$

subject to

$$\sum_{j=1}^{n} a_{ij}x_j + s_j = b_j,\ j = 1, 2, \ldots, m \tag{2.60}$$

$$x_j \geq 0, s_j \geq 0, \ j = 1, 2, \ldots, n. \tag{2.61}$$

1. The combination of variables $x_1, x_2, \ldots, x_n$ is a solution of the general LPP if it satisfies the constraints (2.60).
2. The combination of variables $x_1, x_2, \ldots, x_n$ is a feasible solution of the general LPP if it satisfies both the constraints (2.60) and (2.61). The set of all possible feasible solutions is called a feasible region. An LPP is said to be infeasible if this set is empty.
3. The basic solution is defined as the solution of the m basic variables when each of the n nonbasic variables is required to zero.
4. A basic feasible solution is termed as the basic solution that satisfies the nonnegativity restrictions (2.61).
5. An optimal solution is known as the basic feasible solution that optimizes the objective function (2.59) provided the conditions (2.60) and (2.61) holds.
6. A nondegenerate basic feasible solution is that feasible solution that contains exactly m nonzero basic variables. If any of the basic variables becomes zero, it is called a degenerate basic feasible solution.

Example 2.12. Find all the basic solutions of the following system of equations identifying in each case the basic and nonbasic variables

$$2x_1 + x_2 + 4x_3 = 11, \tag{2.62}$$

$$3x_1 + x_2 + 5x_3 = 14. \tag{2.63}$$

Investigate whether the basic solutions are degenerate basic solutions or not. Hence find the basic feasible solution of the system.

Solution: Since there are $m + n = 3$ variables and $m = 2$ constraints in this problem, a basic solution can be obtained by setting any one variable equal to zero and then solving the resulting equations. Hence the total number of basic solutions will be $m + n_{C_n} = 3_{C_2} = 3$.

The constraints (2.62) and (2.63) can be written in the following matrix equation form

$$\begin{bmatrix} 2 & 1 & 4 \\ 3 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 4 \end{bmatrix}.$$

The different possible matrix equations corresponding to the various basic solutions are

1. When x_3 is zero we get

$$\begin{bmatrix} 2 & 1 \\ 3 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 11 \\ 4 \end{bmatrix}.$$

2. When x_1 is zero we get

$$\begin{bmatrix} 1 & 4 \\ 1 & 5 \end{bmatrix}\begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 4 \end{bmatrix}.$$

3. When x_2 is zero we get

$$\begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 4 \end{bmatrix}.$$

The characteristics of the various basic solutions are given in Table 2.6.

The basic feasible solutions are (1) $x_1 = 3, x_2 = 5, x_3 = 0$ and (2) $x_1 = \frac{1}{2}, x_2 = 0, x_3 = \frac{5}{2}$, which are also nondegenerate basic solutions.

Example 2.13. Find an optimal solution to the following LPP. Compute all basic solutions and find the solution that maximizes the objective function.

$$2x_1 + 3x_2 - x_3 + 4x_4 = 8, \tag{2.64}$$

$$x_1 - 2x_2 + 6x_3 - 7x_4 = -3, \tag{2.65}$$

$$x_1, x_2, x_3, x_4 \geq 0. \tag{2.66}$$

$$\text{Maximum } Z = 2x_1 + 3x_2 + 4x_3 + 7x_4 \tag{2.67}$$

Solution: In this problem, there are four variables and two constraints. So, a basic solution can be obtained by taking any two variables equal to zero and then solving the resulting equations. Accordingly the total number of basic solutions are $4_{C_2} = 6$. The characteristics of the various basic solutions are given in Table 2.7.

From Table 2.7, the optimal basic feasible solution is $x_1 = 0$, $x_2 = 0, x_3 = 44/17, x_4 = 45/17$ and the maximum value of $Z = 28.9$.

Example 2.14. Using matrix–vector notation, show that the following linear equations have degenerate solutions

Table 2.6 Characteristics of the various basic solutions for Example 2.12.

No. of basic solution	Basic variables	Nonbasic variables	Values of basic variables	Solution	Is the solution feasible? (are all $x_j > 0$?)	Is the solution degenerate?
1.	x_1, x_2	x_3	$2x_1 + x_2 = 11$ $3x_1 + x_2 = 14$	$x_1 = 3,$ $x_2 = 5$	Yes	No
2.	x_2, x_3	x_1	$x_2 + 4x_3 = 11$ $x_2 + 5x_3 = 14$	$x_2 = 3,$ $x_3 = -1$	No	Yes
3.	x_1, x_3	x_2	$2x_1 + 4x_3 = 11$ $3x_1 + 5x_3 = 14$	$x_1 = \frac{1}{2},$ $x_3 = \frac{5}{2}$	Yes	No

$$2x_1 + x_2 - x_3 = 2 \tag{2.68}$$

$$3x_1 + 2x_2 + x_3 = 3 \tag{2.69}$$

Solution: The given system of equations can be written in the following matrix form

$$\begin{bmatrix} 2 & 1 & -1 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \tag{2.70}$$

Since the coefficient matrix has the order of 2×3, there can be $3_{C_2} = 3$ submatrices of order 2×2. These are $\begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$, $\begin{bmatrix} 2 & -1 \\ 3 & 1 \end{bmatrix}$, and $\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$, respectively.

Setting $x_3 = 0$, we get

$$\begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \tag{2.71}$$

From (2.71), we obtain $x_1 = 1$ and $x_2 = 0$.
Similarly setting $x_2 = 0$, we get

$$\begin{bmatrix} 2 & -1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \tag{2.72}$$

From (2.72) we obtain $x_1 = 1$ and $x_3 = 0$.
Setting $x_1 = 0$ we get

Table 2.7 Characteristics of the various basic solutions for Example 2.13.

No. of basic solution	Basic variables	Nonbasic variables	Values of basic variables	Solution	Is the solution feasible? (are all $x_j > 0$?)	Value of Z	Is the solution optimal?
1.	x_1, x_2	$x_3 = 0$, $x_4 = 0$	$2x_1 + 3x_2 = 8$ $x_1 - 2x_2 = -3$	$x_1 = 1$, $x_2 = 2$	Yes	8	No
2.	x_1, x_3	$x_2 = 0$, $x_4 = 0$	$2x_1 - x_3 = 8$ $x_1 + 6x_3 = -3$	$x_1 = -\frac{14}{13}$, $x_3 = -\frac{67}{13}$	No	–	–
3.	x_1, x_4	$x_2 = 0$, $x_3 = 0$	$2x_1 + 4x_4 = 8$ $x_1 - 7x_4 = -3$	$x_1 = \frac{22}{9}$, $x_4 = \frac{7}{9}$	Yes	10.3	No
4.	x_2, x_3	$x_1 = 0$, $x_4 = 0$	$3x_2 - x_3 = 8$ $-2x_2 + 6x_3 = -3$	$x_2 = \frac{45}{16}$, $x_3 = \frac{7}{16}$	Yes	10.2	No
5.	x_2, x_4	$x_1 = 0$, $x_3 = 0$	$3x_2 + 4x_4 = 8$ $-2x_2 - 7x_4 = -3$	$x_2 = \frac{132}{39}$, $x_4 = \frac{-7}{13}$	No	–	–
6.	x_3, x_4	$x_1 = 0$, $x_2 = 0$	$-x_3 + 4x_4 = 8$ $6x_3 - 7x_4 = -3$	$x_3 = \frac{44}{17}$, $x_4 = \frac{45}{17}$	Yes	28.9	Yes

$$\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \tag{2.71}$$

From Eq. (2.71) we obtain $x_2 = 5/3$ and $x_3 = -1/3$.

Hence, the solution $x_1 = 0$, $x_2 = 5/3$ and $x_3 = -1/3$ is not feasible. The remaining solutions $x_1 = 1$, $x_2 = 0$ and $x_3 = 0$; and $x_1 = 1$, $x_2 = 0$ and $x_3 = 0$ are a degenerate basic feasible solutions.

2.4.1 Reduction of feasible solution to a basic feasible solution

Consider an LPP with m constraints and n variables where $m < n$. The constraints are in the following forms

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m. \end{gathered} \tag{2.72}$$

The coefficient matrix is defined as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}. \tag{2.73}$$

Suppose there exists a feasible solution for Eq. (2.72) then this feasible solution can be reduced to a basic feasible solution if we consider the steps given below.

STEP 1: If the column vectors of A are linearly dependent then we may express the column vector $v_j = \begin{bmatrix} a_{1j} & a_{2j} & \cdots & a_{mj} \end{bmatrix}^T \in A$, $j = 1, 2, \cdots, n$ as linear combinations the other column vectors. That is for scalars α_j, we may write $v_j = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n$.

STEP 2: Compute the ratios x_j/a_j; $\alpha_j > 0$ and choose the minimum value (say x_k/a_k).

STEP 3: Reduce the value of the variable corresponding to the minimum ratio $\left(x_k/a_k\right)$ to zero.

STEP 4: The values of new variables are

$$x_j^{'} = x_j - \left(\frac{x_k}{\alpha_k}\right)\alpha_j.$$

Example 2.15. Let, $x_1 = 2$, $x_2 = 4$, and $x_3 = 1$ be a feasible solution to the following system of equations

$$2x_1 + x_2 + 2x_3 = 2 \tag{2.74}$$

$$x_1 + 4x_2 = 18 \tag{2.75}$$

Reduce the given feasible solution to a basic feasible solution.

Solution: The given system of equations can be represented as the matrix equation

$$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 4 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 18 \end{bmatrix} \tag{2.76}$$

Here the coefficient matrix $A = [v_1 v_2 v_3] = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 4 & 0 \end{bmatrix}$; $b = \begin{bmatrix} 2 \\ 18 \end{bmatrix}$ and $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$.

As the rank of A is 2, we may write to anyone vector $v_i, i = 1, 2, 3$ as a linear combination of the other two. Let us consider $v_3 = \alpha_1 v_1 + \alpha_2 v_2$. That is,

$$\begin{bmatrix} 2 \\ 0 \end{bmatrix} = \alpha_1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 4 \end{bmatrix}. \tag{2.77}$$

After solving (2.77), we get $\alpha_1 = 8/9$ and $\alpha_2 = -2/9$.

Now (2.77) can also be written as

$$\frac{8}{9}\begin{bmatrix} 2 \\ 1 \end{bmatrix} - \frac{2}{9}\begin{bmatrix} 1 \\ 4 \end{bmatrix} - \begin{bmatrix} 2 \\ 0 \end{bmatrix} = 0. \tag{2.78}$$

where, $\alpha_1 = 8/9$, $\alpha_2 = -2/9$, and $\alpha_3 = -1$.

Now $\min\left\{\frac{x_j}{\alpha_j}; \alpha_j > 0\right\} = \left\{\frac{2}{\frac{8}{9}}, \frac{4}{-\frac{2}{9}}, \frac{1}{-1}\right\} = \frac{9}{4}$.

Hence, the new variables are

$$x_1^{'} = 2 - \left(\frac{9}{4}\right)\frac{8}{9} = 0,$$

$$x_2^{'} = 4 - \left(\frac{9}{4}\right)\left(-\frac{2}{9}\right) = \frac{9}{2},$$

$$x_3' = 1 - \left(\frac{9}{4}\right)(-1) = \frac{13}{4}. \tag{2.79}$$

Therefore the required basic feasible solution is $x_1 = 0$, $x_2 = 9/2$, and $x_3 = 13/4$.

2.4.2 Working procedure of the simplex method

In this section the only maximization type LPP has been discussed. The minimization type LPP can be handled by changing the objective functions that are also included here. Whereas the general procedure to solve both minimization and maximization types of LPP is presented in Fig. 2.7.

Following are the steps of the simplex method to find the optimal solution for maximization type LPP.

STEP 1: Check whether the objective function of the LPP is to be maximized or minimized.

If $z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$ is to be minimized, then convert it into a problem of maximization by writing

Minimize Z = Maximize $(-Z)$

Check whether all bs (right side constant of the constraints of LPP) are positive.

If any of the b_i is negative, multiply both sides of that constraint by -1 to make its right-hand side positive.

STEP 2: Express the LPP in the standard form.

Convert all inequalities of constraints into equations by introducing slack/surplus variables in the constraints giving equations of the form (2.50) and (2.52).

STEP 3: Find an initial basic feasible solution.

If the LPP involves m equations and n unknowns, where $(n > m)$, then assign zero values to any $(n - m)$ of the variables for finding a solution. Start with a basic solution for which x_j, $j = 1, 2, \ldots, (n - m)$ are each zero. Find all s_i values are zero. Then the solution is degenerate. The above information is expressed in Table 2.8.

Here the variables x_1, x_2, x_3, etc. are called nonbasic variables and variables s_1, s_2, s_3, etc. are called basic variables.

Refer to the basic variables s_1, s_2, s_3, etc. and c_j row denotes the coefficients of the variables in the objective function.

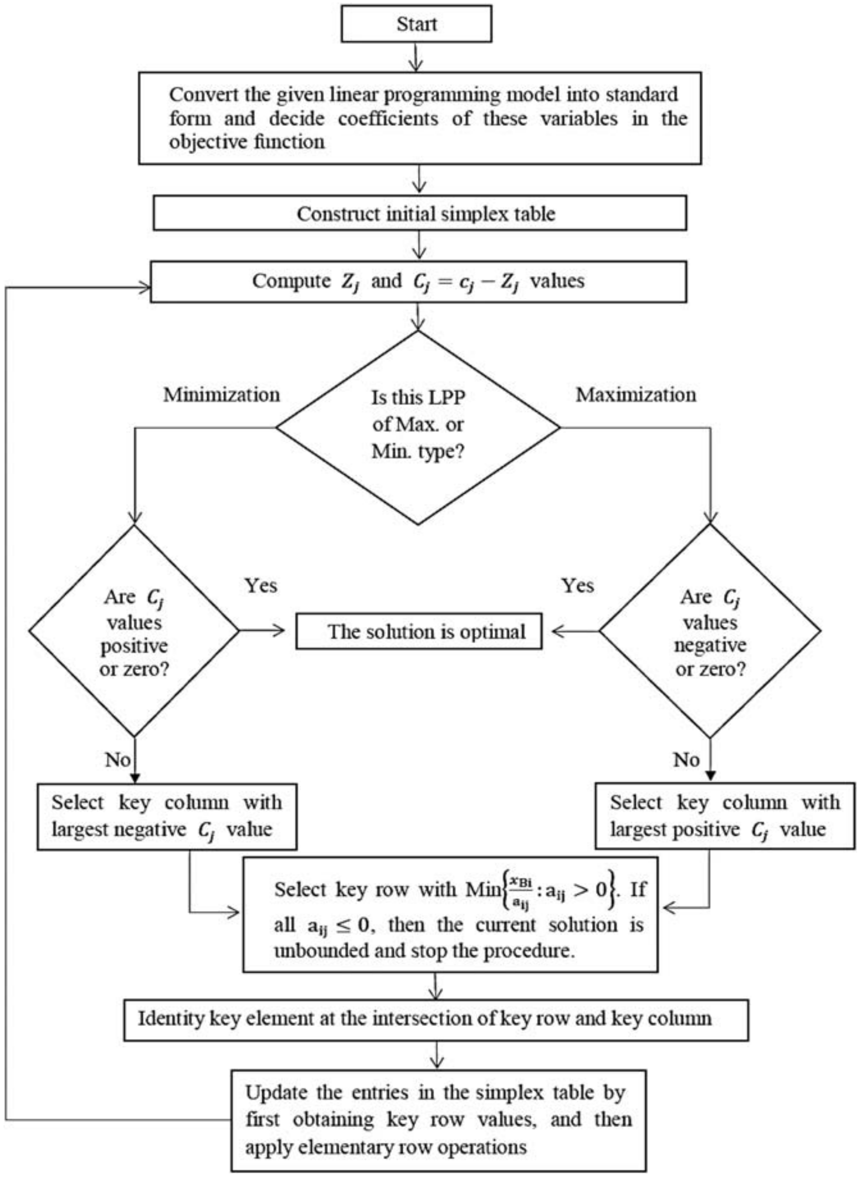

Start
Convert the given linear programming model into standard form and decide coefficients of these variables in the objective function
Construct initial simplex table
Compute Z_j and $C_j = c_j - Z_j$ values
Is this LPP of Max. or Min. type?
Minimization
Maximization
Are C_j values positive or zero?
Are C_j values negative or zero?
Yes
Yes
The solution is optimal
No
No
Select key column with largest negative C_j value
Select key column with largest positive C_j value
Select key row with $\text{Min}\left\{\frac{x_{Bi}}{a_{ij}} : a_{ij} > 0\right\}$. If all $a_{ij} \leq 0$, then the current solution is unbounded and stop the procedure.
Identity key element at the intersection of key row and key column
Update the entries in the simplex table by first obtaining key row values, and then apply elementary row operations

Figure 2.7 Flow diagram of the simplex method.

The c_B column denotes coefficients of the basic variables only in the objective function.

The b column denotes values of the basic variables while remaining variables will always be zero.

Table 2.8 The general structure of the simplex table.

C_B	c_j	c_1	c_2	c_3	$\dots$	**0**	**0**	**0**	$\dots$	
	Basis	x_1	x_2	x_3	$\dots$	s_1	s_2	s_3	$\dots$	b
0	s_1	a_{11}	a_{12}	a_{13}	$\dots$	1	0	0	$\dots$	
0	s_2	a_{21}	a_{22}	a_{23}	$\dots$	0	1	0	$\dots$	
0	s_3	a_{31}	a_{32}	a_{33}	$\dots$	0	0	1	$\dots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$
				Body matrix				Unit matrix		

The coefficients of x's (decision variables) in the constraint equations constitute the body matrix while coefficients of slack variables constitute the unit matrix.

STEP 4: Apply the optimality test.

Compute $C_j = c_j - Z_j$, where $Z_j = \sum c_B a_{ij}, i = 1, 2, \cdots, m$.

C_j is called the net evaluation row and indicates the per unit increase in the objective function if the variable heading the column is brought into the solution.

If all C_j are negative, then the initial basic solution is optimal.

If even one C_j is positive, then the current feasible solution is not optimal (that is, it can be improved) and proceed to the next step.

STEP 5: Identify the incoming and outgoing variables.

If there are more than one positive C_j, then the incoming variable is the one that heads the column containing maximum C_j. The column containing it is known as the key column which is shown marked with an arrow at the bottom. If more than one variable has the same maximum C_j, any of these variables may be selected arbitrarily as the incoming variable.

Then divide the elements under b-column by the corresponding elements of the key column and choose the row containing the minimum positive ratio θ. Then replace the corresponding basic variable (by making its value zero). It is termed as the outgoing variable.

The corresponding row is called the key row which is marked with an arrow on its right end. The element at the intersection of the key row and key column is called the key element, which is bracketed. If all these ratios are less than equal to zero (≤ 0), the incoming variable can be made as large as we want without violating the feasibility condition. Hence the problem has an unbounded solution and no further iteration is required.

STEP 6: Iterate toward an optimal solution

Drop the outgoing variable and introduce the incoming variable along with its associated value under c_B column. Convert the key element to unity by dividing the key row by the key element. Then make all other elements of the key column zero by subtracting proper multiples of the key row from the other rows.

STEP 7: Continuation of the process

Go to STEP 4 and repeat the procedure until either an optimal (or an unbounded) solution is obtained.

Example 2.16. Using the simplex method

$$\text{Maximize } Z = 5x_1 + 3x_2 \tag{2.80}$$

subject to the constraints

$$x_1 + x_2 \leq 2, \tag{2.81}$$

$$5x_1 + 2x_2 \leq 10, \tag{2.82}$$

$$3x_1 + 8x_2 \leq 12, \tag{2.83}$$

$$x_1, x_2 \geq 0. \tag{2.84}$$

Solution:

STEP 1: Here already the objective function is asked maximization type and all b values are also positive.

STEP 2: By using the slack variables s_1, s_2, and s_3, the LPP can be converted into the following standard form

$$\text{Maximize } Z = 5x_1 + 3x_2 + 0s_1 + 0s_2 + 0s_3 \tag{2.85}$$

subject to

$$x_1 + x_2 + s_1 + 0s_2 + 0s_3 = 2 \tag{2.86}$$

$$5x_1 + 2x_2 + 0s_1 + s_2 + 0s_3 = 10 \tag{2.87}$$

$$3x_1 + 8x_2 + 0s_1 + 0s_2 + s_3 = 12 \tag{2.88}$$

$$x_1, x_2, s_1, s_2, s_3 \geq 0. \tag{2.89}$$

STEP 3: Here we have three equations with five unknowns and for obtaining a solution, we have to assign zero values to any two of the variables.

We will start with a basic solution for which we set $x_1 = 0$ and $x_2 = 0$. Substituting $x_1 = x_2 = 0$ in (2.86), (2.87) and (2.88), we get the basic solution $s_1 = 2, s_2 = 10, s_3 = 12$.

Since all s_1, s_2, and s_3 are positive, the basic solution is feasible and nondegenerate.

An initial basic feasible solution is given in the following table.

	c_j	**5**	**3**	**0**	**0**	**0**		
c_B	**Basis**	x_1	x_2	s_1	s_2	s_3	b	θ
0	s_1	**(1)**	1	1	0	0	2	$\frac{2}{1} \leftarrow$
0	s_2	5	2	0	1	0	10	$\frac{10}{5}$
0	s_3	3	8	0	0	1	12	$\frac{12}{3}$
	$Z_j = \sum c_B a_{ij}$	0	0	0	0	0	0	
	$C_j = c_j - Z_j$	**5**	3	0	0	0		
		$\uparrow$						

For x_1 column, that is, $j = 1$, $Z_j = \sum c_B a_{i1} = 0(1) + 0(5) + 0(3) = 0$ and for x_2 column, that is, $j = 2$, $Z_j = \sum c_B a_{i2} = 0(1) + 0(2) + 0(8) = 0$. Similarly $Z_j(b) = 0(2) + 0(10) + 0(12) = 0$.

STEP 4: As C_j is positive under some columns, the initial basic feasible solution is not optimal (that is, it can be improved) and we proceed to the next step.

STEP 5: This table shows that x_1 is the incoming variable as its incremental contribution $C_j = 5$ is maximum and the column in which it appears is the key column (shown with an arrow at the bottom).

Dividing the elements under b-column by the corresponding elements of the key column, we find a minimum positive ratio θ is 2 in two different rows. So we will choose arbitrarily the row containing s_1 as the key row (shown with an arrow on its right end). The element at the intersection of the key row and the key column that is **(1)**, is the key element. Therefore s_1 is the outgoing basic variable that will now become nonbasic.

Having decided that x_1 is to enter the solution, we have tried to find as to what maximum value x_1 could have without violating the constraints. So, removing s_1, the new basis will contain x_1, s_2, and s_3 as the basic variables.

To transform the initial set of equations with a basic feasible solution into an equivalent set of equations with a different basic feasible solution, we make key element unity. Here the key element being unity, we retain the key role as it is. Then to make all other elements in key column zero, we subtract proper multiples of the key row from the other rows.

Here we subtract 5 times the elements of the key row from the second row and 3 times the elements of the key row from the third row. These become the second and the third rows of the next table. We also change the corresponding value underc_B column from 0 to 5, while replacing s_1 by x_1 under the basis. Thus the second basic feasible solution is given in the following table.

	c_j	5	3	0	0	0		
c_B	Basis	x_1	x_2	s_1	s_2	s_3	b	θ
5	x_1	1	1	1	0	0	2	
0	s_2	0	-3	-5	1	0	0	
0	s_3	0	5	-3	0	1	6	
	$Z_j = \sum c_B a_{ij}$	5	5	5	0	0	10	
	$C_j = c_j - Z_j$	0	-2	-5	0	0		

As C_j is zero or negative under all columns, this table gives the optimal basic feasible solution. This optimal solution is $x_1 = 2,\ x_2 = 0$ and maximum $Z = 10$.

Example 2.17. A firm produces three items that are processed on three machines. The relevant data are given below.

Machine	Time per unit (in minutes)			Machine capacity (minutes per day)
	Item *A*	Item *B*	Item *C*	
M_1	2	3	2	440
M_2	4	–	3	470
M_3	2	5	–	430

The profit per unit for items *A*, *B*, and *C* is ₹4, ₹3, and ₹6, respectively. Determine the daily number of units to be manufactured for each product. Assume that all the units produced are consumed in the market.

Solution: Let us consider the firm produces x_1, x_2, and x_3 units of items *A*, *B*, and *C*, respectively. Then we can form the following linear programming model.

$$\text{Maximize } Z = 4x_1 + 3x_2 + 6x_3$$

subject to the constraints

$$2x_1 + 3x_2 + 2x_3 \le 440$$

$$4x_1 + 3x_3 \le 470$$

$$2x_1 + 5x_2 \le 430$$

$$x_1, x_2, x_3 \ge 0.$$

By introducing the slack variables s_1, s_2, and s_3, the standard form of LPP becomes

Maximize $Z = 4x_1 + 3x_2 + 6x_3 + 0s_1 + 0s_2 + 0s_3$subject to

$$2x_1 + 3x_2 + 2x_3 + s_1 + 0s_2 + 0s_3 = 440$$

$$4x_1 + 0x_2 + 3x_3 + 0s_1 + s_2 + 0s_3 = 470$$

$$2x_1 + 5x_2 + 0x_3 + 0s_1 + 0s_2 + s_3 = 430$$

$$x_1, x_2, x_3, s_1, s_2, s_3 \ge 0.$$

Now the initial simplex table can be constructed as

	c_j	**4**	**3**	**6**	**0**	**0**	**0**		
c_B	**Basis**	x_1	x_2	x_3	s_1	s_2	s_3	b	θ
0	s_1	2	3	2	1	0	0	440	$\frac{440}{2}$
0	s_2	4	0	**(3)**	0	1	0	470	$\frac{\mathbf{470}}{\mathbf{3}} \leftarrow$
0	s_3	2	5	0	0	0	1	430	$\frac{430}{0}$
	$Z_j = \sum c_B a_{ij}$	0	0	0	0	0	0	0	
	$C_j = c_j - Z_j$	4	3	**6**	0	0	0		
				$\uparrow$					

Next the first improved simplex table can be constructed as

	c_j	**4**	**3**	**6**	**0**	**0**	**0**		
c_B	**Basis**	x_1	x_2	x_3	s_1	s_2	s_3	b	θ
0	s_1	$-\frac{2}{3}$	3	0	1	$-\frac{2}{3}$	0	$\frac{380}{3}$	$\frac{380}{9} \leftarrow$
6	x_3	$\frac{4}{3}$	0	1	0	$\frac{1}{3}$	0	470/3	∞
0	s_3	2	5	0	0	0	1	430	86
	$Z_j = \sum c_B a_{ij}$	8	0	6	0	2	0	940	
	$C_j = c_j - Z_j$	-4	3	0	0	-2	0		
			$\uparrow$						

The second improved simplex table can be constructed as

	c_j	4	3	6	0	0	0		
c_B	Basis	x_1	x_2	x_3	s_1	s_2	s_3	b	θ
3	x_2	$-\frac{2}{9}$	1	0	$\frac{1}{3}$	$-\frac{2}{9}$	0	$\frac{380}{9}$	
6	x_3	$\frac{4}{3}$	0	1	0	$\frac{1}{3}$	0	$\frac{470}{3}$	
0	s_3	$\frac{28}{9}$	0	0	$-\frac{5}{3}$	$\frac{10}{9}$	0	$\frac{1970}{9}$	
	$Z_j = \sum c_B a_{ij}$	$\frac{22}{3}$	3	6	1	$\frac{4}{3}$	0	$\frac{3200}{3}$	
	$C_j = c_j - Z_j$	$-\frac{10}{3}$	0	0	-1	$-\frac{4}{3}$	0		

Here each $C_j \le 0$, hence it gives the optimal solution $x_1 = 0$, $x_2 = 380/9$ and $x_3 = 470/3$. The optimal value of Z, that is $Z_{\max} = 3200/3 =$ ₹1066.67.

Practice set 2.4

1. Compute all the basic feasible solutions to the system of equations
   $x_1 + 2x_2 + x_3 = 4$ and $2x_1 + x_2 + 5x_3 = 5$.
   Are the solutions degenerate?
2. Find all the basic feasible solutions to be the system of linear equations
   $4x_1 + 2x_2 + x_3 = 7$ and $-x_1 + 4x_2 + 2x_3 = 14$.
3. Is the solution $x_1 = 1$, $x_2 = 1/3$, $x_3 = x_4 = 0$ a basic solution of the equations
   $x_1 + 2x_2 + x_3 + x_4 = 2$ and $x_1 + 2x_2 + 1/2x_3 + x_5 = 2$.
4. Show that the feasible solution $x_1 = 1$, $x_2 = 0$, and $x_3 = 1$, the Z value 6 to the system of equations, $x_1 + x_2 + x_3 = 2$ and $x_1 - x_2 + x_3 = 2$, with maximum $Z = 2x_1 + 3x_2 + 4x_3$ is not basic.
5. A manufacturer of leather belts makes three types of belts B_1, B_2, and B_3, which are processed to three machines M_1, M_2, and M_3. Belt B_1 requires 2 hours on the machine M_1, 3 hours on machine M_2, and 2 hours on the machine M_3. Belt B_2 requires 2 hours on the machine M_1, 2 hours on the machine M_2, and 2 hours on the machine M_3, and Belt B_3 requires 5 hours on the machine M_2 and 4 hours on the machine M_3. There are 8 hours per day available on the machine M_1, 10 hours per day available on the machine M_2, and 15 hours per day available on the machine M_3. The profit gained from the belt B_1 is ₹3.00 per unit, B_2 is ₹5.00 per unit and B_3 is ₹4.00 per unit. What should be the daily production of each type of belt so that the profit is maximum?

6. An animal feed company must produce 200 kg of a mixture consisting of ingredient F_1 and F_2 daily. F_1 costs ₹3 per kg and F_2 costs ₹8 per kg. Not more than 80 kg of F_1 can be used and at least 60 kg of F_2 must be used. Find how much of each ingredient should be used if the company wants to minimize costs.

7. A manufacturer has two products P_1 and P_2, both of which are produced in two steps by machines M_1 and M_2. The processing time per hundred for the products on machines are

	M_1	M_2	**Contribution (per 100 units)**
P_1	4	5	10
P_2	5	2	5
Available time (hours)	100	80	

The manufacturer is in a market upswing and can sell as much as he can produce both products. Formulate the mathematical model and determine product mix, using the simplex method.

8. Solve the following LPP using the simplex method.

a. Maximize $Z = x_1 - 3x_2 + 2x_3$
subject to

$$3x_1 - x_2 + 3x_3 \le 7$$

$$-2x_1 + 4x_2 \le 12$$

$$-4x_1 + 3x_2 + 8x_3 \le 10$$

$$x_1, x_2, x_3 \ge 0.$$

b. Maximize $Z = 3x_1 - x_2$
subject to

$$2x_1 + x_2 \le 2$$

$$x_1 + 3x_2 \ge 3$$

$$x \le 4$$

$$x_1, x_2 \ge 0.$$

c. Maximize $Z = 4x_1 + 5x_2 + 9x_3 + 11x_4$
subject to

$$x_1 + x_2 + x_3 + x_4 \le 15$$

$$7x_1 + 5x_2 + 3x_3 + 2x_4 \leq 120$$

$$3x_1 + 5x_2 + 10x_3 + 15x_4 \leq 100$$

$$x_1, x_2, x_3, x_4 \geq 0.$$

a. Maximize $Z = 107x_1 + x_2 + 2x_3$
subject to

$$14x_1 + x_2 - 6x_3 + 3x_4 \leq 7$$

$$16x_1 + \frac{1}{2}x_2 + 6x_3 \leq 5$$

$$3x_1 - x_2 - x_3 \leq 10$$

$$x_1, x_2, x_3, x_4 \geq 0.$$

b. Maximize $Z = 4x_1 + x_2 + 3x_3 + 5x_4$
subject to

$$4x_1 - 6x_2 - 5x_3 + 4x_4 \geq -20$$

$$3x_1 - 2x_2 + 4x_3 + x_4 \leq 10$$

$$8x_1 - 3x_2 + 3x_3 + 2x_4 \leq 20$$

$$x_1, x_2, x_3, x_4 \geq 0.$$

9. A furniture manufacturer produces and sells desks, chairs, and bookshelves. They have no difficulty in selling their items. However, limited availability of machine time, worker, and floor space restrict production. Data on the usage of resources, supplies, and profits on items are given in the following table.

	Desk	Chairs	Bookshelf	Supply
Machine hours per unit	8	4	5	1000 hours
Worker hours per unit	5	3	3	650 h
Floor space sq. ft. per unit	9	6	9	1260 sq. ft.
Contribution (in ₹) per unit	2700	1440	2250	

Formulate this problem as an LPP and obtain the optimal solution by using the simplex method.

10. A contractor undertakes three types of buildings, namely, A, B, and C. his profit margins are ₹50,000, ₹45,000, and ₹35,000 from each unit types A, B, and C buildings, respectively. The materials required for each unit are given in the following table.

Type of building	Brick required	Steel required (ton)	Concrete required (m^3)
A	50,000	3	10
B	40,000	4.5	20
C	30,000	5	25

The contractor can mobilize only 450,000 bricks, 40 tonnes of steel, and 30 m^3 concrete from his resources. Determine how many buildings of each type he should undertake at any time to make maximum profit by using the simplex method.

2.5 Artificial variable techniques

In the previous section we observed that the introduction of slack/surplus variables provides the initial basic feasible solution. But there are many LPPs where the constraints are of $(\geq)$ or $(=)$ type and slack variables fail to give such a solution. In this context there may be mentioned two methods, such as Big M **Penalty (Big M) method and Two-phase method.**

2.5.1 Big M method

Following are the steps of Big M method.

STEP 1: Express the problem in a standard LPP form.
STEP 2: We can convert the $(\geq)$ or $(=)$ type of inequalities by adding nonnegative variables to the left-hand side of all those constraints. These nonnegative variables are called artificial variables that help to obtain the initial basic feasible solution. But the introduction of artificial variables violates the corresponding constraints. So to get rid of these variables as well as not allowing them to appear in the final solution we have to assign a very large penalty $(-M)$ to these artificial variables in the objective function.
STEP 3: Then solve the modified LPP by the simplex method. But we may get the following three cases while performing the simplex method.

1. If there are no artificial variables in the basis and the optimality condition is satisfied, then the solution is an optimal basic feasible solution to the problem.
2. If there is at least one artificial variable in the basis with zero value in b column and the optimality condition is satisfied, then the solution is a degenerate optimal basic feasible solution.

3. If there is at least one artificial variable in the basis with a positive value in *b* column and the optimality condition is satisfied, then the problem has no feasible solution. The final solution is not optimal since the objective function contains an unknown quantity M. Such a solution satisfies the constraints but does not optimize the objective function and the same is called a pseudo optimal solution.

STEP 4: Continue the simplex method until an optimal feasible solution is obtained or an unbounded solution is indicated.

Form the above-discussed steps we can observe the listed points below.

1. The artificial variable serves as a computational tool to get the starting solution and once it leaves the basis then their job is over. Then the corresponding column for this variable is omitted from the next simplex table.

2. When two or more variables have the same largest positive value in C_j then the choice of an incoming variable is arbitrary among all the incoming variables.

3. Similarly when two or more variables have the same least positive ratio under the column θ, then the choice of outgoing (leaving) variable is arbitrary among all the outgoing (leaving) variables.

4. If we get zero value for the same least positive ratio under the column θ two or more times in any iteration (simplex table), then the simplex method fails and we have to adapt the following degeneracy technique.

Degeneracy: In this technique we have to divide each element of the rows that have the same values (tied rows) by the positive coefficients of the key column in that row. Then compare the resulting ratios (from left to right) first of unit matrix and then of the body matrix column by column. Finally the outgoing variable lies in that row which first contains the smallest ratio value.

The above-mentioned penalty (Big M) method is now demonstrated using the following example problem.

Example 2.18. Maximize $Z = 3x_1 + 2x_2$, subject to the constraints

$$2x_1 + x_2 \leq 2$$

$$3x_1 + 4x_2 \geq 12$$

$$x_1, x_2 \geq 0.$$

Solution: Using STEP 1, we may get the following standard form
Maximize $Z = 3x_1 + 2x_2 + 0s_1 + 0s_2 - MA$,
subject to

$$2x_1 + x_2 + s_1 + 0s_2 + 0A = 2$$

$$3x_1 + 4x_2 + 0s_1 - s_2 + A = 12$$

$$x_1, x_2, s_1, A \geq 0,$$

where A is the artificial variable.

Since the surplus variable s_2 is giving value -12 and not a basic variable that is not feasible, s_2 must be omitted from appearing in the initial solution. Now by taking $s_2 = 0$, the other nonbasic variables x_1 and x_2 are zero and we get the initial basic feasible solution as $x_1 = x_2 = s_2 = 0$, $s_1 = 2$, and $A = 12$. Now the initial simplex table is

	c_j	3	2	0	0	$-M$		
c_B	Basis	x_1	x_2	s_1	s_2	A	b	θ
0	s_1	2	1	1	0	0	2	2 $\leftarrow$
$-M$	A	3	4	0	-1	1	12	3
	$Z_j = \sum c_B a_{ij}$	$-3M$	$-4M$	0	M	$-M$	$-12M$	
	$C_j = c_j - Z_j$	$3 + 3M$	$2 + 4M$	0	$-M$	0		
			$\uparrow$					

Since C_j is positive under some columns, then it is not an optimal solution. Now the improved table is

	c_j	3	2	0	0	$-M$	
c_B	Basis	x_1	x_2	s_1	s_2	A	b
2	x_2	2	1	1	0	0	2
$-M$	A	-5	0	-4	-1	1	4
	$Z_j = \sum c_B a_{ij}$	$4 + 5M$	2	$2 + 4M$	M	$-M$	$4 - 4M$
	$C_j = c_j - Z_j$	$-(1 + 5M)$	0	$-(2 + 4M)$	$-M$	0	

Since each C_j value is negative and an artificial variable appears on the basis, there exists a pseudo optimal solution.

2.5.2 Two-phase method

In this method, the problem with artificial variables is solved in two phases which are included below.

Phase 1

STEP 1: By incorporating slack, surplus, and artificial variables express the given LPP in the standard form.

STEP 2: Then by assigning (-1) to each of the artificial variables A_i and zero to all other variables, formulate an artificial objective function $\hat{Z} = -A_1 - A_2 \ldots - A_m$.

STEP 3: Now maximize $\hat{Z}$ subject to the constraints of the given problem using the simplex method. While solving in a simplex method the following cases arise.

1. If maximum $\hat{Z} < 0$ and at least one artificial variable appears in the optimal basis at a positive level, then the original problem does not possess any feasible solution and the procedure stops here.
2. If maximum $\hat{Z} = 0$ and no artificial variable appears on an optimal basis, then a basic feasible solution is obtained and Phase 2 comes into the picture for finding the optimal basic feasible solution to the given problem.
3. If maximum $\hat{Z} = 0$ and at least one artificial variable appear on the optimal basis at the initial level, then a feasible solution to the auxiliary LPP is also a feasible solution to the given problem with all artificial variables equals zero.

To get a basic feasible solution we have to extend Phase 1 for pushing all the artificial variables out of the basis (without proceeding on to Phase 2).

Phase 2

After getting the basic feasible solution in Phase 1 it is used as the initial solution for the initial simplex table of Phase 2 and the artificial objective function is replaced by the given objective function. Then we find an optimal solution.

Example 2.19. Using the two-phase method,

Minimize $Z = \frac{15}{2}x_1 - 3x_2$
subject to

$$3x_1 - x_2 - x_3 \geq 3$$

$$x_1 - x_2 + x_3 \geq 2$$

$$x_1, x_2, x_3 \geq 0.$$

Solution:

Phase 1

Using STEP 1, we can write the standard form as

Maximum $\hat{Z} = 0x_1 + 0x_2 + 0x_3 + 0s_1 + 0s_2 - A_1 - A_2$

subject to

$$3x_1 - x_2 - x_3 - s_1 + 0s_2 + A_1 + 0A_2 = 3$$

$$x_1 - x_2 + x_3 + 0s_1 - s_2 + 0A_1 + A_2 = 2$$

$$x_1, x_2, x_3, s_1, s_2, A_1, A_2 \geq 0.$$

Now taking $x_1 = x_2 = x_3 = s_1 = s_2 = 0$, we get $A_1 = 3$, $A_2 = 2$, and $\hat{Z} = -5$.

The initial simplex table is

	c_j	**0**	**0**	**0**	**0**	**0**	**−1**	**−1**		
c_B	**Basis**	x_1	x_2	x_3	s_1	s_2	A_1	A_2	b	θ
−1	A_1	(3)	−1	−1	−1	0	1	0	3	1←
−1	A_2	1	−1	1	0	−1	0	1	2	2
	$Z_j = \sum c_B a_{ij}$	−4	2	0	1	1	−1	−1	−5	
	$C_j = c_j - Z_j$	4	−2	0	−1	−1	0	0		
		↑								

Since C_j is positive corresponding to x_1 column, this solution is not optimal.

The first improved simplex table is

	c_j	**0**	**0**	**0**	**0**	**0**	**−1**	**−1**		
c_B	**Basis**	x_1	x_2	x_3	s_1	s_2	A_1	A_2	b	θ
0	x_1	1	$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	0	$\frac{1}{3}$	0	1	−3
−1	A_2	0	$-\frac{2}{3}$	$\left(\frac{4}{3}\right)$	$\frac{1}{3}$	−1	$-\frac{1}{3}$	1	1	$\frac{3}{4}$←
	$Z_j = \sum c_B a_{ij}$	0	$\frac{2}{3}$	$-\frac{4}{3}$	$-\frac{1}{3}$	1	$\frac{1}{3}$	−1	−1	
	$C_j = c_j - Z_j$	0	$-\frac{2}{3}$	$\frac{4}{3}$	$\frac{1}{3}$	−1	$-\frac{1}{3}$	0		
				↑						

Since C_j is positive corresponding to x_3 and s_1 columns, this solution is not optimal.

The second improved simplex table is

	c_j	**0**	**0**	**0**	**0**	**0**	**−1**	**−1**	
c_B	**Basis**	x_1	x_2	x_3	s_1	s_2	A_1	A_2	b
0	x_1	1	$-\frac{1}{2}$	0	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{4}$
0	x_3	0	$-\frac{1}{2}$	1	$\frac{1}{4}$	$-\frac{3}{4}$	$-\frac{1}{4}$	$\frac{3}{4}$	$\frac{3}{4}$
	$Z_j=\sum c_B a_{ij}$	0	0	0	0	0	0	0	0
	$C_j=c_j-Z_j$	0	0	0	0	0	−1	−1	

Since all $C_j \le 0$ this table give an optimal solution $\hat{Z}_{max}=0$ and no artificial variable appears on the basis. Hence an optimal basic feasible solution to the auxiliary problem and therefore to the given problem is attained.

Phase 2

Maximum $Z^* = -\frac{15}{2}x_1 + 3x_2 + 0x_3 + 0s_1 + 0s_2 - 0A_1 - 0A_2$
subject to

$$3x_1 - x_2 - x_3 - s_1 + 0s_2 + A_1 + 0A_2 = 3$$

$$x_1 - x_2 + x_3 + 0s_1 - s_2 + 0A_1 + A_2 = 2$$

$$x_1, x_2, x_3, s_1, s_2, A_1, A_2 \ge 0.$$

The optimal initial feasible solution thus obtained will be an optimal basic feasible solution to the given LPP.

Using the final table of Phase 1, the initial simplex table of Phase 2 is

	c_j	$-\frac{15}{2}$	**3**	**0**	**0**	**0**	
c_B	**Basis**	x_1	x_2	x_3	s_1	s_2	b
$-\frac{15}{2}$	x_1	1	$-\frac{1}{2}$	0	$-\frac{1}{4}$	$-\frac{1}{4}$	$\frac{5}{4}$
0	x_3	0	$-\frac{1}{2}$	1	$\frac{1}{4}$	$-\frac{3}{4}$	$\frac{3}{4}$
	$Z_j=\sum c_B a_{ij}$	$-\frac{15}{2}$	$\frac{15}{4}$	0	$\frac{15}{8}$	$\frac{15}{8}$	$-\frac{75}{8}$
	$C_j=c_j-Z_j$	0	$-\frac{3}{4}$	0	$-\frac{15}{8}$	$-\frac{15}{8}$	

Since all $C_j \le 0$, this solution is optimal. Therefore an optimal basic feasible solution to the given problem is $x_1 = 5/4$, $x_2 = 0$, $x_3 = 3/4$, and min $Z = 75/8$.

Practice set 2.5

1. Using penalty (Big M) method, maximize $Z = 2x_1 + 3x_2 + 4x_3$, subject to the constraints $3x_1 + x_2 + 4x_3 \le 600$, $2x_1 + 4x_2 + 2x_3 \ge 480$, $2x_1 + 3x_2 + 3x_3 = 540$, and $x_1, x_2, x_3 \ge 0$.

2. Using penalty (Big M) method, maximize $Z = 300x + 400y - 600z$, subject to the constraints $z \leq 3000$, $z + 1/3t \leq 4500$, $x - 2z = 0$, $y - z = 0$, and $x, y, z, t \geq 0$.
3. Using penalty (Big M) method, minimize $Z = 9x_1 + 10x_2$, subject to the constraints $2x_1 + 4x_2 \geq 50$, $4x_1 + 3x_2 \geq 24$, $3x_1 + 2x_2 \geq 60$, and $x_1, x_2 \geq 0$.
4. Using penalty (Big M) method, minimize $Z = 30x_1 + 30x_2 + 10x_3$, subject to the constraints $2x_1 + x_2 + x_3 \geq 6$, $4x_1 + x_2 + 2x_3 \leq 8$, and $x_1, x_2, x_3 \geq 0$.
5. Using penalty (Big M) method, maximize $Z = 100x_1 + 300x_2$, subject to the constraints $6x_1 + 10x_2 \leq 720$, $500x_1 + 500x_2x_3 \leq 40,000$, $x_1 \geq 30$, $x_2 \geq 20$, and $x_1, x_2 \geq 0$.
6. A carpenter has started a workshop in which he manufactures handcrafts. Each cart consists of a frame and two wheels. The frame needs 3 hours and a wheel needs 2 hours of hard work of which 90 hours per week are available. The carpenter wants that he should manufacture at least 10 carts during a week. The cost of the frame is ₹1000 and that of a wheel is ₹400. Formulate this problem as a linear programming model and using the simplex method, determine an optimum production plan.
7. A marketing manager wishes to allocate his annual advertising budget ₹200,000 to two media M_1 and M_2. The unit cost of a message in media M_1 is ₹1000 and that of M_2 is ₹1500. Media M_1 is a monthly magazine and not more than one insertion is desired in one issue, whereas at least 5 messages should appear in media M_2. The expected audience for the unit message in M_1 is 40,000 and that of M_2 is 55,000. Develop an LPP and solve it for maximizing the total effective audience.
8. Use the two-phase method to solve the following LPP

   Maximize $Z = 5x_1 - 4x_2 + 3x_3$

   subject to the constraints

$$2x_1 + x_2 - 6x_3 = 20$$

$$6x_1 + 5x_2 + 10x_3 \leq 76$$

$$8x_1 - 3x_2 + 6x_3 \leq 50$$

$$x_1, x_2, x_3 \geq 0.$$

9. Use the two-phase method to solve the following LPP
Maximize $Z = 3x_1 + 2x_2 + 2x_3$
subject to the constraints

$$5x_1 + 7x_2 + 4x_3 \leq 7$$

$$-4x_1 + 7x_2 + 5x_3 \geq -2$$

$$x = 3x_1 + 4x_2 - 6x_3 \geq \frac{29}{7}$$

$$x_1, x_2, x_3 \geq 0.$$

10. Use the two-phase method to minimize $Z = 3x_1 + 2x_2$, subject to $2x_1 + x_2 \geq 2$, $3x_1 + 4x_2 \geq 12$, and $x_1, x_2 \geq 0$.

2.6 Duality Principle

One of the most interesting concepts in linear programming is the duality theory. Every LPP has associated with it another LPP involving the same data and closely related optimal solutions. Such two problems are an aid to be duals of each other. While one of these is called primal, the other is dual.

The importance of the duality concept is due to two main reasons. First if the primal contains a large number of constraints and a smaller number of variables, the labor of computation can be considerably reduced by converting it into the dual problem and then solving it. Second the interpretation of the dual variables from the cost or economic point of view proves extremely useful in making future decisions in the activities being programmed.

2.6.1 Formulation of a dual problem

Consider the following LPP,
Maximize $Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$,
subject to the constraints

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2$$

$$\vdots$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m$$

$$x_1, x_2, \ldots, x_n \ge 0.$$

To construct the dual problem we adopt the following guidelines.

1. The maximization problem in the primal becomes the minimization problem in the dual and vice versa.
2. $(\le)$ type of constraints in the primal become $(\ge)$ type of constraints in the dual and vice versa.
3. The coefficients $c_1, c_2, \ldots, c_n$ in the objective function of the primal become $b_1, b_2, \ldots, b_m$ in the objective function of the dual.
4. The constraints $b_1, b_2, \ldots, b_m$ in the constraints of the primal become $c_1, c_2, \ldots, c_n$ in the constraints of the dual.
5. If the primal has n variables and m constraints, the dual will have m variables and n constraints that is the transpose of the body matrix of the primal problem gives the body matrix of the dual.
6. The variables in both the primal and dual are nonnegative.

Then the dual problem will be
Minimize $W = b_1y_1 + b_2y_2 + \ldots + b_my_m$
Subject to the constraints

$$a_{11}y_1 + a_{21}y_2 + \ldots + a_{m1}y_m \ge c_1$$

$$a_{12}y_1 + a_{22}y_2 + \ldots + a_{m2}y_m \ge c_2$$

$$\vdots$$

$$\vdots$$

$$a_{1n}y_1 + a_{2n}y_2 + \ldots + a_{mn}y \ge c_n$$

$$y_1, y_2, \ldots, y_m \ge 0.$$

Example 2.20. Construct the dual of the following problem.

Maximize $Z = 3x_1 + 5x_2$
subject to

$$-x_1 + 2x_2 \le 2$$

$$x_1 + 3x_2 \leq 4$$

$$x_1 \leq 3$$

$$x_1, x_2 \geq 0.$$

Solution: Using the dual problem technique the dual problem of the given primal is

Minimize $Z' = 2y_1 + 4y_2 + 3y_3$

Subject to

$$-y_1 + y_2 + y_3 \geq 3$$

$2y_1 + 3y_2 \geq 5$

$$y_1, y_2 \geq 0.$$

2.6.1.1 Formulation of a dual problem when the primal has equality constraints

Consider the problem

Maximize $Z = c_1x_1 + c_2x_2$

subject to

$$a_{11}x_1 + a_{12}x_2 = b_1$$

$$a_{21}x_1 + a_{22}x_2 \leq b_2$$

$$x_1, x_2 \geq 0.$$

The equality constraint can be written as

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$

$$a_{11}x_1 + a_{12}x_2 \geq b_1$$

Or

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$

$$-a_{11}x_1 - a_{12}x_2 \leq -b_1$$

Then the above problem can be restated as

Maximize $Z = c_1x_1 + c_2x_2$

Subject to

$$a_{11}x_1 + a_{12}x_2 \leq b_1$$

$$-a_{11}x_1 - a_{12}x_2 \leq -b_1$$

$$a_{21}x_1 + a_{22}x_2 \leq b_2$$

$$x_1, x_2 \geq 0.$$

Now we form the dual using y_1', y_1'', y_2 as the dual variables. Then the dual problem is

Minimize $W = b_1(y_1' - y_1'') + b_2y_2$

Subject to

$$a_{11}(y_1' - y_1'') + a_{21}y_2 \geq c_1$$

$$a_{12}(y_1' - y_1'') + a_{22}y_2 \geq c_2$$

$$y_1', y_1'', y_2 \geq 0.$$

The term $(y_1' - y_1'')$ appears in both the objective function and all the constraints of the dual. This will always happen whenever there is an equality constraint in the primal. Then the new variable $y_1' - y_1'' = y_1$ becomes unrestricted in the sign being the difference of two nonnegative variables and the above dual problem takes the form.

Minimize $W = b_1y_1 + b_2y_2$

Subject to

$$a_{11}y_1 + a_{21}y_2 \geq c_1$$

$$a_{12}y_1 + a_{22}y_2 \geq c_2$$

y_1 unrestricted in sign, $y_2 \geq 0$.

In general if the primal is

Maximize $Z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$,

Subject to the constraints

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$$

$$\vdots$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$$

$$x_1, x_2, \ldots, x_n \geq 0.$$

Then the dual problem is
Minimize $W = b_1y_1 + b_2y_2 + \ldots + b_my_m$
Subject to the constraints

$$\begin{gathered} a_{11}y_1 + a_{21}y_2 + \ldots + a_{m1}y_m \geq c_1 \\ a_{12}y_1 + a_{22}y_2 + \ldots + a_{m2}y_m \geq c_2 \\ \vdots \\ \vdots \\ a_{1n}y_1 + a_{2n}y_2 + \ldots + a_{mn}y \geq c_n \end{gathered}$$

$y_1, y_2, \ldots, y_m$ all are unrestricted in sign.

Thus the dual variables corresponding to equality constraints are unrestricted in sign. Conversely when the primal variables are unrestricted in sign, corresponding dual constraints are equalities.

Example 2.21. Construct the dual of the LPP

Maximize $Z = 3x_1 + 8x_2 + 5x_3$
Subject to

$$\begin{gathered} 2x_1 + 3x_2 + 2x_3 \leq 9 \\ 3x_1 - 2x_2 + 4x_3 = 7 \\ x_1, x_2, x_3 \geq 0. \end{gathered}$$

Solution: Let y_1 and y_2 be the dual variables associated with the first and second constraints. Then the dual problem is
Minimize $Z' = 9y_1 + 7y_2$
subject to

$$\begin{gathered} 2y_1 + 3y_2 \geq 3 \\ 3y_1 - 2y_2 \geq 8 \\ 2y_1 + 4y_2 \geq 5 \end{gathered}$$

$y_1 \geq 0$, y_2 is unrestricted in sign.

2.6.1.2 Duality principle

If the primal and the dual problems have feasible solutions then both have optimal solutions and the optimal value of the primal objective is equal to the optimal value of the dual objective function, that is

Maximize Z = Minimize W

This is the fundamental theorem of duality. It suggests that an optimal solution to the primal problem can directly be obtained from that of the dual problem and vice versa.

Working rules for obtaining an optimal solution to the primal (dual) problem from that of the dual (primal):

Suppose we have already found an optimal solution to the dual (primal) problem by the simplex method.

Rule 1: If the primal variable corresponds to a slack starting variable in the dual problem, then its optimal value is directly given by the coefficient of the slack variable with the changed sign, in the C_j row of the optimal dual simplex table and vice versa.

Rule 2: If the primal variable corresponds to an artificial variable in the dual problem, then its optimal value is directly given by the coefficient of the artificial variable, with a changed sign, in the C_j row of the optimal dual simplex table, after deleting the constant M and vice versa.

On the other hand if the primal has an unbounded solution, then the dual problem will not have a feasible solution and vice versa.

Practice set 2.6

1. Write the dual of the following problems.

a. Maximize $Z = 10x_1 + 20x_2 + 30x_3$
subject to

$$6x_1 + 5x_2 + 3x_3 \leq 25$$

$$4x_1 + 6x_2 + 7x_3 \leq 7$$

$$x_1, x_2, x_3 \geq 0.$$

b. Maximize $Z = 3x_1 + 15x_2 + 5x_3$
subject to

$$x_1 - x_2 + x_3 \geq 3$$

$$-3x_1 + 2x_3 \leq 1$$

$$2x_1 + x_2 - x_3 = 4$$

$$x_1, x_2, x_3 \geq 0.$$

c. Minimize $Z = 2x_1 + 4x_2 + 3x_3$
subject to

$$3x_1 + 4x_2 + x_3 \geq 11$$

$$-2x_1 - 3x_2 + 2x_3 \leq -7$$

$$x_1 - 2x_2 - 3x_3 \leq -1$$

$$3x_1 + 2x_2 + 2x_3 \geq 5$$

$$x_1, x_2, x_3 \geq 0.$$

d. Minimize $Z = 2x_1 + 5x_2 + 9x_3$
subject to

$$2x_1 + 3x_2 + 5x_3 \geq 2$$

$$5x_1 + 7x_2 + x_3 = 3$$

$$x_1 + 6x_2 + 4x_3 \leq 7$$

$x_1, x_2 \geq 0$, and x_3 is unrestricted.

e. Maximize $Z = x_1 + 2x_2 + 3x_3$
subject to

$$x_1 - 2x_2 + x_3 \geq 5$$

$$-3x_1 + 2x_3 \leq 7$$

$$2x_1 + x_2 - 4x_3 = 3$$

$x_1, x_2 \geq 0$, and x_3 is unrestricted.

2. Using duality, solve the following LPP.

a. Minimize $Z = 2x_1 + 9x_2 + x_3$
subject to

$$x_1 + 4x_2 + 2x_3 \geq 5$$

$$3x_1 + x_2 + 2x_3 \geq 4$$

$$x_1, x_2 \geq 0.$$

b. Maximize $Z = 3x_1 + 2x_2$
subject to

$$x_1 + x_2 \geq 1$$

$$x_1 + x_2 \leq 7$$

$$x_1 + 2x_2 \leq 10$$

$$x_2 \leq 3$$

$$x_1, x_2 \geq 0.$$

c. Maximize $Z = 3x_1 + 2x_2 + 5x_3$
subject to

$$x_1 + 2x_2 + x_3 \geq 430$$

$$3x_1 + 2x_3 \leq 460$$

$$x_1 + 4x_2 \leq 420$$

$$x_1, x_2, x_3 \geq 0.$$

2.7 Dual simplex method

We have seen that a set of basic variables giving a feasible solution can be found by introducing artificial variables and using M-method or two-phase method. Using the primal–dual relationships for a problem, we have another method (known as dual simplex method) for finding an initial feasible solution. Whereas the regular simplex method starts with a basic feasible (but non-optimal) solution and works towards optimality, the dual simplex method stats with a basic unfeasible (but optimal) solution and works toward feasibility. The dual simplex method is quite similar to the regular simplex method, the only difference lies in the criterion used for selecting the incoming variables. In the dual simplex method, we first determine the outgoing variable and then incoming variable while in the case of a regular simplex method reverse is done. The general procedure to solve both minimization and maximization type of LPP using the dual simplex method is presented in Fig. 2.8.

2.7.1 Working procedure for a dual simplex method

Step 1

1. Convert the problem to maximization form if it is not so.
2. Convert $(\geq)$ type constraints, if any to $(\leq)$ type by multiplying such constraints by -1.
3. Express the problem in standard form by introducing slack variables.

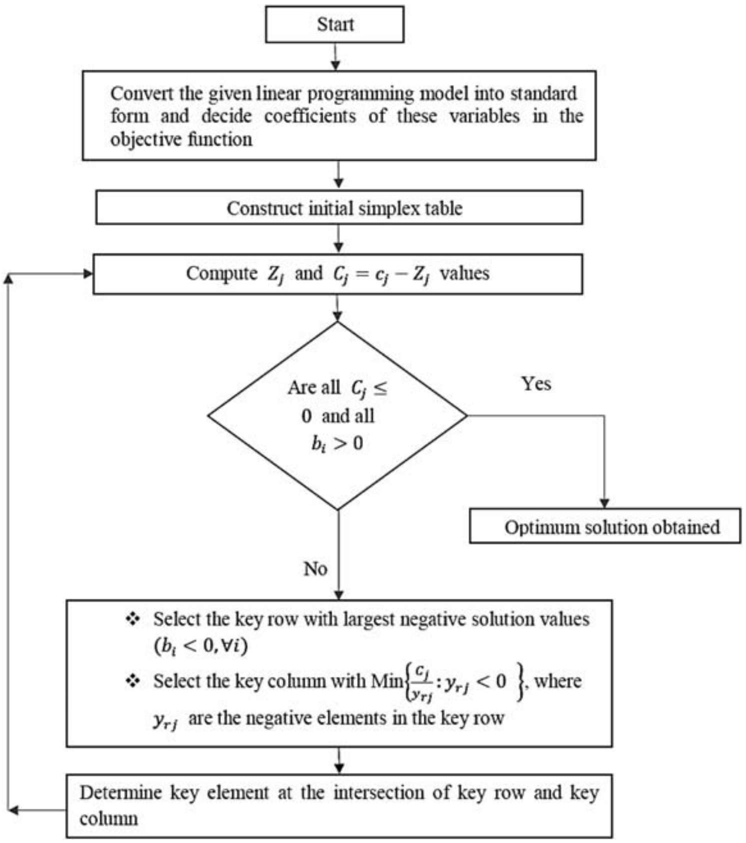

Start
Convert the given linear programming model into standard form and decide coefficients of these variables in the objective function
Construct initial simplex table
Compute Z_j and $C_j = c_j - Z_j$ values
Are all $C_j \leq 0$ and all $b_i > 0$
Yes
Optimum solution obtained
No
Select the key row with largest negative solution values $(b_i < 0, \forall i)$
Select the key column with $\text{Min}\left\{\frac{C_j}{y_{rj}} : y_{rj} < 0\right\}$, where y_{rj} are the negative elements in the key row
Determine key element at the intersection of key row and key column

Figure 2.8 Flow diagram of dual simplex method.

Step 2: Find the initial basic feasible solution and express this information in the form of a dual simplex table.

Step 3: Test the nature of $C_j = c_j - Z_j$

1. If all $C_j \leq 0$ and all $b_i \geq 0$, then the optimal basic feasible solution has been attained.
2. If all $C_j \leq 0$ and at least one $b_i < 0$, then go to Step 4.
3. If any $C_j \geq 0$, the method fails.

Step 4: Mark the outgoing variable. Select the row that contains the most negative b_i. This will be the key row and the corresponding basic variable is the outgoing variable.

Step 5: Test the nature of key row elements:

a. If all these elements are ≥ 0, the problem does not have a feasible solution.

b. If at least one element < 0, find the ratios of the corresponding elements of C_j-row to these elements. Choose the smallest of these ratios. The corresponding column is the key column and the associated variable is the incoming variable.

Step 6: Iterate toward an optimal feasible solution. Make the key element of unity. Perform row operations as in the regular simplex method and repeat iterations until either an optimal feasible solution is attained or there is an indication of a nonexistence solution.

Example 2.22. Using the dual simplex method,

maximize $Z = -3x_1 - 2x_2$
subject to

$$x_1 + x_2 \geq 1$$

$$x_1 + x_2 \leq 7$$

$$x_1 + 2x_2 \geq 10$$

$$x_2 \geq 3$$

$$x_1, x_2 \geq 0.$$

Solution: Convert the first and third constraints into $(\leq)$ type. Now these constraints become

$$-x_1 - x_2 \leq -1,$$

$$-x_1 - 2x_2 \leq -10.$$

The standard form will be
maximize $Z = -3x_1 - 2x_2 + 0s_1 + 0s_2 + 0s_3 + 0s_4$
subject to

$$-x_1 - x_2 + s_1 = -1$$

$$x_1 + x_2 + s_2 = 7$$

$$-x_1 - 2x_2 + s_3 = -10$$

$$x_2 + s_4 = 3$$

$$x_1, x_2, s_1, s_2, s_3, s_4 \geq 0.$$

To get the initial solution, we have to set the decision variables x_1 and x_2 equal to zero, then the basic solution is $x_1 = x_2 = 0$, $s_1 = -1$, $s_2 = 7$, $s_3 = -10$, $s_4 = 3$, and $Z = 0$.

The initial solution is given by the following table.

	c_j	-3	-2	0	0	0	0	
c_B	Basis	x_1	x_2	s_1	s_2	s_3	s_4	b
0	s_1	-1	-1	1	0	0	0	-1
0	s_2	1	1	0	1	0	0	7
0	s_3	-1	**(-2)**	0	0	1	0	$-10\leftarrow$
0	s_4	0	1	0	0	0	1	3
	$Z_j=\sum c_B a_{ij}$	0	0	0	0	0	0	0
	$C_j=c_j-Z_j$	-3	-2	0	0	0	0	
			$\uparrow$					

Since all C_j values are ≤ 0 and $b_1=-1$, $b_3=-10$, the initial solution is optimal but infeasible. We therefore proceed further. Since, b_3 is negative and numerically largest, the third row is the key row and s_3 is the outgoing variable.

Calculate the ratios of elements in C_j row to the corresponding negative elements of the key row. These ratios are $-3/-1=3$ and $-2/-2=1$ (neglecting ratios corresponding to positive or zero elements of the key row). Since the smaller ratio is 1, therefore x_2 column is the key column and (-2) is the key element. Drop s_3 and introduce x_2 along with its associated value -2 under c_B column. Convert the key element to unity and make all other elements of the key column zero. Then the second solution is given by the following table.

	c_j	-3	-2	0	0	0	0	
c_B	Basis	x_1	x_2	s_1	s_2	s_3	s_4	b
0	s_1	$-\frac{1}{2}$	0	1	0	$-\frac{1}{2}$	0	4
0	s_2	$\frac{1}{2}$	0	0	1	$\frac{1}{2}$	0	2
-2	x_2	$\frac{1}{2}$	1	0	0	$-\frac{1}{2}$	0	5
0	s_4	$\left(-\frac{1}{2}\right)$	0	0	0	$\frac{1}{2}$	1	$-2\leftarrow$
	$Z_j=\sum c_B a_{ij}$	-1	-2	0	0	1	0	-10
	$C_j=c_j-Z_j$	-2	0	0	0	-1	0	
		$\uparrow$						

Since all C_j values are ≤ 0 and $b_4=-2$, this solution is optimal but infeasible. We therefore proceed further. Since b_4 is negative, the fourth row is the key row and s_4 is the outgoing variable. Calculate ratios of elements in C_j row to the corresponding negative elements of the key row.

This ratio is $-2/(-1/2)=4$ (neglecting other ratios corresponding to positive or 0 elements of key row).

Therefore x_1 column is the key column and $(-1/2)$ is the key element.

Drop s_4 and introduce x_1 with its associated value -3 under the c_B column. Convert the key element to unity and make all other elements of the key column zero. Then the third solution is given in the following table.

	c_j	-3	-2	0	0	0	0	
c_B	**Basis**	x_1	x_2	s_1	s_2	s_3	s_4	b
0	s_1	0	0	1	0	-1	-1	6
0	s_2	0	0	0	1	1	1	0
-2	x_2	0	1	0	0	0	1	3
-3	x_1	1	0	0	0	-10	-2	4
	$Z_j=\sum c_B a_{ij}$	-3	-2	0	0	3	4	-18
	$C_j=c_j-Z_j$	0	0	0	0	-3	-4	

Since all C_j values are ≤ 0 and all b's are ≥ 0, therefore this solution is optimal and feasible. Thus the optimal solution is $x_1=4$, $x_2=3$, and $Z_{\max}=-18$.

Practice set 2.7

1. Use the dual simplex method to solve the following LPP.

a. Minimize $Z=x_1+x_2$
subject to

$$2x_1+x_2\geq 2$$
$$-x_1-x_2\geq 1$$
$$x_1,x_2\geq 0.$$

b. Maximize $Z=-3x_1-2x_2$
subject to

$$x_1+x_2\geq 1$$
$$x_1+x_2\leq 7$$
$$x_1+2x_2\geq 10$$
$$x_2\leq 3$$
$$x_1,x_2\geq 0.$$

c. Maximize $Z = -2x_1 - x_3$
subject to

$$x_1 + x_2 - x_3 \geq 5$$

$$x_1 - 2x_2 + 4x_3 \geq 8$$

$$x_1, x_2, x_3 \geq 0.$$

d. Minimize $Z = 3x_1 + x_2$
subject to

$$x_1 + x_2 \geq 1$$

$$2x_1 + 3x_2 \geq 2$$

$$x_1, x_2 \geq 0.$$

e. Minimize $Z = 3x_1 + 5x_2 + 4x_3$
subject to

$$-2x_1 - x_2 + 5x_3 \geq 2$$

$$3x_1 + 2x_2 + 4x_3 \geq 16$$

$$x_1, x_2, x_3 \geq 0.$$

Further reading

A. Ravindran, K. M. (2006). *Engineering optimization: Methods*. New York: Wiley.
Bhavikatti, S. S. (2010). *Fundamentals of optimum design in engineering*. New Delhi: New Age International (P) Limited Publishers.
Gass, S. I. (1985). *Linear programming: Methods and applications*. New York: McGraw-Hill.
Grewal, B. S. (2006). *Higher engineering mathematics*. New Delhi: Khanna Publishers.
Rao, S. S. (1995). *Optimization theory and applications*. New Delhi: New Age International (P) Limited Publishers.
Sharma, J. K. (2006). *Operations research theory & applications*. New Delhi: Macmillan India Ltd.
Swarup, K., Gupta, P. K., & Mohan, M. (2010). *Operations research*. New Delhi: Sulthanchand.
Taha, H. A. (1992). *Operations research: An introduction*. New York: Macmillan.
Winston, W. L. (1991). *Operations research: Applications and algorithms*. Boston: PWS-Kent.

CHAPTER THREE

Single-variable nonlinear optimization

This chapter includes a single variable and unconstrained functions with various techniques to investigate the optimum value of single-variable optimization problems. The presence of only one variable in the objective function of single-variable optimization problems makes the optimization procedure easier to understand.

This chapter mainly focuses on algorithms used to investigate the minimization problem. The minimization problem is defined as follows:

$$\text{Minimize } f(x) \tag{3.1}$$

where $f(x)$ is the objective function and x is a real variable. The very cause of an optimization algorithm is to obtain a solution x, for which the function $f(x)$ is minimum. There are two basic approaches, viz., direct search and gradient-based methods that are followed in this chapter. Direct search methods use only objective function values to locate the minimum point, whereas gradient-based methods use derivatives (generally the first and/or second order) of the objective function to locate the minimum point. Furthermore, the derivatives in gradient-based methods are computed numerically, and accordingly, optimization algorithms are discussed. Even though the optimization methods described here are for minimization problems, they can also be used to solve a maximization problem by adopting any of the following two procedures. In the first procedure, an equivalent dual problem $(-f(x))$ is formulated and minimized. In this case, the algorithm described in this chapter can be directly used to solve a maximization problem. In the second procedure, if then else clauses used in the algorithms have to be modified to directly solve maximization problems. The first procedure is simpler to use and is therefore popular, whereas the second procedure is more elegant but requires a proper understanding of the working of the optimization algorithms. Before insight discussion, first, the necessary and sufficient conditions for optimality are presented. Then, two bracketing algorithms, direct and gradient-based search methods, are included. There are three types of optimal points, which are defined as follows:

1. *Local optimal point*: A point or solution x^* is said to be a locally optimal point if there exists no point in the neighborhood of x^*, which is

© 2020 Elsevier Inc.
All rights reserved.

better (function value is either maximum or minimum) than x^*. For the minimization problem, a point x^* is a locally minimal point if no point in the neighborhood has a function value smaller than $f(x^*)$.

2. *Global optimal point*: A point or solution x^* is said to be a global optimal point if there exists no point in the entire search space, which is better (function value is either maximum or minimum) than the point x^*. For minimization problem, a point x^* is a global minimal point if no point in the entire search space has a function value smaller than $f(x^*)$.
3. *Inflection point*: A point x^* is said to be an inflection point if the function value increases locally as x^* increases and decreases locally as x^* reduces or if the function value decreases locally as x^* increases and increases locally as x^* decreases.

Certain characteristics of the underlying objective function can be exploited to check whether a point is either a local minimum or a global minimum, or an inflection point. Assuming that the first- and the second-order derivatives of the objective function $f(x)$ exist, the chosen search space, we may expand the function in Taylor's series at any point $\tilde{x}$ and satisfy the condition that any other point in the neighborhood has a larger function value. It can then be shown that conditions for a point $\tilde{x}$ to be a minimum point is that $f'(\tilde{x}) = 0$ and $f''(\tilde{x}) > 0$, where f' and f'' represent the first- and second-order derivatives of the function. The first condition alone suggests that the point is a minimum, a maximum, or an inflection point, and both conditions together suggest that the point is a maximum. In general, the sufficient conditions of optimality are given as follows.

Suppose at point x^*, the first derivative is zero and the first nonzero higher-order derivative is denoted by n, then (1) if n is odd, x^* is an inflection point. (2) If n is even, x^* is a local optimum.

1. If the derivative is positive, x^* is a local minimum.
2. If the derivative is negative, x^* is a local maximum.

3.1 Classical method for single-variable optimization

A function of one variable $f(x)$ is said to have a relative or local minimum at $x = x^*$ if $f(x^*) \leq f(x^* + \delta)$ for all sufficiently small positive and negative values of δ. Similarly, a point x^* is called a relative or local maximum if $f(x^*) \geq f(x^* + \delta)$ for all values of δ sufficiently close to zero. A function $f(x)$ is said to have a global or absolute minimum at x^* if

$f(x^*) \le f(x)$ for all x and not just for all x close to x^*, in the domain over which $f(x)$ is defined. Similarly, a point x^* will be a global maximum of $f(x)$ if $f(x^*) \ge f(x)$ for all x in the domain.

A single-variable optimization problem is one in which the value of $x = x^*$ is to be found in the interval $[a, b]$, such that x^* minimizes $f(x)$. There are two theorems that give the necessary and sufficient conditions for the relative minimum of a single-variable function.

Theorem 3.1. Necessary condition

If a function $f(x)$ is defined in the interval $a \le x \le b$ and has a relative minimum at $x = x^*$, where $a < x^* < b$, and the derivative $df(x)/dx = f'(x)$ exists as a finite number at $x = x^*$, then $f'(x^*) = 0$.

Proof:. The definition of derivative provides

$$f'(x^*) = \lim_{\delta \to 0} \frac{f(x^* + \delta) - f(x^*)}{\delta} \tag{3.2}$$

which exists as a definite number. But, the aim is to prove that it is zero.

Since x^* is a relative minimum, we have $f(x^*) \le f(x^* + \delta)$ for all values of $\delta > 0$ (sufficiently close to zero). Hence,

$$\frac{f(x^* + \delta) - f(x^*)}{\delta} \ge 0, \text{ if } \delta > 0;$$

$$\frac{f(x^* + \delta) - f(x^*)}{\delta} \le 0, \text{ if } \delta < 0.$$

Eq. (3.2) gives the limit as δ tends to zero through positive values as follows:

$$f'(x^*) \ge 0 \tag{3.3}$$

However, it gives the limit as δ tends to zero through negative values as follows:

$$f'(x^*) \le 0 \tag{3.4}$$

The only way to satisfy both Eqs. (3.3) and (3.4) is to have

$$f'(x^*) = 0 \tag{3.5}$$

This proves the theorem.

From Theorem 3.1, we conclude the following statements.

1. This theorem can be proved even if x^* is a relative maximum.
2. The theorem does not say what happens if a minimum or maximum occurs at a point x^* where the derivative fails to exist.

3. The theorem does not say what happens if a minimum or maximum occurs at an endpoint of the interval. In this case,

$$\lim_{\delta \to 0} \frac{f(x^* + \delta) - f(x^*)}{\delta} \tag{3.6}$$

exists for positive values of δ only or negative values of δ only, and hence, the derivative is not defined at the endpoints.

4. It does not say that the function necessarily will have a minimum or maximum at every point where the derivative is zero. For example, the derivative $f'(x) = 0$ at $x = 0$ for the function $f(x) = x^3$. However, this point is neither a minimum nor a maximum. In general, a point x^* at which $f'(x^*) = 0$ is called a stationary point.

If the function $f(x)$ possesses continuous derivatives of every order that come in question, in the neighborhood of $x = x^*$, the following theorem provides sufficient conditions for the minimum or maximum value of the function.

Theorem 3.2. Sufficient condition.

Let $f'(x^*) = f''(x^*) = \cdots = f^{(n-1)}(x^*) = 0$, but $f^{(n)}(x^*) \neq 0$. Then, $f(x^*)$ is (1) minimum value of $f(x)$ if $f^{(n)}(x^*) > 0$ and n is even; (2) maximum value of $f(x)$ if $f^{(n)}(x^*) < 0$ and n is even; and (3) neither a maximum nor a minimum if n is odd.

Proof:. Applying Taylor's theorem with remainder after n terms, we have

$$\begin{aligned} f(x^* + \delta) = & f(x^*) + \delta f'(x^*) + \frac{\delta^2}{2!} f''(x^*) + \cdots + \frac{\delta^{n-1}}{(n-1)!} f^{(n-1)}(x^*) \\ & + \frac{\delta^n}{n!} f^{(n)}(x^* + \theta h) \quad \text{for } 0 < \theta < 1 \end{aligned} \tag{3.7}$$

Since, $f'(x^*) = f''(x^*) = \cdots = f^{(n-1)}(x^*) = 0$, Eq. (3.7) becomes $f(x^* + \delta) - f(x^*) = (\delta^n / n!) f^{(n)}(x^* + \theta\delta)$.

As $f^{(n)}(x^*) \neq 0$, there exists an interval around x^* for every point x of which the nth derivative $f^{(n)}(x)$ has the same sign, namely, that of $f^{(n)}(x^*)$. Thus for every point $x^* + \delta$ of this interval, $f^{(n)}(x^* + \theta\delta)$ has the sign of $f^{(n)}(x^*)$. When n is even, $\delta^n / n!$ is positive irrespective of whether δ is positive or negative, and hence, $f(x^* + \delta) - f(x^*)$ will have the same sign as that of $f^{(n)}(x^*)$. Thus, $f(x^*)$ will be a relative minimum if $f^{(n)}(x^*)$ is positive and a relative maximum if $f^{(n)}(x^*)$ is negative. When n is odd, $\delta^n/n!$ changes sign with the change in the sign of δ, and hence, the point x^* is neither a maximum nor a minimum. In this case, the point x^* is called a point of inflection.

3.2 Exhaustive search method

One of the very basic ideas to obtain the optimal (minimum and maximum) value is to bound the optimal by lower and upper values. This idea can be illustrated in two phases. First, a crude technique is used to find a lower and an upper bound of the minimum. Then, a sophisticated approach is used to search within these limits and find the optimal solution with the desired accuracy. Because of this, the exhaustive search method is discussed here to bracket the minimum point.

We begin with the exhaustive search method, simply because this method is the simplest of all other methods. In this exhaustive search method, the optimum of a function is bracketed by calculating the function values at several equally spaced points. Usually, the search begins from a lower bound on the variable, and three consecutive function values are compared at a time based on the assumption of unimodality (the presence of only one peak or valley) of the function. Based on the outcome of the comparison, the search is either terminated or continued by replacing one of the three points by a new point. The search continues until the minimum is bracketed. The following is the algorithm for the same.

Algorithm 3.1.

Step 1: Assume $x_1 = a$, and step size $\Delta = (b - a)/n$ (n is the number of intermediate points). Two search points are $x_2 = x_1 + \Delta$, and $x_3 = x_2 + \Delta = x_1 + 2\Delta$.

Step 2: If $f(x_1) \geq f(x_2) \leq f(x_3)$, the minimum point lies in (x_1, x_3). Else, $x_1 = x_2$, $x_2 = x_3$, $x_3 = x_2 + \Delta$. Go to Step 3.

Step 3: If $x_3 \leq b$, then go to Step 2. Else, no minimum exists in (a, b) or a boundary point (a or b) is the minimum point.

In this method, it is noticed that (1) the final interval obtained by using this algorithm is $(2(b - a))/n$, (2) the average number of function evaluations to get the desired accuracy is $\left(\left(n/2\right) + 2\right)$. For better visualization and construction of programming codes, flowchart of the algorithm is presented in Fig. 3.1.

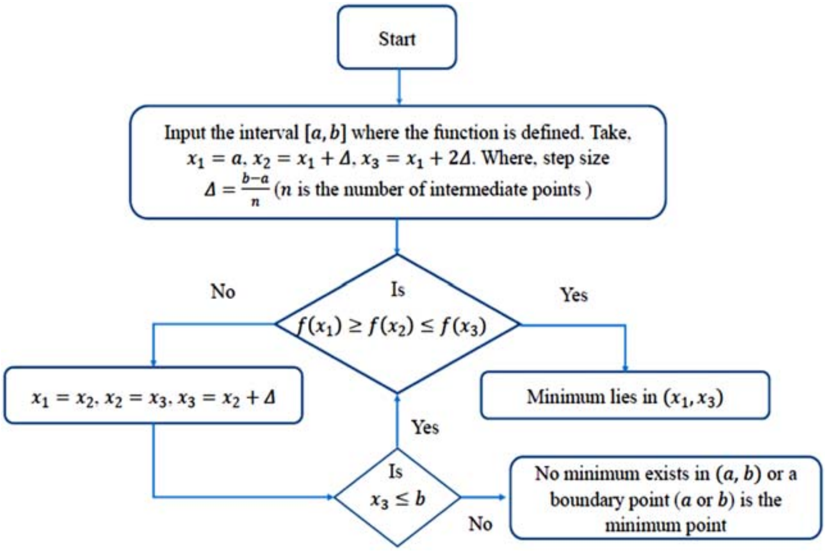

Start
Input the interval $[a,b]$ where the function is defined. Take, $x_1 = a$, $x_2 = x_1 + \Delta$, $x_3 = x_1 + 2\Delta$. Where, step size $\Delta = \frac{b-a}{n}$ (n is the number of intermediate points)
Is $f(x_1) \geq f(x_2) \leq f(x_3)$
No
Yes
$x_1 = x_2$, $x_2 = x_3$, $x_3 = x_2 + \Delta$
Minimum lies in (x_1, x_3)
Yes
Is $x_3 \leq b$
No
No minimum exists in (a, b) or a boundary point (a or b) is the minimum point

Figure 3.1 Flow diagram of the exhaustive search method.

To illustrate the working of Algorithm 3.3, we consider the following minimization problem.

Example 3.1. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represents height at t, defined in the interval $(0, 5)$. Find the value of t at which the height is minimum.

Solution: To start the procedure and get the bracket where minimum exist, we need to calculate Δ value, which is $(b-a)/n$. Here, $a = 0$ and $b = 5$; assume $n = 10$. Then, we get $\Delta = 0.5$. Using these values, we get, $t_1 = 0$, $t_2 = t_1 + 0.5 = 0.5$, and $t_3 = t_1 + 2\Delta = 1$. Step 2 says compute the function values at different points. Hence, $h(0) = \infty$, $h(0.5) = 108.25$, and $h(1) = 55$. To compare these values, we observe that $h(t_1) > h(t_2) > h(t_3)$. So, here we have to check if the optimal condition $h(t_1) \geq h(t_2) \leq h(t_3)$ is satisfied, if not then $t_1 = t_2$, $t_2 = t_3$, and $t_3 = t_2 + \Delta$. The earlier calculated function values show that the optimality condition does not satisfied and the minimum do not lie in the interval $(0, 1)$. Hence, $t_1 = 0.5$, $t_2 = 1$, and $t_3 = 1.5$. Then, proceed to step 3. From step 3, it is seen that $t_3 < 5$. Therefore we move to step 2 and continue the process till the optimal condition is reached. In Table 3.1, all possible iterations are recorded, and the bracket is shown where minimum lies.

Table 3.1 The solution set for Example 3.1.

No. of iterations	t_1	t_2	t_3	$h(t_1)$	$h(t_2)$	$h(t_3)$	$h(t_1) \geq h(t_2) \leq h(t_3)$
1	0	0.5	1	∞	108.25	55	No
2	0.5	1	1.5	108.25	55	38.25	No
3	1	1.5	2	55	38.25	32	No
4	1.5	2	2.5	38.25	32	27.85	No
5	2	2.5	3	32	27.85	27	No
6	*2.5*	*3*	*3.5*	*27.85*	*27*	*27.68*	*Yes*

From Table 3.1, it is obtained that at iteration 6, the optimal condition is attained. Hence, the minimum lies in (t_1, t_3), which is $(2.5, 3.5)$ and the object will possess minimum height at $t \in (2.5, 3.5)$. The following MATLAB code of the exhaustive search method is given for the same.

MATLAB Code

```
clear all
a=input('the left value of the domain where the function is defined:');
b=input('the right value of the domain where the function is defined:');
n=input('the number of intermediate points:');
fprintf('   \n');
dt=(b-a)/n;
t1=0;
t2=t1+dt;
t3=t2+dt;
while t3<=b

h1=t1^2+(54/t1);
h2=t2^2+(54/t2);
h3=t3^2+(54/t3);
if (h1>= h2 && h2<=h3)
     fprintf('The minimum lie between %d  and  %d.\n',t1,t3);
     fprintf('   \n');
     fprintf('The optimal solution is %d, which is at x =  %d.\n',h2,t2);
     break
else
    fprintf('The minimum does not lie between %d  and  %d.\n',t1,t3);
    fprintf('   \n');
end
t1=t2;
t2=t3;
t3=t2+dt;
end
```

3.3 Bounding phase method

The bounding phase method is also used to bracket the minimum of a single-variable function. This method guarantees to bracket the minimum of a unimodal function. The procedure of searching a minimum starts with an initial guess and thereby finds a search direction based on two more function evaluations in the vicinity of the initial guess. Then, a search strategy is adopted to reach the optimum. The detailed approach to obtain a minimum using the bounding phase method is outlined in the following algorithm.

Algorithm 3.2.

Step 1: Choose an initial guess $x^{(0)}$ and an increment Δ. Set $k = 0$.
Step 2: If $f\left(x^{(0)} - |\Delta|\right) \geq f\left(x^{(0)}\right) \geq f\left(x^{(0)} + |\Delta|\right)$, then Δ is positive. Else, $f\left(x^{(0)} - |\Delta|\right) \leq f\left(x^{(0)}\right) \leq f\left(x^{(0)} + |\Delta|\right)$, then Δ is negative. Else go to Step 1.
Step 3: Set $x^{(k+1)} = x^{(k)} + 2^k\Delta$.
Step 4: If $f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$, set $k = k + 1$, and go to Step 3. Else, the minimum lies in the interval $\left(x^{(k-1)}, x^{(k+1)}\right)$ and terminate.

An example problem is discussed to demonstrate this method.

Example 3.2. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represents the height at t, defined in the interval $(0, 5)$. Using the bounding phase method, find the value of t at which the height is minimum.

Solution: To start the bounding phase algorithm, we need to choose initial guess value $\left(t^{(0)}\right)$ and step length (Δ). Then, we have to calculate three function values, that is, $h\left(t^{(0)} - |\Delta|\right) = h(0.6 - 0.5) = h(0.1) = 540.01, h\left(t^{(0)}\right)$ $h\left(t^{(0)}\right) = h(0.6) = 90.36$, and $h\left(t^{(0)} + |\Delta|\right) = h(0.6 + 0.5) = h(1.1) = 50.301$. Here, it observes that $h(0.1) > h(0.6) > h(1.1)$. Hence, $\Delta = 0.5$ is positive. Then, in step 3, we have to compute the next point using the initial guess value. So, $t^{(1)} = t^{(0)} + 2^0\Delta = 0.6 + 0.5 = 1.1$. Now, $h\left(t^{(1)}\right) = 50.301 < h\left(t^{(0)}\right) = 90.36$. Thus, we set $k = 1$ and go to step 3. This completes the first iteration. Similarly, other iterations are performed till satisfying the optimal condition, that is, $h\left(t^{(k+1)}\right) > h\left(t^{(k)}\right)$. All the desired iterations are presented in Table 3.2, and the solution bound is obtained.

From Table 3.2, it is seen that at the fourth iteration the optimality condition is satisfied, and hence, the minimum lies in $\left(t^{(k-1)}, t^{(k+1)}\right) = (2.1, 8.1)$.

Table 3.2 The solution set for Example 3.2.

No. of iterations	$t^{(k)}$	$t^{(k+1)}$	$h\left(t^{(k)}\right)$	$h\left(t^{(k+1)}\right)$	$h\left(t^{(k+1)}\right) > h\left(t^{(k)}\right)$
1	0.6	1.1	90.36	50.301	No
2	1.1	2.1	50.301	30.124	No
3	**2.1**	4.1	30.124	29.981	No
4	4.1	**8.1**	29.981	72.277	Yes

Bold values decide the solution bound.

Hence, the minimum lies in $(2.1, 8.1)$, and the object will possess minimum height at $t \in (2.1, 8.1)$. In the following, MATLAB code of bounding phase method is given for the same.

MATLAB Code

```
clear all
a=input('the left value of the domain where the function is defined: ');
b=input('the right value of the domain where the function is defined: ');
t0 = input('Enter the initial guess: ');
    dt = input ('Enter the increment: ');

    k = 0;
    t1 = t0 - dt;
    t2 = t0;
    t3 = t0 + dt;
    h1 = t1^2+(54/t1);
    h2 = t2^2+(54/t2);
    h3 = t3^2+(54/t3);
    if h1 >= h2 && h2 >= h3
        increment_t = abs(dt);
    else
        increment_t = -abs(dt);
    end
    while (t3 <= b)
        h_old = t2^2+(54/t2);
        h_new = t3^2+(54/t3);
        if h_new > h_old
            break
        else
            k = k+1;
            t1=t2;
            t2 = t3;
            t3 = t2 + (2^k)*increment_t;
            h2 = t2^2+(54/t2);
            h3 = t3^2+(54/t3);
        end
    end
    fprintf('  \n')
    fprintf('The solution lies between %f & %f \n',t1,t3);
```

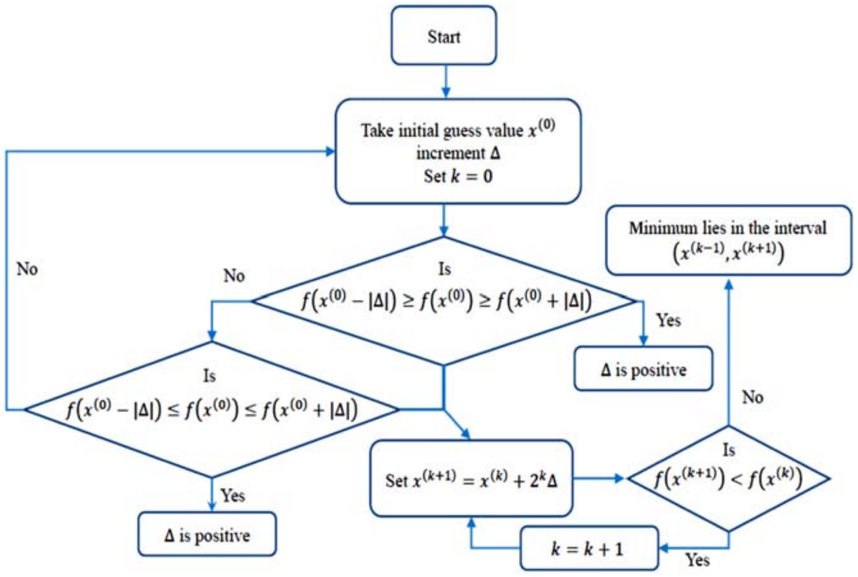

Start
Take initial guess value $x^{(0)}$ increment Δ Set $k = 0$
Is $f(x^{(0)} - |\Delta|) \geq f(x^{(0)}) \geq f(x^{(0)} + |\Delta|)$
Yes
Δ is positive
No
Is $f(x^{(0)} - |\Delta|) \leq f(x^{(0)}) \leq f(x^{(0)} + |\Delta|)$
Yes
Δ is positive
No
Set $x^{(k+1)} = x^{(k)} + 2^k\Delta$
Is $f(x^{(k+1)}) < f(x^{(k)})$
Yes
$k = k + 1$
No
Minimum lies in the interval $(x^{(k-1)}, x^{(k+1)})$

Figure 3.2 Flow diagram of the bounding phase method.

If the chosen Δ is large, the bracketing accuracy of the minimum point is poor, but the bracketing of the minimum is faster. On the other hand, if the chosen Δ is small, the bracketing accuracy is better, but more function evaluations may be necessary to bracket the minimum. This method of bracketing the optimum is usually faster than the exhaustive search method. For easy illustration and getting an insight understanding of constructing programming code, the flowchart of this method is mentioned in Fig. 3.2.

3.4 Interval halving method

In Section 3.3, we have bracketed the minimum point. But the next thing is the accuracy of the solution. In this section, one of the methods, viz., interval halving method is considered to improve the accuracy of the solution. The interval halving method works on the principle of the region elimination rule. This method requires comparatively smaller function evaluations. The procedure of this method depends on the function evaluation at two points, and assuming that the function is unimodal in the chosen search space, it can be concluded that the desired minimum cannot lie in some portion of the search space. Considering the first fundamental rule for region elimination, the algorithm of this method will be discussed. The fundamental rule for region elimination methods is as follows.

Let us consider two points x_1 and x_2 which lie in the interval (a, b) and satisfy $x_1 < x_2$. For unimodal functions for minimization, we can conclude the following.

1. If $f(x_1) > f(x_2)$, then the minimum does not lie in (a, x_1).
2. If $f(x_1) < f(x_2)$, then the minimum does not lie in (x_2, b).
3. If $(x_1) = f(x_2)$, then the minimum does not lie in (a, x_1) and (x_2, b).

Consider a unimodal function, viz., $f(x) = x^2$ defined in $[a, b]$. If the function value at $x_1 \in [a, b]$ is larger than that at $x_2 \in [a, b]$, the minimum point x^* cannot lie on the left side of x_1. Thus we can eliminate the region (a, x_1) from further consideration. Therefore we reduce our interval of interest from (a, b) to (x_1, b). Similarly, the second possibility, $f(x_1) < f(x_2)$, can be explained. If the third situation occurs, that is, when $f(x_1) = f(x_2)$, we can conclude that regions (a, x_1) and (x_2, b) can be eliminated with the assumption that there exists only one local minimum in the search space (a, b).

In this method, function values at three different points are considered. Three points divide the search space into four regions. The fundamental region elimination rule is used to eliminate a portion of the search space based on the function values at three chosen points. Three points are chosen in the interval (a, b), which are all equidistant from each other and equidistant from the boundaries by the same amount. Two of the function values are compared at a time, and some region is eliminated. Three cases may occur.

1. If $f(x_1) < f(x_m)$, then the minimum cannot lie beyond x_m. Therefore, we reduce the interval from (a, b) to (a, x_m). The point x_m being the middle of the search space, this elimination reduces the search space to 50% of the original search space.
2. If $f(x_1) > f(x_m)$, the minimum cannot lie in the interval (a, x_1). The point x_1 being at one-fourth point in the search space, this reduction is only 25% of the search space.
3. Compare function values at x_m and x_2 to eliminate further 25% of the search space. This process continues until a small enough interval is found.

Considering the aforementioned cases with the fundamental rule of region elimination, the algorithm of the interval halving method is described here. Since in each iteration of the algorithm, exactly half of the search space is retained, the algorithm is called the interval halving method.

Algorithm 3.3.

Step 1: Consider a lower bound a and an upper bound b. Choose a small number $\epsilon > 0$. Take, $x_m = (a + b)/2$, $L_0 = L = b - a$. Compute $f(x_m)$.
Step 2: Set $x_1 = a + (L/4), x_2 = b - (L/4)$. Compute $f(x_1)$ and $f(x_2)$.
Step 3: If $f(x_1) < f(x_m)$, set $b = x_m$; $x_m = x_1$. Go to Step 5. Else go to Step 4.
Step 4: If $f(x_2) < f(x_m)$, set $a = x_m$; $x_m = x_2$. Go to Step 5. Else set $a = x_1$, $b = x_2$. Go to Step 5.
Step 5: Calculate $L = b - a$. If $|L| < \epsilon$, then stop the process. Else go to Step 2.

Example 3.3. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represents height at t, defined in the interval $(0, 5)$. By using the interval halving method, find the value of t at which the height is minimum.

Solution: To start the interval halving method, we need to know the left bound, the right bound, and the tolerance value. Here, the left bound a is 0 and the right bound b is 1. We have assumed the tolerance value $\epsilon = 0.001$ to stop the procedure and get the desired values.

Iteration 1: Here, we have $a = 0$, $b = 5$. So, $L = b - a = 5$; $t_m = (a + b)/2 = 2.5$; $t_1 = a + (L/4) = 1.25$; and $t_2 = b - (L/4) = 3.75$. Now, we have to compute $h(t_m)$, $h(t_1)$, and $h(t_2)$ and then perform Step 3 and Step 4 of the algorithm. We get $h(t_m) = 27.85$, $h(t_1) = 44.7625$, and $h(t_2) = 28.4625$; $h(t_1) > h(t_m)$ and $h(t_2) > h(t_m)$. By using the region elimination rule, we observe that the region $(a, t_1) = (0, 1.25)$ and $(t_2, b) = (3.75, 5)$ will be eliminated and the minimum lies in $(1.25, 3.75)$. The new L becomes $3.75 - 1.25 = 2.5$; $|L| < \epsilon$ is not satisfied. So, the next iteration is needed. This completes the first iteration.

Iteration 2: Here, we have $a = 1.25$ and $b = 3.75$. So, $L = b - a = 2.5$; $t_m = (a + b)/2 = 2.5$; $t_1 = a + (L/4) = 1.8750$; and $t_2 = b - (L/4) = 3.1250$. Now, we have to compute $h(t_m)$, $h(t_1)$, and $h(t_2)$ and then perform Step 3 and Step 4 of the algorithm. We obtain $h(t_m) = 27.85$, $h(t_1) = 32.3156$, and $h(t_2) = 27.0456$; $h(t_1) > h(t_m)$ and $h(t_2) < h(t_m)$. By using region elimination rule, we observe that the region $(a, t_m) = (1.25, 2.5)$ will be eliminated and the minimum lies in $(2.5, 3.75)$. The new L becomes $3.1250 - 2.5 = 0.625$; $|L| < \epsilon$ is not satisfied. So, the next iteration is needed. This completes the second iteration.

Iteration 3: Here, we have $a = 2.5$, $b = 3.75$. So, $L = b - a = 1.25$; $t_m = (a + b)/2 = 3.1250$; $t_1 = a + (L/4) = 2.8125$; and $t_2 = b - (L/4) = 3.4375$.

Now, we have to compute $h(t_m)$, $h(t_1)$, and $h(t_2)$ and then perform Step 3 and Step 4 of the algorithm. We obtain $h(t_m) = 27.0456$, $h(t_1) = 27.1102$, and $h(t_2) = 27.5255$; $h(t_1) > h(t_m)$ and $h(t_2) > h(t_m)$. By using the region elimination rule, we observe that the region $(a, t_1) = (2.5, 2.8125)$ and $(t_2, b) = (3.4375, 3.75)$ will be eliminated and the minimum lies in $(2.8125, 3.4375)$. The new L becomes $3.4375 - 2.8125 = 0.625$; $|L| < \epsilon$ is not satisfied. So, the next iteration is needed. This completes the third iteration. Similarly, the iterations are performed until the optimality ($|L| < \epsilon$) is satisfied.

In the following, MATLAB code of interval halving method is given for the same.

MATLAB Code

```
clear all
clc
a=input('enter the left bound: ');
b=input('enter the right bound: ');
tol=input('enter the value of Tolerance: ');
L=abs(b-a);
tm=(a+b)/2;
t1=a+L/4;
t2=b-L/4;
ht1=t1^2+54/t1;
ht2=t2^2+54/t2;
htm=tm^2+54/tm;
while L>=tol
L=abs(b-a);
t1=a+L/4;
t2=b-L/4;
ht1=t1^2+54/t1;
ht2=t2^2+54/t2;
if ht2<=htm
        a = tm;
        tm = t2;
        htm = ht2;
    elseif ht1<=htm
        b = tm;
        tm = t1;
        htm = ht1;
    else
        a = t1;
        b = t2;
    end
end
fprintf('The minimum lies between %d and %d ', a, b);
```

At every iteration, two new function evaluations are performed, and the interval reduces to half of that at the previous iterations. Thus, the interval reduces to about $\left(\frac{1}{2}\right)^{n/2} L_0$ after n function evaluations. Hence, the function evaluations required to achieve the desired accuracy ϵ can be computed by solving the following equation:

$$\left(\frac{1}{2}\right)^{n/2}(b-a)=\epsilon \tag{3.8}$$

It is observed that at the end of each iteration, the interval is reduced to half of its original size, and after three iterations, the interval is $\left(\frac{1}{2}\right)^{3} L_0$. Since two function evaluations are required per iteration and half of the region is eliminated at each iteration, the effective region elimination per function evaluation is 25%.

For better visualization and construction of programming codes, a flowchart of the algorithm is presented in Fig. 3.3.

The interval halving method can be improved if the number of function evaluations will be less. Lesser number of function evaluations can decrease the computational time. In the Section 3.5, we will consider the algorithm, which will compute a lesser number of functions than the interval halving method.

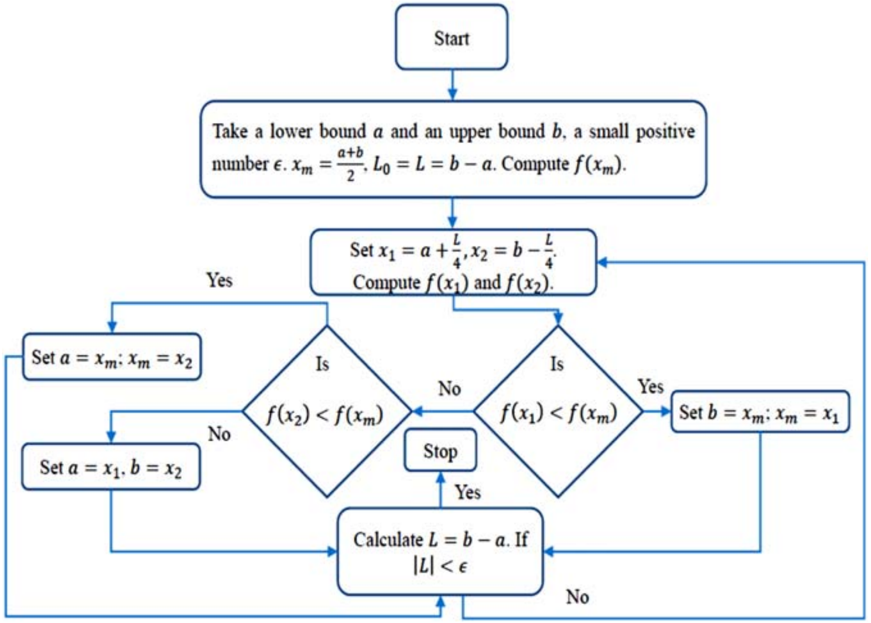
3.5 Fibonacci search method

This is also a search method that can be used to bracket the minimum with better accuracy. In this method, the search interval is reduced according to the Fibonacci numbers. The Fibonacci numbers can be constructed by using the following equation:

$$F_n = F_{n-1} + F_{n-2} \tag{3.9}$$

where $n = 2, 3, 4, \cdots$.

Here, F_{n-2} and F_{n-1} are two consecutive numbers. The first few Fibonacci numbers are $F_0 = 1, F_1 = 1, F_2 = 2, F_3 = 3, F_4 = 5, F_5 = 8, F_6 = 13, F_7 = 21$, and so on. The property of the Fibonacci numbers can be used to create a search algorithm that requires only one function evaluation at each iteration. The principle of the Fibonacci search is that of two points required for the region elimination rule, one is always the previous point and the other point is new. Thus, only one function evaluation is required



Start
Take a lower bound a and an upper bound b, a small positive number ϵ. $x_m = \frac{a+b}{2}$, $L_0 = L = b - a$. Compute $f(x_m)$.
Set $x_1 = a + \frac{L}{4}, x_2 = b - \frac{L}{4}$. Compute $f(x_1)$ and $f(x_2)$.
Yes
Set $a = x_m$; $x_m = x_2$
Is $f(x_2) < f(x_m)$
No
Is $f(x_1) < f(x_m)$
Yes
Set $b = x_m$; $x_m = x_1$
No
Set $a = x_1, b = x_2$
Stop
Yes
Calculate $L = b - a$. If $|L| < \epsilon$
No

Figure 3.3 Flow diagram of the interval halving method.

at each iteration. At iteration k, two intermediate points, each L_k^* away from either end of the search space ($L = b - a$) is chosen. When the region elimination rule eliminates a portion of the search space depending on the function values at these two points, the remaining search space is L_k. By defining $L_k^* = \big((F_{n-k+1})/(F_{n+1})\big)L$ and $L_k = \big((F_{n-k+2})/(F_{n+1})\big)L$, it can be shown that $L_k - L_k^* = L_{k+1}^*$, which means that one of the two points used in iteration k remains as one point in iteration $(k+1)$. If the region (a, x_2) is eliminated in the kth iteration, the point x_1 is at a distance $(L_k - L_k^*)$ or L_{k+1}^* from the point x_2 in the $(k+1)$th iteration. Since the first two Fibonacci numbers are the same, the algorithm usually starts with $k = 2$.

Algorithm 3.4.

Step 1: Choose a lower bound a and an upper bound b. Set $L = b - a$. Assume the desired number of function evaluations to be n. Set $k = 2$.

Step 2: Compute $L_k^* = \big((F_{n-k+1})/F_{n+1}\big)L$. Set $x_1 = a + L_k^*$ and $x_2 = b - L_k^*$.

Step 3: Compute $f(x_1)$ or $f(x_2)$, which was not evaluated earlier. Use fundamental region elimination rule to eliminate a region. Set new a and b.

Step 4: Is $k = n$? If no, set $k = k + 1$ and go to Step 2. Else terminate.

Example 3.4. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represents the height at t, defined in the interval $(0, 5)$. By using the Fibonacci search method, find the value of t at which the height is minimum.

Solution: Assume the value of $a = 0$ and $b = 5$; then, the length $L = b - a = 5$; $n = 3$. In the Fibonacci search method, the length L will be the same throughout all the iterations. Set $k = 2$, $L_2^* = \big((F_{n-k+1})/F_{n+1}\big)$ $L = (F_2/F_4) \times L = (2/5) \times 5 = 2$. Then, $t_1 = a + L_2^* = 0 + 2 = 2$ and $t_2 = b - L_2^* = 5 - 2 = 3$; $h(t_1) = 31$, and $h(t_2) = 27$. Hence, $h(t_1) > h(t_2)$. By using the region elimination rule, the region $(0, 2)$ will be eliminated and the minimum lies in $(2, 5)$. This completes the first iteration.

For the next iteration, the value of $a = 2$ and $b = 5$; length will be $L = 5$. Here, $k = 3$, so $L_3^* = (F_1/F_4) \times L = (1/5) \times 5 = 1$; $t_1 = 3$ and $t_2 = 4$; $h(t_1) = 27$ and $h(t_2) = 29.5$. Hence, $h(t_2) > h(t_1)$. By using the region elimination rule, the region $(4, 5)$ will be eliminated and the minimum lies in $(2, 4)$. Here, $k = 3$, which is the same as the value n is considered. So, the process will be terminated and the minimum lies in the bracket $(2, 4)$. The desired minimum value can be taken as the midpoint of $(2, 4)$, which is 3.

In the following, a general MATLAB code is given for the same problem.

MATLAB Code

```
function f_t = fibo(t)
if t == 0
    f_t = 1;
elseif t == 1
    f_t = 1;
else
    a = 1;
    b = 1;
    for i = 2:t
        f_t = a + b;
        a = b;
        b = f_t;
    end
end
end


clc
clear all
a = input ('Enter the left boundary point: ');
b = input ('Enter the right boundary point: ');
n = input ('Enter the desired number of function evaluation (greater than
2): ');
k = 2;
L = b - a;
h1 = a^2+54/a;
h2 = b^2+54/b;
while k <= n

    t1 = n-k+1;
    ht1 = fibo(n-k+1);
    ht2 = fibo(n+1);
    Lk = (ht1/ht2)* L;
    t1 = a + Lk;
    t2 = b - Lk;
        h1 = t1^2+54/t1;
        h2 = t2^2+54/t2;
    if h1 > h2
        a = t1;
    else
        b = t2;
    end
fprintf(' \n');
fprintf ('No. of iteration is %d \n',k-1);
fprintf('a= %f \n',a);
fprintf('b= %f \n',b);
fprintf('Lk= %f \n',Lk);
    k = k + 1;
end
```

In this algorithm, the interval reduces to $(2/F_{n+1})L$ after n function evaluations. Thus, for the desired accuracy ϵ, the number of required function evaluations n can be calculated using the following equation:

$$\frac{2}{F_{n+1}}(b-a)=\epsilon. \tag{3.10}$$

Hence, it is clear from the algorithm that only one function evaluation is required at each iteration. At iteration k, a proportion of $\frac{F_{n-k}}{F_{n-k+2}}$ of the search space at the previous iteration is eliminated. For large values of n, this quantity is close to 38.2%, which is better than that in the interval halving method. The difficulty with this algorithm is that the Fibonacci numbers must be calculated in each iteration. For easy illustration and understanding of constructing programming code, a flowchart of the Fibonacci search method is shown in Fig. 3.4.

3.6 Golden section search method

From the Section 3.5, it can be noticed that the Fibonacci search method has the following disadvantages:

1. The Fibonacci numbers have to be calculated and stored.
2. At every iteration, the portion of the eliminated region is not the same.

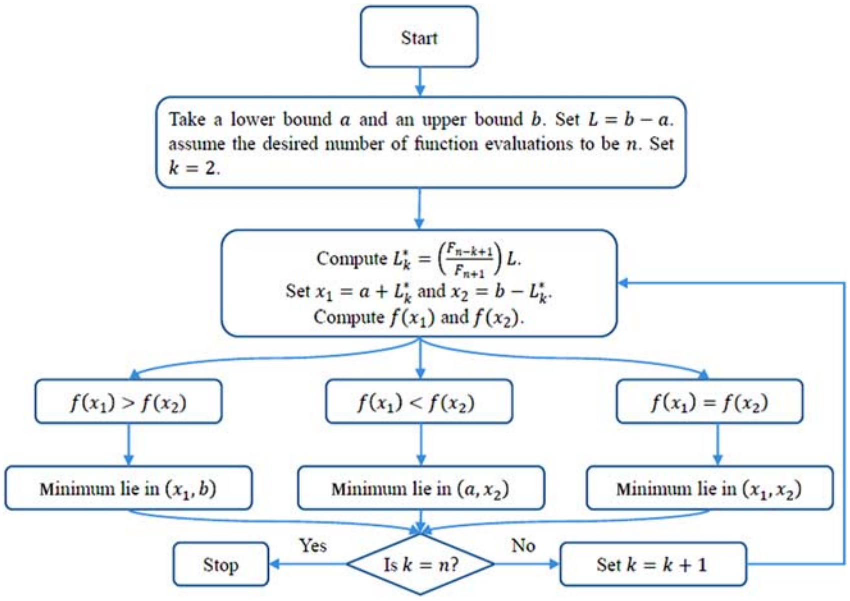

Start
Take a lower bound a and an upper bound b. Set $L=b-a$. assume the desired number of function evaluations to be n. Set $k=2$.
Compute $L_k^*=\left(\frac{F_{n-k+1}}{F_{n+1}}\right)L$. Set $x_1=a+L_k^*$ and $x_2=b-L_k^*$. Compute $f(x_1)$ and $f(x_2)$.
$f(x_1)>f(x_2)$
$f(x_1)<f(x_2)$
$f(x_1)=f(x_2)$
Minimum lie in (x_1,b)
Minimum lie in (a,x_2)
Minimum lie in (x_1,x_2)
Stop
Yes
Is $k=n$?
No
Set $k=k+1$

Figure 3.4 Flow diagram of the Fibonacci search method.

The aforementioned two disadvantages can be overcome, and a new method called the golden section search method can be developed in the following way.

1. Consider the number n is too large, that is, $n \rightarrow \infty$.
2. The length of the search interval L will vary in each iteration.
3. Convert the original interval (a, b) to $(0, 1)$.

Initially, in this algorithm, the search space (a, b) is linearly mapped to a unit interval search space $(0, 1)$. Then, two points t from either end of the search space are chosen, so that at every iteration, the eliminated region is $(1-t)$ to that in the previous iteration. This can be done by equating $(1-t)$ with t^2. Solving the equation $t^2 = 1 - t$, we obtain the golden number $t = 0.618$. In each iteration, one of the two points x_1 and x_2 is always a point considered in the previous iteration.

Algorithm 3.5.

Step 1: Choose a lower bound a and an upper bound b. Also, choose a small number ϵ. Normalize the variable x by using the equation $w = (x-a)/(b-a)$. Thus, $a_w = 0, b_w = 1$ and $L_w = 1$. Set $k = 1$.

Step 2: Set $w_1 = a_w + (0.618)L_w$ and $w_2 = b_w - (0.618)L_w$. Compute $f(w_1)$ or $f(w_2)$ depending on whichever of the two was not evaluated earlier. Use the fundamental region elimination rule to eliminate a region. Set new a_w and b_w.

Step 3: Is $|L_w| < \epsilon$ small? If no, set $k = k + 1$, go to Step 2. Else terminate.

Example 3.5. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represents the height at t, defined in the interval $(0, 5)$. By using the golden section search method, find the value of t at which the height is minimum.

Solution: Take $a = 0$ and $b = 5$; assume $\epsilon = 0.001$. Applying Step 1, the transformed equation is $w = \frac{t}{5} \Rightarrow t = 5w$. Thus $a_w = 0$, $b_w = 1$, and $L_w = 1$. Step 2 says that we have to calculate w_1 and w_2 and then compute $h(w_1)$ and $h(w_2)$. Here, $h(w) = (54/5w) + 25w^2$. Now, $w_1 = a + 0.618 \times L_w = 0.618$ and $w_2 = b - 0.618 \times L_w = 0.382$. The corresponding function values are $f(w_1) = 27.02$ and $f(w_2) = 31.92$. Hence, $f(w_1) < f(w_2)$, and by using region elimination, we get the minimum lie in $(0.382, 1)$. Here, $|L_w| = 0.618 > \epsilon$, which does not satisfy the optimality condition. This completes the first iteration. The obtained values in terms of w will be converted to t by using the transformed equation $t = 5w$, and the desired value of t is investigated. Similarly, iterations will be performed until the optimality condition is satisfied. The obtained results for the first three iterations are summarized in Table 3.3.

In the following, a general MATLAB code is given for the same problem.

MATLAB Code

```
clc
clear
str = input('Give an equation in x: ','s');
f = inline(str,'x');
a = input ('Enter lower boundary point: ');
b = input ('Enter upper boundary point: ');
tol = input ('Enter the value of Tolerance: ');
k = 2;
a_new = a;
b_new = b;
Lw = b - a;
aw = (a_new - a)/(b  - a);
bw = (b_new - a)/(b  - a);
Lw = bw - aw;
w1 = aw + .618 * Lw;
w2 = bw - .618 * Lw;
y1 = feval(f,(w1));
y2 = feval(f,(w2));
while Lw > tol
    aw = (a_new - a)/(b  - a);
    bw = (b_new - a)/(b  - a);
    Lw = bw - aw;
    w1 = aw + .618 * Lw;
    w2 = bw - .618 * Lw;
    if mod (Lw,2) == 0
        y1 = feval(f,(w1));
    else
        y2 = feval(f,(w2));
    end
    if y1 > y2
        a_new = w1;

    else
        b_new = w2;
    end
    k = k + 1;
end
```

Table 3.3 The solution set for Example 3.5.

No. of iteration	w_1	w_2	$f(w_1)$	$f(w_2)$	Eliminated region	New interval
1	0.618	0.382	27.02	31.92	$(0, 0.382)$	$(0.382, 1)$
2	0.764	0.618	28.73	27.02	$(0.764, 1)$	$(0.382, 0.764)$
3	0.618	0.528	27.02	27.43	$(0.382, 0.528)$	$(0.528, 0.764)$

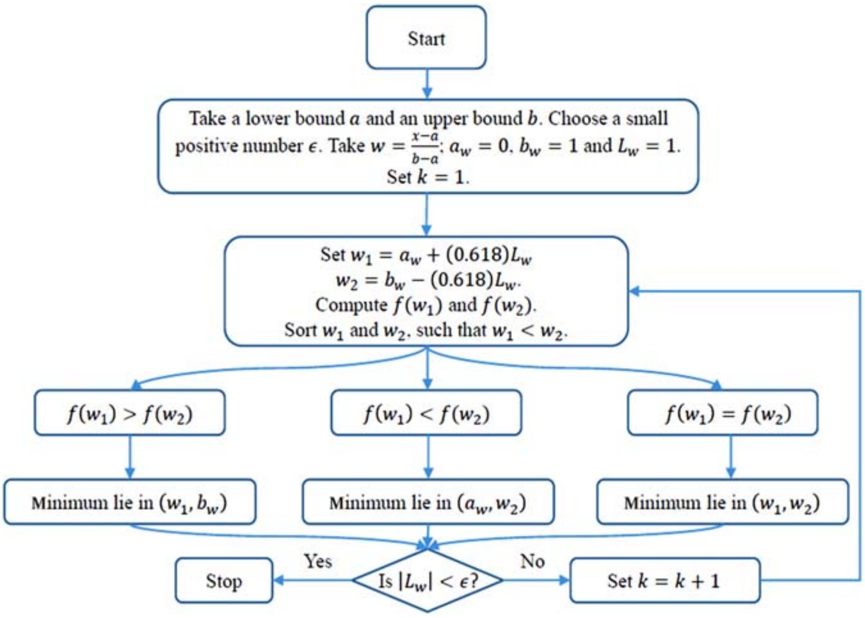

Start
Take a lower bound a and an upper bound b. Choose a small positive number ϵ. Take $w = \frac{x-a}{b-a}$; $a_w = 0$, $b_w = 1$ and $L_w = 1$.
Set $k = 1$.
Set $w_1 = a_w + (0.618)L_w$
$w_2 = b_w - (0.618)L_w$.
Compute $f(w_1)$ and $f(w_2)$.
Sort w_1 and w_2, such that $w_1 < w_2$.
$f(w_1) > f(w_2)$
$f(w_1) < f(w_2)$
$f(w_1) = f(w_2)$
Minimum lie in (w_1, b_w)
Minimum lie in (a_w, w_2)
Minimum lie in (w_1, w_2)
Stop
Yes
Is $|L_w| < \epsilon$?
No
Set $k = k + 1$

Figure 3.5 Flow diagram of the golden section search method.

In this algorithm, the interval reduces to $(0.618)^{n-1}$ after n function evaluations. Thus, the number of function evaluations n required to get the desired accuracy ϵ is calculated by solving the following equation

$$(0.618)^{n-1}(b-a) = \epsilon. \tag{3.11}$$

Like the Fibonacci method, only one function evaluation is required at each iteration, and the effective region elimination per function evaluation is exactly 38.2%, which is higher than that in the interval halving method. This quantity is the same as that in the Fibonacci search for large n. In fact, for a large n, the Fibonacci search is equivalent to the golden section search. For the insight understanding of building programming code, a flowchart of the golden section search method is shown in Fig. 3.5.

3.7 Bisection method

This method is similar to the region elimination method, but here derivatives are used to make the decision about the region to be eliminated. The algorithm of this method requires to compute the first derivative. Both the function value and the sign of the first derivative at two points are used to eliminate a certain portion of the search space. The assumption is that the function is unimodal. By using the derivative information, the minimum is said to be bracketed in the interval (a, b) if the condition $f'(a) \cdot f'(b) < 0$ (provided $f'(a) < 0$ and $f'(b) > 0$) is satisfied. Like other region elimination methods, this algorithm also requires two initial boundary points to bracket the minimum. In the bisection method, derivatives at two boundary points and the middle point are calculated and compared. Of the three points, two consecutive points with derivatives having opposite signs are chosen for the next iteration.

Algorithm 3.6.

Step 1: Choose two points a and b, such that $f'(a) < 0$ and $f'(b) > 0$. Also, choose a small number ϵ. Set $x_1 = a$ and $x_2 = b$.
Step 2: Calculate $z = (x_1 + x_2)/2$ and evaluate $f'(z)$.
Step 3: If $|f'(z)| \leq \epsilon$, terminate. Else if $f'(z) < 0$, set $x_1 = z$ and go to Step 2; Else if $f'(z) > 0$, set $x_2 = z$ and go to Step 2.

Example 3.6. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represents the height at t, defined in the interval $(0, 5)$. By using the bisection method, find the value of t at which the height is minimum.

Solution: Choose two points $a = 2$ and $b = 5$, such that $h'(a) = -9.501$ and $h'(b) = 7.841$, which satisfy the condition $h'(a) \cdot h'(b) < 0$. Take $t_1 = a$ and $t_2 = b$ and calculate $t_3 = (t_1 + t_2)/2 = 3.5$ by using difference formula (Eq. 3.12) $h'(t_3) = 2.591$. Since, $h'(t_3) > 0$, the right half of the search space will be eliminated. Thus, we get $t_1 = 2$ and $t_2 = t_3 = 3.5$.

$$f'\left(x^{(t)}\right) = \frac{f\left(x^{(t)} + \Delta x^{(t)}\right) - f\left(x^{(t)} - \Delta x^{(t)}\right)}{2\Delta x^{(t)}} \tag{3.12}$$

This completes the first iteration. Similarly, for the second iteration, $t_3 = (2 + 3.5)/2 = 2.750$ and $h'(t) = -1.641$. Since $h'(t_3) < 0$, we take

$t_1 = 2.750$ and $t_2 = 3.5$. In this manner, iterations are performed and the algorithm will be terminated once the optimality condition $h'(t) \leq \epsilon$ is reached.

In the following, a general MATLAB code is given for the same problem.

MATLAB Code

```
clc
clear all
str = input('Give an equation in t: ','s');
f = inline(str,'t');
a=2
b=5
k=1;
while k<=n
    t1=a;

    t2=b;
    t3=(a+b)/2;
    deltat=.001;
    Df1=(f(t1+deltat)-f(t1-deltat))/(2*deltat);
    Df2=(f(t2+deltat)-f(t2-deltat))/(2*deltat);
    Df3=(f(t3+deltat)-f(t3-deltat))/(2*deltat);
    if Df1*Df2>0
        disp('No minimum exist in (a,b)')
    elseif (Df1*Df3)>0
        a=t3;
    elseif (Df2*Df3)>0
        b=t3;
    end
    tmin=t3;
    fmin=f(t3);
    fprintf('%4d  %20e %14e %14e\n', k-1,t3,f(t3),Df3)
    k=k+1;
end
```

The sign of the first derivative at the midpoint of the current search region is used to eliminate half of the search region. If the derivative is negative, the minimum cannot lie in the left half of the search region, and if the derivative is positive, the minimum cannot lie in the right half of the search space.

The bisection method requires two function evaluations per iteration. In this method, exactly half the region is eliminated at every iteration; but using the magnitude of the gradient, a faster algorithm can be designed to adaptively eliminate variable portions of the search region a matter, which

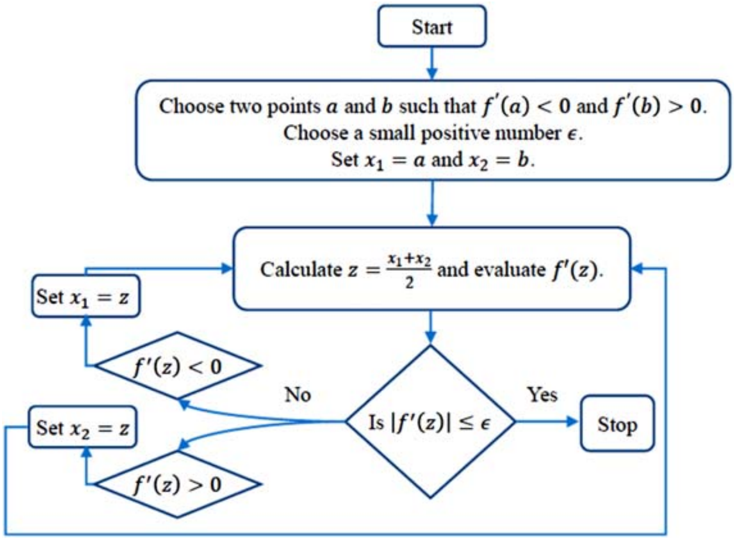

Start
Choose two points a and b such that $f'(a) < 0$ and $f'(b) > 0$.
Choose a small positive number ϵ.
Set $x_1 = a$ and $x_2 = b$.
Calculate $z = \frac{x_1+x_2}{2}$ and evaluate $f'(z)$.
Set $x_1 = z$
$f'(z) < 0$
No
Is $|f'(z)| \le \epsilon$
Yes
Stop
Set $x_2 = z$
$f'(z) > 0$

Figure 3.6 Flow diagram of the bisection method.

we discuss in Section 3.8. For better visualization and making of programming codes, a flowchart of the algorithm is presented in Fig. 3.6.

3.8 Newton–Raphson method

This method requires derivative information to find the optimum of the optimization problem. The optimality property that at a local or a global optimum the gradient is zero can be used to terminate the search process.

The goal of an unconstrained local optimization method is to achieve a point having as small a derivative as possible. In the Newton–Raphson method, a linear approximation to the first derivative of the function is made at a point using Taylor's series expansion. That expression is equated to zero to find the next guess. If the current point at iteration t is $x^{(t)}$, the point in the next iteration is governed by the following simple equation (obtained by considering up to the linear term in Taylor's series expansion):

$$x^{(t+1)} = x^{(t)} - \frac{f'\left(x^{(t)}\right)}{f''\left(x^{(t)}\right)}, f''\left(x^{(t)}\right) \neq 0 \tag{3.13}$$

Algorithm 3.7.

```
Step 1: Choose initial guess $x^{(1)}$ and a small positive number $\epsilon$.
Set $k = 1$. Compute $f'\left(x^{(1)}\right)$.
Step 2: Compute $f''\left(x^{(k)}\right)$.
Step 3: Calculate $x^{(k+1)} = x^{(k)} - \left(f'\left(x^{(k)}\right)\right)/\left(f''\left(x^{(k)}\right)\right)$, compute $f'\left(x^{(k+1)}\right)$.
Step 4: If $\left|f'\left(x^{(k+1)}\right)\right| < \epsilon$, terminate. Else set $k = k + 1$ and go to Step 2.
```

Example 3.7. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represents the height at t, defined in the interval $(0, 5)$. By using the Newton–Raphson method, find the value of t at which the height is minimum.

Solution: W have to first choose the initial guess value, that is, $t^{(1)} = 1$, and a termination parameter $\epsilon = 0.001$. Set the iteration number $k = 1$. Then, we have to compute the derivative using Eq. (3.14) with increment 0.01.

$$f''\left(x^{(t)}\right) = \frac{f\left(x^{(t)} + \Delta x^{(t)}\right) - 2f\left(x^{(t)}\right) + f\left(x^{(t)} - \Delta x^{(t)}\right)}{\left(\Delta x^{(t)}\right)^2} \tag{3.14}$$

The derivative $h'\left(t^{(1)}\right)$ is -52.005. The second derivative $h''\left(t^{(1)}\right)$ using Eq. (3.14) is 110.111. Now, we compute $t^{(2)} = t^{(1)} - \left(h'\left(t^{(1)}\right)/h''\left(t^{(1)}\right)\right) = 1 - \left(-(52.005/110.111)\right) = 1.473$. Again, by using Eq. (3.12), we obtain $h'\left(t^{(2)}\right) = -21.944$. This completes the first iteration. Since $\left|h'\left(t^{(2)}\right)\right| \nless \epsilon$, we have to perform another iteration set $k = 2$ and then repeat the procedure till the optimality condition $h'\left(t^{(k)}\right) < \epsilon$ is achieved.

In the following, a general MATLAB code of the Newton–Raphson method is given for the same problem.

MATLAB Code

```
clc
clear
str = input('Give an equation in x: ','s');
f = inline(str,'x');
epsilon = input ('Enter the value of Tolerance: ');
t1 = input('Enter the guess value: ');
k = 1;
if abs(t1)>0.01
    delta_t1 = 0.01*abs(t1);
else
    delta_t1 = 0.0001;
end
f_dash_t1 = (feval(f,(t1+delta_t1))-(feval(f,(t1-delta_t1))))/(2*delta_t1);
f_doubledash_t1 = (feval(f,(t1+delta_t1))-(2*feval(f,t1))+(feval(f,(t1-
delta_t1))))/(delta_t1^2);
t2 = t1 - f_dash_t1/f_doubledash_t1;
if abs(t2)>0.01
    delta_t2 = 0.01*abs(t2);
else
    delta_t2 = 0.0001;
end
f_dash_t2 = (feval(f,(t2+delta_t2))-(feval(f,(t2-delta_t2))))/(2*delta_t2);
while abs(f_dash_t2)>epsilon
    t1 = t2;
    if abs(t1)>0.01
        delta_t1 = 0.01*abs(t1);
    else
        delta_t1 = 0.0001;
    end
    f_dash_t1 = (feval(f,(t1+delta_t1))-(feval(f,(t1-
delta_t1))))/(2*delta_t1);
    f_doubledash_t1 = (feval(f,(t1+delta_t1))-(2*feval(f,t1))+(feval(f,(t1-
delta_t1))))/(delta_t1^2);
    t2 = t1 - f_dash_t1/f_doubledash_t1;
    if abs(t2)>0.01
        delta_t2 = 0.01*abs(t2);
    else
        delta_t2 = 0.0001;
    end
    f_dash_t2 = (feval(f,(t2+delta_t2))-(feval(f,(t2-
delta_t2))))/(2*delta_t2);
    k = k+1;
end
fprintf('The solution lies at %f \n & it has been obtained after %f \n
iterations \n',t2,k);
```

The convergence of the algorithm depends on the initial point and the nature of the objective function. For mathematical functions, the derivative may be easy to compute, but in practice, the gradients have to be computed numerically. At a point $x^{(t)}$, the first- and second-order derivatives are computed as follows, using the central difference method.

In many real-world problems, it is difficult to obtain information about derivatives, either due to the nature of the problem or due to the computations involved in calculating the derivatives. Despite these difficulties, gradient-based methods are popular and are often found effective. However, it is recommended to use these algorithms in problems where the derivative information is available or can be calculated easily.

3.9 Secant method

This is also a gradient-based method, which follows the same condition $f'(a) \cdot f'(b) < 0$ defined in the bisection method. As such, both sign and magnitude of the derivatives are considered for creating a new point. It is assumed that the derivative of the function vary linearly between the two chosen boundary points that x_1 and x_2. Since $f'(x_1) \cdot f'(x_2) < 0$ and the derivatives vary linearly between the boundary points, there exists a point between these two points with a zero derivative. Hence, the following point z can be created, such that the derivative is zero at the same point:

$$z = x_2 - \frac{f'(x_2)}{\left(\frac{f'(x_2) - f'(x_1))}{x_2 - x_1}\right)}. \tag{3.15}$$

In this method, in one iteration, more than half the search space may be eliminated depending on the gradient values at the two chosen points. However, smaller than half the search may also be eliminated in one iteration.

Algorithm 3.8.

The algorithm is same as the bisection method except Step 2 is modified as follows: *Step 2*: Compute the new point z using Eq. (3.15) and evaluate $f'(z)$.

This algorithm also requires only one gradient evaluation at every iteration. Thus, only two function values are required per iteration.

3.10 Successive quadratic point estimation method

It is noticed that previous methods provide optimal bound solutions having only iterative function values of two points were considered to guide the search. But the function values at the chosen points may also provide useful information about the location of the minimum in the search space. As such, the successive quadratic estimation method described below represents a class of point estimation methods that use both the magnitude and the sign of function values to steer the search. The basic idea of this method is to compute the function values at several chosen points, and a unimodal function is fitted through these points exactly. Then, the minimum of the fitted function is considered to be a guess of the minimum of the original objective function. In this algorithm, the fitted curve is a quadratic polynomial. As a quadratic function can be defined with three points, we begin the algorithm with three initial points. The original function and three initial points x_1, x_2, and x_3 are shown in Fig. 3.7. Also, a fitted quadratic polynomial through these three points is drawn with a dashed line. The minimum point $\tilde{x}$ of this polynomial is used as one of the candidate points for the next iteration. For nonquadratic polynomials, a number of iterations of this algorithm is necessary, whereas for quadratic objective functions, the exact minimum can be found in one iteration only.

A general quadratic polynomial passing through two points x_1 and x_2 can be written as follows:

$$q(x) = a_0 + a_1(x - x_1) + a_1(x - x_1)(x - x_2). \tag{3.16}$$

If (x_1,f_1), (x_2,f_2), and (x_3,f_3) are three points on this function, then the following relationships can be obtained.

$$a_0 = f_1, \tag{3.17}$$

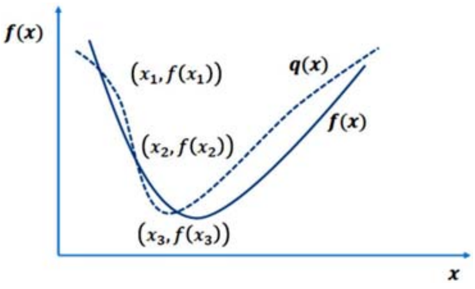


Figure 3.7 Diagram for the function $f(x)$ and fitted quadratic polynomial $q(x)$.

$$a_1 = \frac{f_2 - f_1}{x_2 - x_1}, \tag{3.18}$$

$$a_2 = \frac{1}{x_3 - x_2}\left(\frac{f_3 - f_1}{x_3 - x_1} - a_1\right). \tag{3.19}$$

By differentiating $q(x)$ with respect to x and setting it to zero, it can be shown that the minimum of the above function is

$$\tilde{x} = \frac{x_1 + x_2}{2} - \frac{a_1}{2a_2}. \tag{3.20}$$

The above point is an estimate of the minimum point provided $q''(\overline{x}) > 0$ or $a_2 > 0$, which depends only on the choice of the three basic points. Among the four points x_1, x_2, x_3, and $\overline{x}$, the best three points are kept and a new interpolated function $q(x)$ is found again. This procedure continues until two consecutive estimates are close to each other.

Based on these results, we present Powell's algorithm

Algorithm 3.9.

Step 1: Let x_1 be an initial point and Δ be the step size. Compute $x_2 = x_1 + \Delta$.

Step 2: Evaluate $f(x_1)$ and $f(x_2)$.

Step 3: If $f(x_1) > f(x_2)$, let $x_3 = x_1 + 2\Delta$; Else let $x_3 = x_1 - \Delta$. Evaluate $f(x_3)$.

Step 4: Determine $F_{\min} = \min(f_1, f_2, f_3)$, and $X_{\min}$ is the point x_i that corresponds to $F_{\min}$.

Step 5: Use points x_1, x_2, and x_3 to calculate $\tilde{x}$ using Eq. (3.20).

Step 6: Are $|F_{\min} - f(\tilde{x})|$ and $|X_{\min} - \tilde{x}|$ small? If not, go to Step 7. Else the optimum is the best of current four points and terminate.

Step 7: Save the best point and two bracketing it, if possible; otherwise, save the best three points. Relabel them according to $x_1 < x_2 < x_3$ and go to Step 4.

Example 3.8. Consider an object is moving on a path having function $h(t) = (54/t) + t^2$, where $h(t)$ represent the height at t, defined in the interval $(0, 5)$. By using the Newton–Raphson method, find the value of t at which the height is minimum.

Solution: We have to choose $t_1 = 1$, and $\Delta = 1$.

Thus, $t_2 = t_1 + \Delta = 1 + 1 = 2$. Compute $h(t_1) = 55$ and $h(t_2) = 31$.

Since $h(t_1) > h(t_2)$, we take $t_3 = 1 + 2\Delta = 1 + 2 = 3$ and the height is $h(t_3) = 27$.

Now, comparing the heights at different t, we obsrve that the minimum height is $H_{\min} = \min(55, 31, 27)$ and the corresponding point is $t_{\min} = t_3 = 3$.

By using Eqs. (3.17)–(3.19), we obtain $a_0 = h(t_1) = 55$, $a_1 = (h(t_2) - h(t_1))/(t_2 - t_1) = -24$, and $a_2 = \frac{1}{t_3 - t_2}\left((h(t_3) - h(t_1))/(t_3 - t_1) - a_1\right) = 10$. Since $a_2 > 0$, the estimated minimum is $\tilde{t} = (t_1 + t_2)/2 - (a_1/2a_2) = 2.7$.

The corresponding height is $h(\tilde{t}) = 27.29$. Now $|27 - 27.29| = 0.29 \not< \epsilon = 0.001$ and $|3 - 2.7| \not< \epsilon = 0.001$. Thus, we have to perform next iteration and the best three points for the next iteration are $t_1 = 2$, $t_2 = 2.7$, and $t_3 = 3$.

Then, the algorithm is applied again to find nest best three points and the iterations will be terminated when the optimality condition ($|F_{\min} - f(\tilde{t})|$ and $|t_{\min} - \tilde{t}|$ are small or $\left|F_{\min} - f(\tilde{t})\right| < \epsilon$ and $|t_{\min} - \tilde{t}| < \epsilon$) is satisfied.

In the above algorithm, no check is made to satisfy $a_2 > 0$. The same can be incorporated in Step 5. If a_2 is found to be negative, one of the three points may be replaced by a random point. This process is continued until the quantity a_2 becomes nonnegative.

From the quadratic point estimation, we found the following conclusions.

1 For well-behaved unimodal functions, obtain a minimum point faster than the region elimination methods.

2 For skewed functions, the golden section search is better than the successive quadratic point estimation method.

Practice set

1. Find the optimum points of the function $f(x) = x^3 - 10x - 2x^2 + 10$.
2. Find the optimum points of the function

$$f(x) = \begin{cases} 2 - (x-1)^2, & 0 \le x < 3 \\ -3 + (x-4)^2, & 3 \le x \le 6 \end{cases}.$$

3. Identify the optimum points of the following functions. Find the optimum function values.
   a. $f(x) = x^3 - 10x - 2x^2 + 10$
   b. $f(x) = 2x - x^3 - 5x^2 - 2e^{0.01x}$
   c. $f(x) = 2(x-2)e^{x-2} - (x+3)^2$
   d. $f(x) = e^x - x^3$
   e. $f(x) = 0.01x^5 - 2x^4 + 500(x-2)^2$
4. Write the differences between the bounding phase method and the exhaustive search method for single-variable optimization.

5. What proportion of the original search space is retained after 10 function evaluations using
   a. golden section search,
   b. interval halving method and
   c. bisection method. *dx*
6. Under what circumstances can the condition $(df(x))/dx = 0$ not be used to find the minimum of the function $f(x)$?
7. Prove that a convex function is unimodal.
8. Compare the ratios of intervals of uncertainty L_{n/L_0} obtainable in the following methods for $n = 2, 3, \cdots, 10$.
   a. Exhaustive search method
   b. Interval halving method
   c. Fibonacci method
   d. Golden section search method
9. Find the value of t in the interval $(0, 1)$, which minimizes the function $f(t) = t(t - 1.5)$ to within ± 0.05 by
   a. golden section search method
   b. Fibonacci method.
10. Write two disadvantages of the Fibonacci search method for single-variable optimization.
11. What proportion of the original search space is retained after five function evaluations using the golden section search method?
12. Compare the golden section search and the interval halving method in terms of the obtained interval after 10 function evaluations for the minimization of the function $f(x) = x^2 - 10e^{0.1x}$ in the interval $(-10, 5)$.
13. Use three iterations of the golden section search method to maximize the function $f(x)=10+x^3 - 2x - 5e^{0.1x}$ in the interval $(-5, 5)$.
14. Find at least one root of the following functions:
    a. $f(x)=(2x-5)^4 - \left(x^2-1\right)^3$
    b. $f(x)=(x+10)^2 - 0.01x^4$
    c. $f(x) = ((x+2)^2+10)^2 - x^4$
15. Bracket the minimum of the following functions using the bounding phase method. Use an initial point $x^{(0)} = 0$ and an initial $\Delta = 1$ in all cases. Also, use the golden section search code to find the minimum point with the three decimal places of accuracy.
    a. $f(x)=x^2 - 3x - 20$
    b. $f(x)=x^3 - 2x + 10$
    c. $f(x)=(1-x)^4 - (2x+10)^2$
    d. $f(x)=e^x-400x^3 + 10$
    e. $f(x)=0.1\left(x^2-3x+5\right)^2 + (x-3)^2$

16. Use three iterations of the bisection and the secant method to minimize the following functions $e^{(0.2x)}-(x+3)^2-0.01x^4$.

Compare the algorithms in terms of the interval obtained at the end of three iterations.

Further reading

Bazaraa, M., & Shetty, C. (1979). *Nonlinear programming: Theory and algorithms*. New York: Wiley.

Beveridge, G., & Schechter, R. (1970). *Optimization: Theory and practice*. New York: McGraw-Hill.

Bhatti, M. A. (2013). *Practical optimization methods: With Mathematica® applications*. New York: Springer Verlag.

Chakraverty, S., & Nayak, S. (2017). *Neutron diffusion: Concepts and uncertainty analysis for engineers and scientists*. Boca Raton, FL: CRC Press.

Deb, K. (2004). *Optimization for engineering design algorithms and examples*. New Delhi: Prentice-Hall of India.

Edwin, C. K., & Stanislaw, Z. H. (2013). *An introduction to optimization*. New Delhi: Wiley.

Grewal, B. S. (2006). *Higher engineering mathematics*. New Delhi: Khanna Publishers.

Gue, R., & Thomas, M. (1968). *Mathematical methods of operations research*. New York: 1968.

Hancock, H. (1960). *Theory of maxima and minima*. New York: Dover.

Howell, J., & Buckius, R. (1992). *Fundamentals of engineering thermodynamics*. New York: McGraw-Hill.

Jaluria, Y. (2007). *Design and optimization of thermal systems*. Boca Raton, FL: CRC Press.

Johnson, R. C. (1980). *Optimum design of mechanical elements*. New York: Wiley.

Kolman, B., & Trench, W. (1971). *Elementary multivariable calculus*. New York: Academic Press.

Kuhn, H., & Tucker, A. (1951). *Nonlinear programming*. *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, CA: University of California Press.

Levenson, M. (1967). *Maxima and minima*. New York: Macmillan.

Panik, M. (1976). *Classical optimization: Foundations and extensions*. Amsterdam: North-Holland.

Powell, M. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 155–162.

Reklaitis, G., Ravindran, A., & Ragsdell, K. (1983). *Engineering optimization—Methods and applications*. New York: Wiley.

Richmond, A. (1972). *Calculus for electronics*. New York: McGraw-Hill.

Simmons, D. (1975). *Nonlinear programming for operations research*. Englewood Cliffs, NJ: Prentice-Hall.

Thomas, G. (1967). *Calculus and analytic geometry*. Boston, MA: Addison-Wesley.

CHAPTER FOUR

Multivariable unconstrained nonlinear optimization

4.1 Classical method for multivariable optimization

This section includes the necessary and sufficient conditions for the minimum or maximum of an unconstrained function of several variables. The basis of multivariable optimization problems and its theoretical study primarily depends on Taylor's series expansion of a multivariable function. As such, initially, we will discuss Taylor's series expansion of a multivariable function.

4.1.1 Definition: *r*th differential of a function *f*(**X**)

As a multivariable function has two or more variables, the derivatives of the same are nothing but the partial derivatives only. Hence, always a picture of partial derivatives comes into play. If all partial derivatives of a function $f(X)$ through order $r \geq 1$ exist and are continuous at a point X^*, then the polynomial

$$d^r f(X^*) = \sum_{i=1}^{n} \sum_{j=1}^{n} \cdots \sum_{k=1}^{n} h_i h_j \cdots h_k \frac{\partial^r f(X^*)}{\partial x_i \partial x_j \cdots \partial x_k}, \tag{4.1}$$

is called the *r*th differential of f at X^*. It is noted that there are r summations, and one h_i is associated with each summation in Eq. (4.1). The h_i corresponds the small increment along x_i, where $i = i, j, \cdots, k$.

© 2020 Elsevier Inc.
All rights reserved.

Consider the following example problem to understand *r*th differential of function $f(X)$.

Example 4.1. Let $r=2$ and $n=3$, then using Eq. (4.1) we have the second-order differential:

$$d^2f(X^*) = d^2f\left(x_1^*, x_2^*, x_3^*\right) = \sum_{i=1}^{3}\sum_{j=1}^{3} h_i h_j \frac{\partial^r f(X^*)}{\partial x_i \partial x_j}$$

$$= h_1^2 \frac{\partial^2 f}{\partial x_1^2}(X^*) + h_2^2 \frac{\partial^2 f}{\partial x_2^2}(X^*) + h_3^2 \frac{\partial^2 f}{\partial x_3^2}(X^*) + 2h_1h_2 \frac{\partial^2 f}{\partial x_1 \partial x_2}(X^*)$$
$$+ 2h_2h_3 \frac{\partial^2 f}{\partial x_2 \partial x_3}(X^*) + 2h_1h_3 \frac{\partial^2 f}{\partial x_1 \partial x_3}(X^*)$$

The aforementioned *r*th differential can be used, and Taylor's series expansion of a function $f(X)$ about a point X^* is given by

$$\begin{aligned} f(X) = &f(X^*) + df(X^*) + \frac{1}{2!} d^2f(X^*) + \frac{1}{3!} d^3f(X^*) \\ &+ \cdots + \frac{1}{N!} d^Nf(X^*) + R_N(X^*, h), \end{aligned} \tag{4.2}$$

where the last term $R_N(X^*, h)$ is called the remainder term, which is represented as follows:

$$R_N(X^*, h) = \frac{1}{(N+1)!} d^{N+1} f(X^* + \theta h), \tag{4.3}$$

where $0 < \theta < 1$ and $h = X - X^*$.

Example 4.2. Find the second-order Taylor's series approximation of the function

$$f(x_1, x_2, x_3) = x_2^2 x_3 + x_1 e^{x^3}$$

about the point $X^* = (1, 0, -2)^T$.

Solution: The second-order Taylor's series approximation of the function f about the point X^* is given by

$$f(X) = f\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} + df\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} + \frac{1}{2!}d^2f\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix},$$

where $f\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = 0 \times (-2) + 1 \times e^{-2} = e^{-2}$.

$$df\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = h_1\frac{\partial f}{\partial x_1}\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} + h_2\frac{\partial f}{\partial x_2}\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} + h_3\frac{\partial f}{\partial x_3}\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

$$= \left[h_1e^{x_3} + h_2(2x_2x_3) + h_3x_2^2 + h_3x_1e^{x_3}\right]\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

$$= h_1e^{-2} + h_3e^{-2}.$$

$$d^2f\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix} = \sum_{i=1}^{3}\sum_{j=1}^{3}h_ih_j\frac{\partial^2 f}{\partial x_i\partial x_j}\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

$$= \left(h_1^2\frac{\partial^2 f}{\partial x_1^2} + h_2^2\frac{\partial^2 f}{\partial x_2^2} + h_3^2\frac{\partial^2 f}{\partial x_3^2} + 2h_1h_2\frac{\partial^2 f}{\partial x_1\partial x_2} + 2h_2h_3\frac{\partial^2 f}{\partial x_2\partial x_3} + 2h_1h_3\frac{\partial^2 f}{\partial x_1\partial x_3}\right)\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

$$= \left[h_1^2(0) + h_2^2(2x_3) + h_3^2\left(x_1e^{x_3}\right) + 2h_1h_2(0) + 2h_2h_3(2x_2) + 2h_1h_3\left(e^{x_3}\right)\right]\begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

$$= -4h_2^2 + e^{-2}h_3^2 + 2h_1h_3e^{-2}.$$

Thus, Taylor's series approximation is represented by

$$f(X) = e^{-2} + e^{-2}(h_1 + h_3) + \frac{1}{2!}\left(-4h_2^2 + e^{-2}h_3^2 + 2h_1h_3e^{-2}\right)$$

where $h_1 = x_1 - 1$, $h_2 = x_2$, and $h_3 = x_3 + 2$.

Following are the necessary and sufficient conditions of multivariable function for the existence of an extreme point.

4.1.2 Necessary condition

If $f(X)$, $X = (x_1, x_2, \cdots, x_n)^T$ has an extreme point (maximum or minimum) at $X = X^*$, and if the first partial derivatives of $f(X)$ exist at X^*, then

$$\frac{\partial f}{\partial x_1}(X^*) = \frac{\partial f}{\partial x_2}(X^*) = \cdots = \frac{\partial f}{\partial x_n}(X^*) = 0. \tag{4.4}$$

4.1.3 Sufficient condition

A sufficient condition for a stationary point X^* to be an extreme point is that the matrix of second-order partial derivatives (Hessian matrix) of $f(X)$ evaluated at X^*.

1. Positive definite when X^* is a relative minimum point
2. Negative definite when X^* is a relative maximum point
3. Indefinite when X^* is a neither a relative minimum nor a relative maximum point

   The matrix of the form

$$H = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]|_{X=X^*} \tag{4.5}$$

is called the Hessian matrix of function f at $X = X^*$.

Also, using the following two different ways, the positive or negative definiteness of the Hessian matrix can be obtained.

1. If all the eigenvalues of H are positive, then H is a positive definite matrix. If all the eigenvalues of H are negative, then H is negative definite. Otherwise, the matrix is called indefinite.
2. The positive or negative definiteness can be obtained by its principal minors. If all the principal minors are positive, then H is positive definite. If the principal minors occur in the following alternate sign, then H is called negative definite.

$$H_1 = \left|\frac{\partial^2 f}{\partial x_1^2}\right|_{X=X^*}$$

is negative,

$$H_2 = \left| \begin{matrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} \end{matrix} \right|_{X=X^*}$$

is positive, and

$$H_2 = \left| \begin{matrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_3} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \dfrac{\partial^2 f}{\partial x_2 \partial x_3} \\ \dfrac{\partial^2 f}{\partial x_3 \partial x_1} & \dfrac{\partial^2 f}{\partial x_3 \partial x_2} & \dfrac{\partial^2 f}{\partial x_3^2} \end{matrix} \right|_{X=X^*}$$

is negative, and so on.

Otherwise, H is indefinite.

Generally, when the Hessian matrix is of higher order, then the second case (principal minor form) is easier to obtain.

In the successive sections, we will discuss numerical optimization methods to obtain the minimum of multivariable unconstrained optimization problem.

4.2 Unidirectional search method

One of the very basic search techniques for multivariable optimization is a unidirectional search method. The idea of this method is to convert the multivariable function to a single variable and then investigate the optimum. Many multivariable optimization techniques use successive unidirectional search techniques to find the minimum point along a particular search direction. Let us illustrate here how a unidirectional search can be performed on a multivariable function.

A unidirectional search is a one-dimensional search performed by comparing function values only in a specified direction. Usually, a unidirectional search is performed from a point x in a specified direction s.

That is, only points that lie on a line (in an N-dimensional space) passing through the point x and oriented along the search direction s are allowed to be considered in the search process. Any arbitrary point on that line can be expressed as follows:

$$x(\alpha) = x + \alpha s \tag{4.6}$$

The parameter α is a scalar quantity, specifying a relative measure of the distance of the point $x(\alpha)$ from x. Note, however, that Eq. (4.6) is a vector equation specifying all design variables $x_i(\alpha)$. Thus, for a given value of α, the point $x(\alpha)$ can be known. Any positive and negative value of α will create a point on the desired line. If $\alpha = 0$, the current point x is obtained. To find the minimum point on the specified line, we can rewrite the multivariable objective function in terms of a single variable α by substituting each variable x_i taking the expression $x_i(\alpha)$ given in Eq. (4.6) and by using a suitable single-variable search method described in Chapter three, Single-variable nonlinear optimization. Once the optimal value α^* is found, the corresponding point can also be found using Eq. (4.6).

4.3 Evolutionary search method

The important factor that involves search methods is the directions in which the current point explored and reached to the optimum point. In a single-variable optimization problem, the single-variable function possesses only two search directions that are either positive or negative direction of the field variable. For example, if the single-variable function is $f(x)$, then the current point can be explored along positive or negative x direction. As such, if we generalize this concept then for two-variable function $f(x, y)$, there are four different directions (each variable having two directions). Hence, in the multivariable objective function optimization problem, each variable can be explored either in the positive or in the negative direction, thereby totaling 2^N (N is the number of independent or field variables) different ways. But, the extent of increment or decrement in each direction depends on the current point and the objective function. In the following, we are considering one of the search techniques, viz., Box's evolutionary search method to obtain an optimal solution.

4.3.1 Box's evolutionary optimization method

The basic idea of this method is to construct a N-dimensional hypercube where the center point is the current point. There are 2^N corner points, and all together corner points and current points we will get $2^N + 1$ points. For all $2^N + 1$ points, function values are obtained and compared. Then, the best point is identified. Next, an improved point is formed around this best point, and the process continues. But there may occur some situations where the best point and current point have the same function values or an improved point is not found, then at that instance, size of the hypercube is reduced and again process continues until the hypercube becomes very small. In the following, the detailed algorithm of this method is given.

Algorithm 4.1.

Step 1: Choose an initial point $x^{(0)}$ and size reduction parameters Δ_i for all design variables, $i = 1, 2, \cdots, N$. Choose a termination parameter ϵ. Set, $\overline{x} = x^{(0)}$.
Step 2: If $\|\Delta\| < \epsilon$, then stop. Otherwise, construct a hypercube that is create 2^N points by adding and subtracting $\Delta_i/2$ from each design variable at the point $\overline{x}$.
Step 3: Compute function values at all $2^N + 1$ points. Find the point that gives minimum function value. Denote the minimum point to be $\overline{x}$. *Step 4*: If $\overline{x} = x^{(0)}$, then reduce size parameters Δ_i by 2 that is $\Delta_i = \frac{\Delta_i}{2}$ and go to Step 2. Else set $x^{(0)} = \overline{x}$ and go to Step 2.

From the Algorithm 4.1, we observed that $x^{(0)}$ is always considered as the current best point. Hence, at the end of all the iterations, $x^{(0)}$ becomes the obtained optimum point. Again, it is noticed from the Algorithm 4.1 that at most 2^N functions are evaluated at each iteration. As such, the required number of function evaluations increases exponentially with the value of N. However, the algorithm is simple to implement and useful in solving many industrial optimization problems successfully. For better understanding and computer programming of this algorithm, flow chart of Box's evolutionary optimization method is shown in Fig. 4.1.

We illustrate this algorithm through an Example 4.3.

Example 4.3. Solve the following nonlinear system of equations in the interval $0 \leq x, y \leq 5$, by using Box's evolutionary optimization algorithm.

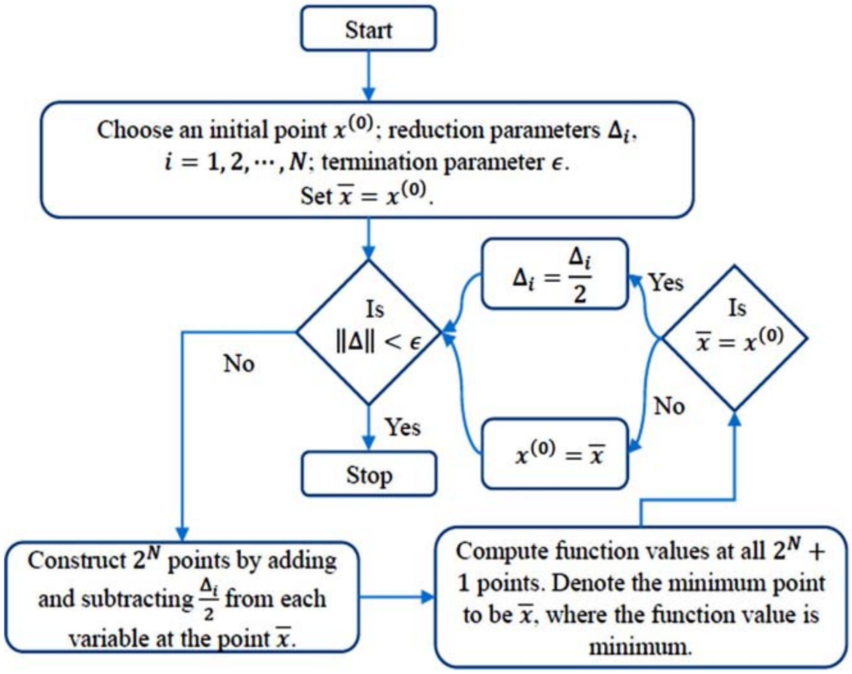

Start
Choose an initial point $x^{(0)}$; reduction parameters Δ_i, $i = 1, 2, \cdots, N$; termination parameter ϵ. Set $\bar{x} = x^{(0)}$.
Is $\|\Delta\| < \epsilon$
No
Yes
Stop
$\Delta_i = \frac{\Delta_i}{2}$
Yes
Is $\bar{x} = x^{(0)}$
No
$x^{(0)} = \bar{x}$
Construct 2^N points by adding and subtracting $\frac{\Delta_i}{2}$ from each variable at the point $\bar{x}$.
Compute function values at all $2^N + 1$ points. Denote the minimum point to be $\bar{x}$, where the function value is minimum.

Figure 4.1 Flow diagram of Box's evolutionary optimization method.

Take, the initial point $X^{(0)} = \left(x^{(0)}, y^{(0)}\right)^T = (1, 1)^T$ and size reduction parameter $\Delta = (2, 2)^T$.

$$x^2 + y = 11$$

$$x + y^2 = 7$$

Solution: Before applying the Box's evolutionary optimization algorithm, we need to convert the nonlinear system of equations into the following unconstrained optimization problem.

$$\text{Minimize}\, f(x, y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right)^2$$

We can initialize the algorithm by the initial point $X^{(0)} = (1, 1)^T$. Consider a termination parameter $\epsilon = 0.001$. Since $\|\Delta\| = \sqrt{\left(2^2 + 2^2\right)} = \sqrt{8} = 2.8284 > 0.001$, we construct a square around $X^{(0)}$.

Hence, $X^{(1)} = (0, 0)^T$, $X^{(2)} = (2, 0)^T$, $X^{(3)} = (0, 2)^T$, and $X^{(4)} = (2, 2)^T$.

The function values at the five points are $f\left(X^{(0)}\right) = 106$, $f\left(X^{(1)}\right) = 170$, $f\left(X^{(2)}\right) = 74$, $f\left(X^{(3)}\right) = 90$, and $f\left(X^{(4)}\right) = 26$.

The minimum of five function values is 26, and the corresponding point is $X^{(4)} = (2,2)^T$. So, we denote $\overline{X} = (2,2)^T$.

Since $\overline{X} \neq X^{(0)}$, we can set $X^{(0)} = (2,2)^T$ and go to Step 2 of Algorithm 4.1. This completes the first iteration. Now, the initial point moved from $(1,1)^T$ to $(2,2)^T$.

Second iteration:

As there is no change in Δ value, so $\|\Delta\| = 2.8284 > 0.001$. Construct a square around the point $(2,2)^T$, and the corner points of the square are as follows:

$$X^{(1)} = (1,1)^T,\ X^{(2)} = (3,1)^T,\ X^{(3)} = (1,3)^T, \text{ and } X^{(4)} = (3,3)^T$$

Corresponding function values are as follows:

$$f(X^{(0)}) = 26, f(X^{(1)}) = 106, f(X^{(2)}) = 10, f(X^{(3)}) = 58, \text{ and} f(X^{(4)}) = 26.$$

The minimum point occurs at $\overline{X} = (3,1)^T$, and the corresponding function value at $\overline{X}$ is 10. Since $\overline{X} \neq X^{(0)}$, we can set $X^{(0)} = (3,1)^T$ and go to Step 2 of Algorithm 4.1. This completes the second iteration. Now, the initial point is moved from $(2,2)^T$ to $(3,1)^T$.

Third iteration:

Using Algorithm 4.1, construct a square around $(3,1)^T$ and the corner points are represented as follows:

$$X^{(1)} = (2,0)^T,\ X^{(2)} = (4,0)^T,\ X^{(3)} = (2,2)^T, \text{ and } X^{(4)} = (4,2)^T.$$

Corresponding function values are as follows:

$f\left(X^{(0)}\right) = 10$, $f\left(X^{(1)}\right) = 74$, $f\left(X^{(2)}\right) = 34$, $f\left(X^{(3)}\right) = 26$, and $f\left(X^{(4)}\right) = 50$. Here, the minimum function value is 10, which occur at $(3,1)^T$. Since the new point is the same as the previous point that is $\overline{X} = X^{(0)}$, we reduce the parameter $\Delta = \frac{\Delta}{2} = (1,1)^T$ and go to Step 2. This complete third iteration.

Fourth iteration:

Here, $\|\Delta\| = \sqrt{2} = 1.4142 > 0.001$, construct a square around $(3,1)^T$ and the corner points are as follows:

$$X^{(1)} = (2.5,0.5)^T,\ X^{(2)} = (3.5,0.5)^T,\ X^{(3)} = (2.5,1.5)^T, \text{ and } X^{(4)} = (3.5,1.5)^T.$$

After finding and then comparing the corresponding function values at five points, we get minimum value of the function at $(3.5,1.5)^T$ with the function value 9.125. This completes the fourth iteration.

Similarly, all the possible iterations are performed until the optimum criteria, that is, $\|\Delta\| < 0.001$ is achieved. A general MATLAB code for the same is given as follows.

MATLAB Code

```
function funval = evofun(x)
x1 = x(1);
x2 = x(2);
fval = ((x1.^2+ x2 -11).^2 + (x1 + x2.^2-7).^2);
end

x0 =  input('Enter the initial point = ');
dt  = input('Enter the size reduction parameter delta = ');
tol = input('Enter the value of Tolerance = ');
y = x0;
dt1 = (sqrt(sum(dt.^2)));
j=1;
while dt1 > tol
    p1 = x0 - dt./2;
    p2 = x0 + ([-(dt(1)),dt(2)])./2;
    p3 = x0 + ([dt(1),-(dt(2))]).2;
    p4 = x0 + dt./2;
    x = [x0;p1;p2;p3;p4]'
    f = zeros (1,5);
    for i=  1:5
        f(i) = evofun(x(:,i))
    end
    [fmin , idx] = min(f(:,:))
    y = x(:,idx);
    i=1;
    funval(j) = fmin;
    h(j,:) = y;
    fprintf('Iteration number is %d \n', 'Point is (%d,%d) \n' Function
value is %.4f \n',j,y,fmin);
```

4.4 Simplex search method

The simplex search method consider less number of points in comparison with Box's evolutionary optimization method. Due to the less number of points, the total number of function evaluations reduces. As a result, this method performs nicely and computationally efficiently. So, let us discuss the procedure and algorithm of this method for practical implementation. The idea of this method is to construct a simplex and then compare the function values at the corner points of the simplex. Based on the comparison, identify the best point, worst point, and then next to the worst point. Find the reflected point of the worst point through the centroid and compare the same with the previous three points. According to the prescribed formulation, decide the best point and then proceed till the optimal condition is achieved. Here, for N variables, only $N+1$ points are used in the initial simplex. It is noted that the points chosen for the initial simplex should not form a zero volume N-dimensional hypercube. For example, in a two-variable function, the chosen three points in the simplex should not be collinear. Similarly, in a three-variable function, four points in the simplex should not be coplanar. If we see the algorithm, it is observed that at each iteration, the worst point in the simplex is found first. Then, a new simplex is created from the old point in the simplex by some fixed rules that drive the search away from the worst point in the simplex. But the quantity of steering depends on the relative function values of the simplex. As such, this arises four different situations, which are depicted in Figs. 4.2–4.5.

Initially, the centroid x_c of all excluding the worst point is calculated. Then, the worst point in the simplex is reflected about the

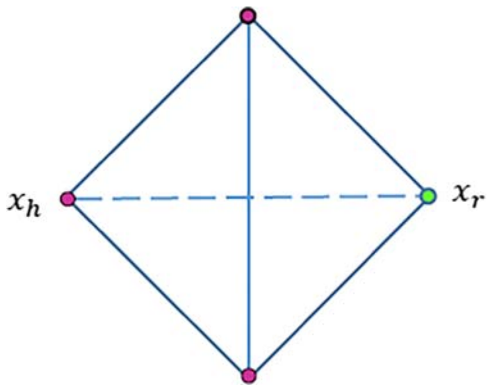


Figure 4.2 Reflection.

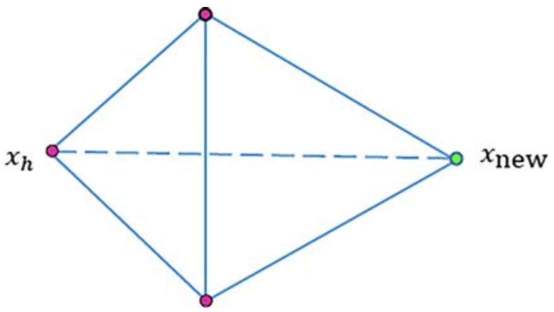


Figure 4.3 Expansion, $\gamma > 1$.

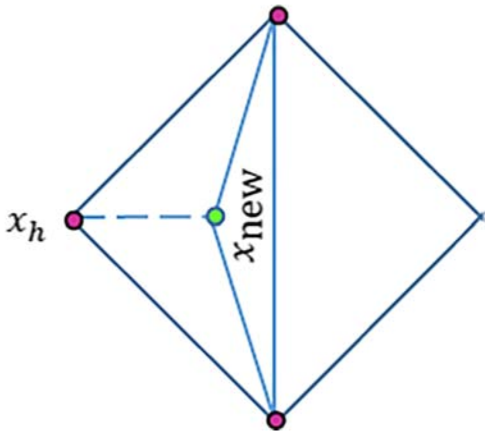


Figure 4.4 Contraction, $\beta < 1$.

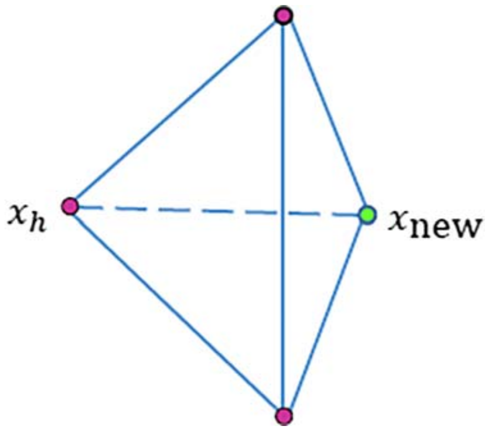


Figure 4.5 Contraction, $\beta > 1$.

centroid, and a new point x_r is found. The reflection of the worst point about the centroid is shown in Fig. 4.2. If the function value at x_r is found better than the best point in the simplex, then the expansion rule is applied to the simplex that represented in Fig. 4.3. For expansion, the extent of steering is controlled by the factor γ. If the function value at x_r is worse than the worst in the simplex, then a contraction in the direction from the centroid to the reflected point is made. The same can be illustrated in Fig. 4.4. The extent of contraction is controlled by the factor β. In this case, a negative value of β is considered for evaluation. Finally, if the function value at x_r is better than the worst and worse or lies in between the best and worst point, then a contraction with positive β value is performed. Contraction with positive β value is shown in Fig. 4.5. Otherwise, the obvious case is the reflected point x_r itself. To continue the algorithm obtained, a new point replaces the worst point in the simplex.

Algorithm 4.2.

Step 1: Assume expansion parameter $\gamma > 1$, contraction parameter $\beta \in (0,1)$, and a tolerance value ϵ. *Step 2*: Create an initial simplex by finding three points x_h (the worst point), x_l (the best point), and x_g (next to the worst point), such that $f(x_h) < f(x_g) < f(x_l)$. Then, calculate the centroid

$$x_c = \frac{1}{N} \sum_{i=1, i\neq h}^{N+1} x_i$$

where N is the number of dimensions. *Step 3*: Compute the reflected point $x_r = 2x_c - x_h$. Set $x_{\text{new}} = x_r$. If $f(x_r) < f(x_l)$, set $x_{\text{new}} = (1+\gamma)x_c - \gamma x_h$ (expansion); Else if $f(x_r) \geq f(x_h)$, set $x_{\text{new}} = (1-\beta)x_c + \beta x_h$ (contraction); Else if $f(x_g) < f(x_r) < f(x_h)$, set $x_{\text{new}} = (1+\beta)x_c - \beta x_h$ (contraction). Evaluate $f(x_{\text{new}})$ and replace x_h by x_{new}. *Step 4*: If $M = \left\{ \sum_{i=1}^{N+1} \frac{(f(x_i)-f(x_c))^2}{N+1} \right\}^{\frac{1}{2}} \leq \epsilon$, terminate. Else go to.

Step 2: In this algorithm, the values of β and γ play an important role to steer the search process. If a large value of γ or small value of β is used, the approach to the optimum point may be faster, but the convergence to the optimum point may be difficult. Conversely, smaller values of γ or larger value of β may require more function evaluations to converge near the optimum point. In general, the values of parameters γ and β are set as 2 and 0.5, respectively. For a systematic approach and easy implementation of programing codes, a flow diagram of the simplex search method is illustrated in Fig. 4.6.

Example 4.4. Solve the following nonlinear system of equations in the interval $0 \le x, y \le 5$ by using the simplex search algorithm. Take initial simplex with points $X^{(1)} = (0,0)^T$, $X^{(2)} = (2,0)^T$, and $X^{(3)} = (1,1)^T$; $\gamma = 1.5$, $\beta = 0.5$, and a tolerance value $\epsilon = 10^{-3}$.

$$x^2 + y = 11$$

$$x + y^2 = 7$$

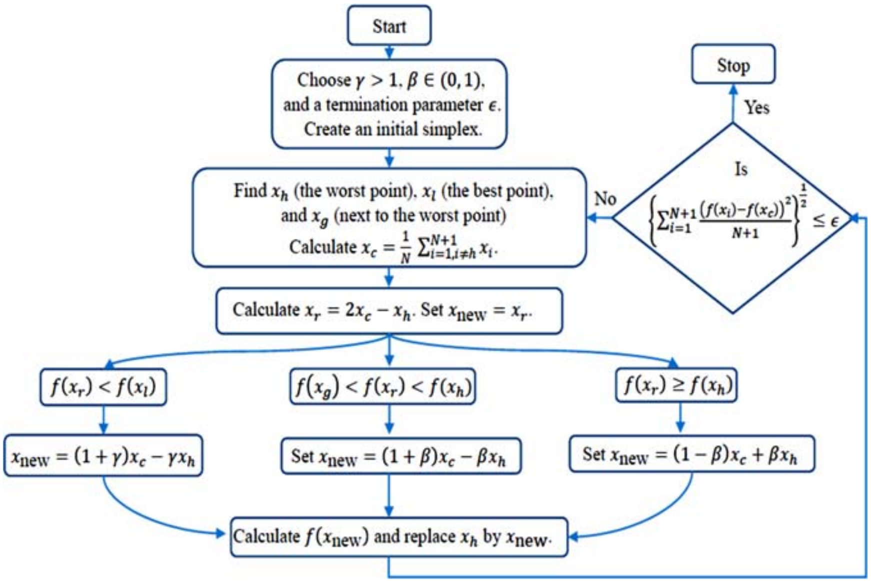

Start
Choose $\gamma > 1$, $\beta \in (0,1)$, and a termination parameter ϵ. Create an initial simplex.
Find x_h (the worst point), x_l (the best point), and x_g (next to the worst point) Calculate $x_c = \frac{1}{N}\sum_{i=1, i\neq h}^{N+1} x_i$.
Calculate $x_r = 2x_c - x_h$. Set $x_{new} = x_r$.
$f(x_r) < f(x_l)$
$f(x_g) < f(x_r) < f(x_h)$
$f(x_r) \ge f(x_h)$
$x_{new} = (1+\gamma)x_c - \gamma x_h$
Set $x_{new} = (1+\beta)x_c - \beta x_h$
Set $x_{new} = (1-\beta)x_c + \beta x_h$
Calculate $f(x_{new})$ and replace x_h by x_{new}.
Is $\left\{\sum_{i=1}^{N+1} \frac{(f(x_i)-f(x_c))^2}{N+1}\right\}^{\frac{1}{2}} \le \epsilon$
No
Yes
Stop

Figure 4.6 Flow diagram of the simplex search method.

Solution: Before applying the simplex search algorithm, we need to convert the nonlinear system of equations into the following unconstrained optimization problem.

$$\text{Minimize}\, f(x,y) = \left(x^2+y-11\right)^2 + \left(x+y^2-7\right)^2$$

First, we find the function values at $X^{(1)} = (0,0)^T$, $X^{(2)} = (2,0)^T$, and $X^{(3)} = (1,1)^T$. The function values are as follows:

$$f(X^{(1)}) = 170,\ f(X^{(2)}) = 74,\ \text{and}\, f(X^{(3)}) = 106.$$

As $f(X^{(1)}) > f(X^{(3)}) > f(X^{(2)})$, the point $X^{(1)}$ is named as worst point, $X_h = X^{(1)}$; $X_l = X^{(2)}$ is the best point, and next to the worst point is $X_g = X^{(3)}$.

The centroid will be calculated by considering X_l and X_g as follows:

$$X_c = \frac{X_l + X_g}{2} = (1.5, 0.5)^T.$$

We compute the reflected point as follows:

$$X_r = 2X_c - X_h = 2(1.5, 0.5)^T - (0,0)^T = (3,1)^T.$$

The corresponding function value at the reflected point is 10. Since $f(X_r) < f(X_l)$, we expand the simples to get a new point:

$$X_{\text{new}} = (1+\gamma)X_c - \gamma X_h = (1+1.5)(1.5, 0.5)^T - 1.5(0,0)^T = (3.75, 1.25)^T.$$

Function value at X_{new} is 21.44. Thus, the new simplex is $X^{(1)} = (3.75, 1.25)^T$, $X^{(2)} = (2,0)^T$, and $X^{(3)} = (1,1)^T$. Here, it is noted that the function value at a reflected point is better than the function value at the new point, but the simplex search algorithm does not allow this point in the new simplex. Now, compute the quantity M. We obtain $M = 44.86 > \epsilon$. So, go to Step 2. This completes the first iteration. Similarly, all other iterations are performed until the optimality condition $(M < \epsilon)$ is reached.

Three iterations are presented in Table 4.1.

Example 4.5. By using the simplex search method, perform three iterations and minimize $f(X) = x_1^2 + x_2^2 - 16x_1 - 12x_2$. Let expansion factor

Table 4.1 Three iterative solutions for Example 4.4.

Iterations	**1**	**2**	**3**
X_h	$(0,0)^T$	$(1,1)^T$	$(2,0)^T$
X_l	$(2,0)^T$	$(3.75,1.25)^T$	$(3.75,1.25)^T$
X_g	$(1,1)^T$	$(2,0)^T$	$(1.937,0.812)^T$
$f(X_h)$	170	106	74
$f(X_l)$	74	21.44	21.44
$f(X_g)$	106	74	60.772
X_r	$(3,1)^T$	$(4.75,0.25)^T$	$(3.687,2.062)$
$f(X_r)$	10	144.32	22.5594
Condition	$f(X_r)<f(X_l)$	$f(X_r)<f(X_h)$	$f(X_r)<f\left(X_g\right)$
X_{new}	$(3.75,1.25)^T$	$(1.937,0.812)^T$	$(3.26525,1.5165)^T$
$f(X_{\text{new}})$	21.44	60.772	3.2640

Table 4.2 Three iterative solutions for Example 4.5.

Iterations	**1**	**2**	**3**
X_h	$(2,3)^T$	$(4,4)^T$	$(6,3)^T$
X_l	$(6,3)^T$	$(9.5,4.25)^T$	$(9.5,4.25)^T$
X_g	$(4,4)^T$	$(6,3)^T$	$(9.625,3.4375)^T$
$f(X_h)$	-55	-80	-87
$f(X_l)$	-87	-94.6875	-94.6875
$f(X_g)$	-80	-87	-90.7930
X_r	$(8,4)^T$	$(11.5,3.25)^T$	$(13.125,4.6875)^T$
$f(X_r)$	-96	-80.1875	22.5594
Condition	$f(X_r)<f(X_l)$	$f(X_r)<f(X_h)$	$f(X_r)>f(X_h)$
X_{new}	$(9.5,4.25)^T$	$(9.625,3.4375)^T$	$(7.7813,3.4219)^T$
$f(X_{\text{new}})$	-94.6875	-90.7930	-93.3056

$\gamma = 1.5$ and contraction factor $\beta = 0.5$. Consider initial simplex as $X_1 = (6,3)^T$, $X_2 = (4,4)^T$, and $X_3 = (2,3)^T$.

Solution: Starting with the initial simplex if we apply the simplex algorithm, then we get the following solution included in Table 4.2.

A general MATLAB code for Algorithm 4.2 is given.

MATLAB code

```
function funval = fv(x)
x1 = x(1);
x2 = x(2);
fval = ((x1.^2+ x2 -11).^2 + (x1 + x2.^2-7).^2);
end

clear all
N= input ('Max number of iterations is ');
dt  = input('Enter the size reduction parameter delta = ');
tol = input('Enter the value of Tolerance = ');
p0= input ('The first point of the simplex');
p1= input ('The second point of the simplex');
p2= input ('The first point of the simplex');
p=[p0(1,:); p1(1,:); p1(1,:)];
iter=0;
while iter<N  && dt>tol
    V=[fv(p(1,1),p(1,2)); fv(p(2,1),p(2,2)); fv(p(3,1),p(3,2))];
    [V_ord, ind]=sort(V);
    V_ot=V(ind(3));
    p_ord=p(ind,:);
    p_new=[p_ord(3,1)+2*((p_ord(1,1)+p_ord(2,1))/2-p_ord(3,1))
p_ord(3,2)+2*((p_ord(1,2)+p_ord(2,2))/2-p_ord(3,2))];
    if fv(p_new(1,1),p_new(1,2))<=V_ot
        p(ind(3),:)=p_new(1,:);
    else
        V_ot2=V(ind(2));
        p_new=[p_ord(2,1)+2*((p_ord(1,1)+p_ord(3,1))/2-p_ord(2,1))
p_ord(2,2)+2*((p_ord(1,2)+p_ord(3,2))/2-p_ord(2,2))];
        if fv(p_new(1,1),p_new(1,2))<=V_ot2
            p(ind(2),:)=p_new(1,:);
        else
            p=[p_ord(1,1) p_ord(1,2); (p_ord(2,1)-p_ord(1,1))/2+p_ord(1,1)
(p_ord(2,2)-p_ord(1,2))/2+p_ord(1,2); (p_ord(3,1)-p_ord(1,1))/2+p_ord(1,1)
(p_ord(3,2)-p_ord(1,2))/2+p_ord(1,2)];
            delta=delta/2;
        end
    end
    ValFunc=fv(p(1,1),p(1,2));
    var_min=[p(1,1), p(1,2)];
    iter=iter+1;
end
```

4.5 Hooke–Jeeves pattern search method

The fundamental idea of the pattern search method is to construct a set of search directions iteratively. It is noted that the search directions should completely span the search space. That is, starting from any point in the search space, any other point in the search space can be reached by traveling through these search directions only. If we have a N-dimensional problem, then we require at least N linearly independent search directions. For example, in a three-variable function optimization problem, at least three search directions are needed to reach any other point from any point. The Hooke–Jeeves method drives the current point to a great extent with better approximation than the previous methods. The steering of a current point depends on two different moves, which we have to perform one after another in each iteration. These moves are exploratory and pattern moves. An exploratory move is performed in the vicinity of the current point systematically to find the best point around the current point. Thereafter, two such points are used to make a pattern move.

4.5.1 Exploratory move

Consider the current solution (the starting point) is denoted by x^c and the variable x_i^c is perturbed by $\Delta = (\Delta_1, \Delta_2, \cdots, \Delta_N)$. Set $i = 1$ and $x = x^c$.

Algorithm 4.3.

```
*Step 1*: Calculate $f = f(x), f^+ = f(x_i + \Delta_i)$, and $f^- = f(x_i - \Delta_i)$,
    $i = 1, 2, \cdots, N$.
*Step 2*: Find $f_{min} = \min\left(f, f^+, f^-\right)$. Set $x$ corresponds to $f_{min}$.
*Step 3*: If $i = N$, then set the value $x$ and go to Step 4. Else, set
    $i = i + 1$ and go to Step 1.
*Step 4*: If $x \neq x^c$, then success. Else, failure.
```

We can summarize Algorithm 4.3 (exploratory move) in the following way for the sake of better understanding. The current (or starting) point will be explored in positive and negative directions along with each variable one at a time with perturbation parameter Δ_i and the best point is calculated. Now, the current point is changed to the best point at the end of the search in each variable with perturbation. Then, we consider the exploratory move as success if the point found at the end of all variable perturbations is different than the original point (starting point of exploratory move), otherwise it is a failure.

4.5.2 Pattern move

After performing the exploratory move, we need to find a new point by jumping from the current best point x^c along a direction connecting the previous best point $x^{(k-1)}$ and the current base point $x^{(k)}$ is represented as follows:

$$x_p^{(k+1)} = 2x^{(k)} - x^{(k-1)}. \tag{4.7}$$

The Hooke–Jeeves method is the combination of an exploratory move in the vicinity of the current point and a subsequent jump using the pattern move iteratively. If the pattern move does not take the solution to a better region, the pattern move is not accepted, and the extent of the exploratory search is reduced. Considering both the exploratory and the pattern move, this algorithm works as follows.

Algorithm 4.4.

Step 1: Take a starting point $x^{(0)}$, increment $\Delta = (\Delta_1, \Delta_2, \cdots, \Delta_N)$, a reduction factor $\alpha > 1$, and a tolerance value, ϵ. Set $k = 0$.

Step 2: Consider the base point $x^{(k)}$ and perform an exploratory move. Say, the outcome of the exploratory move is x. If the exploratory move is a success, set $x^{(k+1)} = x$ and then go to Step 4. Else go to Step 3.

Step 3: If $\|\Delta\| < \epsilon$, then stop. Else set $\Delta_i = \Delta_i/\alpha$ for $i = 1, 2, \cdots, N$ and then go to Step 2.

Step 4: Set $k = k + 1$ and perform the following pattern move

$$x_p^{(k+1)} = 2x^{(k)} - x^{(k-1)}.$$

Step 5: Consider $x_p^{(k+1)}$ as the best point and perform another exploratory move. Set the result of this exploratory move is $x^{(k+1)}$.

Step 6: If $f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$, then go to Step 4. Else go to Step 3.

The search technique of this method is simply straight forward. This algorithm stores only two points $x^{(k)}$ and $x^{(k+1)}$ at any iteration, which requires less storage capacity. As it is a derivative-free method, it reduces the computational efforts. As the search greatly relies on the moves along with the directions of the coordinate $(x_1, x_2, x_3, \cdots, x_N)$ during the exploratory move, the algorithm may lead to a wrong solution prematurely. This may occur in the case of functions with highly nonlinear interactions among variables. Other difficulties may occur when the algorithm gets stuck in the loop either between Steps 2 and 3 or between Steps 5 and 6. Finally, the convergence to the optimum point mainly depends on the parameter α, and usually the value $\alpha = 2$ is recommended.

Example 4.6. Solve the following nonlinear system of equations in the interval $0 \le x, y \le 5$, by using the Hooke–Jeeves pattern search algorithm. Take the initial point $X^{(0)} = (0, 0)^T$, increment vector $\Delta = (0.5, 0.5)^T$, reduction factor $\alpha = 2$, and a tolerance value $\epsilon = 10^{-3}$.

$$x^2 + y = 11$$
$$x + y^2 = 7$$

Solution: Before applying the Hooke–Jeeves pattern search algorithm, we need to convert the nonlinear system of equations into the following unconstrained optimization problem.

$$\text{Minimize} f(x, y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right)^2$$

First, we explore the neighborhood of x. Then, we compute the function values at the following three points:

$$\left(x^{(0)} - \Delta_1, y^{(0)}\right)^T = (-0.5, 0)^T,\ (x^{(0)}, y^{(0)})^T = (0, 0)^T,$$
$$\text{and } (x^{(0)} + \Delta_1, y^{(0)})^T = (0.5, 0)^T$$

The corresponding function values are $f^- = f\left((-0.5, 0)^T\right) = 171.81$, $f = f\left((0, 0)^T\right) = 170$, and $f^+ = f\left((0.5, 0)^T\right) = 157.81$.

The minimum of f^-, f, and f^+ is $f_{min} = 157.81$, and the corresponding point is $(0.5, 0)^T$.

We explore the vicinity of y from $(0.5, 0)^T$. Then, we compute the function values at the following three points:

$$\left(x^{(1)}, y^{(0)} - \Delta_2\right)^T = (0.5, -0.5)^T,\ \left(x^{(1)}, y^{(0)}\right)^T = (0.5, 0)^T,$$
$$\text{and } \left(x^{(1)}, y^{(0)} + \Delta_2\right)^T = (0.5, 0.5)^T$$

The corresponding function values are $f^- = f\left((0.5, -0.5)^T\right) = 165.62$, $f = f\left((0.5, 0)^T\right) = 157.81$, and $f^+ = f\left((0.5, 0.5)^T\right) = 144.12$.

The minimum of f^-, f, and f^+ is $f_{min} = 144.12$, and the corresponding point is $(0.5, 0.5)^T$. At this step, the exploration move finished, and the new point is $(0.5, 0.5)^T$. Since $X \neq X^c$, the exploratory move is a success.

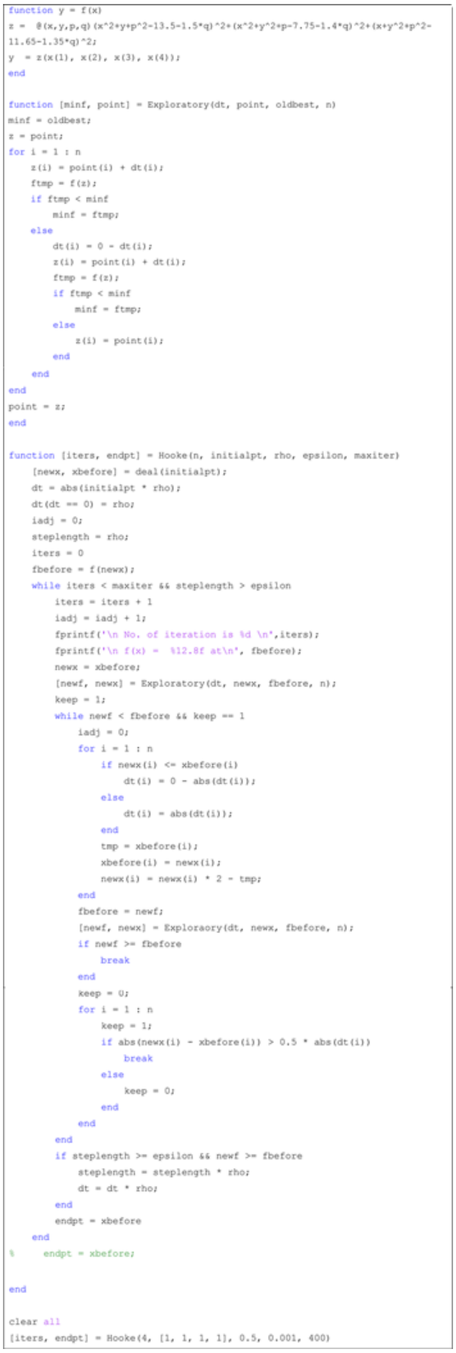
We set $X^{(1)} = (0.5, 0.5)^T$ and perform pattern move. Set $k = 1$ and compute $X_p^{(2)} = 2X^{(1)} - X^{(0)} = 2(0.5, 0.5)^T - (0, 0)^T = (1, 1)^T$.

Then, similar way we perform another exploratory move from $(1, 1)^T$ and obtain the new point $(1.5, 1.5)^T$. We set the new point $X^{(2)} = (1.5, 1.5)^T$. Calculate $f\left(X^{(1)}\right) = 144.12$ and $f\left(X^{(2)}\right) = 63.12$. This completes the first iteration.

Second iteration:

Set $k = 2$ and create a new point $X_p^{(3)} = 2X^{(2)} - X^{(1)} = 2(1.5, 1.5)^T - (0.5, 0.5)^T = (2.5, 2.5)^T$. Then perform an exploratory move from $(2.5, 2.5)^T$. After completion, we get the new point $X^{(3)} = (3, 2)^T$. At this point, the function value is exactly the minimum that is zero. Hence, accidentally we get the exact solution. Otherwise, we may proceed for the next iteration and perform all the iteration until the optimality condition $\|\Delta\| < \epsilon$ is satisfied. A general MATLAB code for the Hooke–Jeeves pattern search method is mentioned here.

MATLAB code



```
function y = f(x)
z =  @(x,y,p,q)(x^2+y+p^2-13.5-1.5*q)^2+(x^2+y^2+p-7.75-1.4*q)^2+(x+y^2+p^2-
11.65-1.35*q)^2;
y  = z(x(1), x(2), x(3), x(4));
end

function [minf, point] = Exploratory(dt, point, oldbest, n)
minf = oldbest;
z = point;
for i = 1 : n
    z(i) = point(i) + dt(i);
    ftmp = f(z);
    if ftmp < minf
        minf = ftmp;
    else
        dt(i) = 0 - dt(i);
        z(i) = point(i) + dt(i);
        ftmp = f(z);
        if ftmp < minf
            minf = ftmp;
        else
            z(i) = point(i);
        end
    end
end
point = z;
end

function [iters, endpt] = Hooke(n, initialpt, rho, epsilon, maxiter)
    [newx, xbefore] = deal(initialpt);
    dt = abs(initialpt * rho);
    dt(dt == 0) = rho;
    iadj = 0;
    steplength = rho;
    iters = 0
    fbefore = f(newx);
    while iters < maxiter && steplength > epsilon
        iters = iters + 1
        iadj = iadj + 1;
        fprintf('\n No. of iteration is %d \n',iters);
        fprintf('\n f(x) =  %12.8f at\n', fbefore);
        newx = xbefore;
        [newf, newx] = Exploratory(dt, newx, fbefore, n);
        keep = 1;
        while newf < fbefore && keep == 1
            iadj = 0;
            for i = 1 : n
                if newx(i) <= xbefore(i)
                    dt(i) = 0 - abs(dt(i));
                else
                    dt(i) = abs(dt(i));
                end
                tmp = xbefore(i);
                xbefore(i) = newx(i);
                newx(i) = newx(i) * 2 - tmp;
            end
            fbefore = newf;
            [newf, newx] = Exploraory(dt, newx, fbefore, n);
            if newf >= fbefore
                break
            end
            keep = 0;
            for i = 1 : n
                keep = 1;
                if abs(newx(i) - xbefore(i)) > 0.5 * abs(dt(i))
                    break
                else
                    keep = 0;
                end
            end
        end
        if steplength >= epsilon && newf >= fbefore
            steplength = steplength * rho;
            dt = dt * rho;
        end
        endpt = xbefore
    end
%     endpt = xbefore;

end

clear all
[iters, endpt] = Hooke(4, [1, 1, 1, 1], 0.5, 0.001, 400)
```

4.6 Conjugate direction method

The conjugate direction method is very powerful and widely used direct search technique in various engineering and scientific optimization problems. This method uses previously recorded solutions to construct new search directions. Also, this method possesses a convergence proof for quadratic objective functions. The fundamental idea of this method is to create a set of N linearly independent search directions and perform a series of unidirectional searches along each of these search directions. It needs to start each time from the previous best point. This procedure guarantees to find the minimum of a quadratic function by one pass of N unidirectional searches along each search direction. The algorithm of this method is made based on solving a quadratic function, and it has the following property.

4.6.1 Parallel subspace property

Given a quadratic function $q(x) = a + b^T x + \frac{1}{2} x^T C x$ of two variables (where a is a scalar quantity, b is a vector, and C is a 2×2 matrix), two arbitrary but distinct points $x^{(1)}$ and $x^{(2)}$, and a direction S. If $y^{(1)}$ is the solution to the problem

$$\text{Minimize } q\left(x^{(1)} + \lambda S\right) \tag{4.8}$$

and $y^{(2)}$ is the solution to the problem

$$\text{minimize } q\left(x^{(2)} + \lambda S\right) \tag{4.9}$$

then the direction $\left(y^{(2)} - y^{(1)}\right)$ is conjugate to S or, in other words, the quantity $\left(y^{(2)} - y^{(1)}\right)^T CS$ is zero.

Thus, if two arbitrary points $x^{(1)}$ and $x^{(2)}$ and an arbitrary search direction S are chosen, two unidirectional searches, one from each point will create two points $y^{(1)}$ and $y^{(2)}$. For quadratic functions, we can say that the minimum of the function lies on the line joining the points $y^{(1)}$ and $y^{(2)}$. The vector $\left(y^{(2)} - y^{(1)}\right)$ forms a conjugate direction with the original direction vector S.

Instead of using two points $x^{(1)}$ and $x^{(2)}$ and a direction vector S to create one pair of conjugate directions, one point $x^{(1)}$ and both coordinate directions $(1, 0)^T$ and $(0, 1)^T$ can be used to create a pair of conjugate directions d and $\left(y^{(2)} - y^{(1)}\right)$. The point $y^{(1)}$ is obtained by performing a unidirectional search along $(1, 0)^T$ from the point $x^{(1)}$. Then, the point $x^{(2)}$ is obtained by performing a unidirectional search along $(0, 1)^T$ from

$y^{(1)}$, and finally, the point $y^{(2)}$ is found by a unidirectional search along the direction $(1, 0)^T$ from the point $x^{(2)}$.

For a quadratic polynomial, the minimum lies in the direction $\left(y^{(2)} - y^{(1)}\right)$, but for higher-order polynomials, the true minimum may not lie in the aforementioned direction. Thus, in the case of a quadratic polynomial, four unidirectional searches will find the minimum point and in higher-order polynomials, more than four unidirectional searches may be necessary. This concept of parallel subspace property can also be extended to higher dimensions.

4.6.2 Extended parallel subspace property

Assume that the point $y^{(1)}$ is found after unidirectional searches along each of $m(<N)$ conjugate directions from a chosen point $x^{(1)}$ and, similarly, the point $y^{(2)}$ is found after unidirectional searches along each of m conjugate directions from another point $x^{(2)}$. The vector $\left(y^{(2)} - y^{(1)}\right)$ is the conjugate to all m search directions.

The extended parallel subspace property can also be used starting from one point $x^{(1)}$ and N coordinate directions. The point $y^{(1)}$ can be found after a unidirectional search along with one of the search directions (say $e^{(1)} = (1, 0, \cdots, 0)^T$). Thereafter, the point $y^{(2)}$ is found after searches in N coordinate directions ($e^{(i)}, i = 2, 3, \cdots, N, 1$) with $e^{(1)}$ being the final search direction. Then, the vector $\left(y^{(2)} - y^{(1)}\right)$ and the same procedure can be followed starting from $e^{(2)}$. With this property, we now present the algorithm.

Algorithm 4.5.

Step 1: Choose an initial point $x^{(0)}$ and a set of N linearly independent directions. Set $s^{(i)} = e^{(i)}$ for $i = 1, 2, \cdots, N$.

Step 2: Using unidirectional search technique, minimize the previous minimum point along N directions to begin the next search. Begin with the search direction $s^{(1)}$ and end with $s^{(N)}$. Thereafter, perform another unidirectional search along $s^{(1)}$.

Step 3: Form a new conjugate direction d using the extended parallel subspace property.

Step 4: If $\|d\|$ is small or search directions are linearly dependent, then terminate the process. Else replace $s^{(j)} = s^{(j-1)}\ \forall\ j = N, N-1, \cdots, 2$. Set $s^{(1)} = \frac{d}{\|d\|}$ and go to Step 2.

If the function is quadratic, exactly $(N-1)$ loops through Steps 2 to 4 are required. Since in every iteration of Algorithm 4.5 exactly $(N+1)$

unidirectional searches are necessary, a total of $(N-1)\times(N+1)$ or (N^2-1) unidirectional searches are necessary to find N conjugate directions. Thereafter, one final unidirectional search is necessary to obtain the minimum point. Thus, to find the minimum of a quadratic objective function, the conjugate direction method requires a total of N^2 unidirectional searches. For other functions, more loops of Algorithm 4.5 may be required. One difficulty with this algorithm is that since unidirectional searches are carried out numerically by using the single-variable search method, the computation of the minimum for unidirectional searches may not be exact. Thus, the resulting directions may not be exactly conjugate to each other. To calculate the extent of the deviation, the linear independence of the conjugate directions is usually checked. If the search directions are not found to be linearly independent, a completely new set of search directions (possibly conjugate to each other) may be created at the current point. To make the implementation simpler, the coordinate directions can be used again as search directions at the current point.

4.7 Steepest descent method

The steepest descent method is one of the gradient search methods where gradients are used to steer the search. A search direction $s^{(k)}$ at point $x^{(k)}$ is called a descent direction if the condition $\nabla f\left(x^{(k)}\right)\cdot s^{(k)}\leq 0$ is satisfied in the neighborhood of the point $x^{(k)}$. This condition can be obtained by the comparison of function values at two points along any descent direction that is $f\left(x^{(k+1)}\right)<f\left(x^{(k)}\right)$ and expanding the expression $f\left(x^{(k+1)}\right)$ in Taylor's series up to the following linear term:

$$f\left(x^{(k+1)}\right)=f\left(x^{(k)}+\alpha s^{(k)}\right)=f\left(x^{(k)}\right)+\alpha\nabla f\left(x^{(k)}\right)\cdot s^{(k)} \tag{4.10}$$

The magnitude of the vector $\nabla f\left(x^{(k)}\right)\cdot s^{(k)}$ for descent direction $s^{(k)}$ determines the extent of descent. For example, if $s_2^{(k)}=-\nabla f\left(x^{(k)}\right)$, the quantity $\nabla f\left(x^{(k)}\right).s^{(k)}$ is maximally negative. Thus, the direction $-\nabla f\left(x^{(k)}\right)$ is called the steepest descent direction.

Example 4.7. Consider the following function

$$\text{Minimize} f(x_1,x_2)=\left(x_1^2+x_2-11\right)^2+\left(x_1+x_2^2-7\right)^2$$

First, we check whether the direction $s^{(t)}=(1,0)^T$ at the point $x^{(t)}=(1,1)^T$ is a descent. It can be easily seen from the function value that moving locally along $s^{(t)}$ from the point $x^{(t)}$ will reduce the function value. We obtain this aspect by computing the derivative $\nabla f\left(x^{(t)}\right)$ at that point. The derivative at that point is $\nabla f\left(x^{(t)}\right) = (-46, -38)^T$. Considering the dot product between $\nabla f\left(x^{(t)}\right)$ and $s^{(t)}$, we obtain

$$\nabla f\left(x^{(t)}\right)^T s^{(t)} = (-46, -38)\begin{pmatrix} 1 \\ 0 \end{pmatrix} = -46,$$

which is a negative quantity. Hence, the search direction $s^{(t)} = (1, 0)^T$ is a descent direction. Here, the value 46 tells the extent of descent in the direction. If the search direction $s^{(t)} = -\nabla f\left(x^{(t)}\right) = (46, 38)^T$ is used, the magnitude of this dot product becomes

$$(-46, -38)^T \begin{pmatrix} 46 \\ 38 \end{pmatrix} = -3560.$$

Thus, the direction $(46, 38)^T$ is the steepest descent direction at the point $x^{(t)}$. It is noteworthy that for any nonlinear function, the steepest descent direction at any point may not exactly pass through the true minimum point. The steepest descent direction is a direction, which is a local best direction. It is not guaranteed that moving along the steepest descent direction will always take the search closer to the true minimum point.

Most gradient-based methods work by searching along with several directions iteratively. The algorithms vary according to the way the search directions are defined. Along with each direction, $s^{(k)}$, a unidirectional search, is defined. Along each direction, $s^{(t)}$ as follows:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} s^{(k)}, \tag{4.11}$$

where $\alpha^{(k)}$ is the step length. Since $x^{(k)}$ and $s^{(k)}$ are known, the point $x^{(k+1)}$ can be expressed with only one variable $\alpha^{(k)}$. Thus, a unidirectional search on $\alpha^{(k)}$ can be performed, and the new point $x^{(k+1)}$ can be obtained. Thereafter, the search is continued from the new point along another search direction $s^{(k+1)}$. This process continues until the search converges to a local minimum point. If a gradient-based search method is used for unidirectional search, the search can be terminated using the following procedure. By differentiating the expression $f\left(x^{(k+1)}\right) = f\left(x^{(k)} + \alpha s^{(k)}\right)$ with respect to α and satisfying the optimality criterion, it can be shown that the minimum of the unidirectional search occurs when

$$\nabla f\left(x^{(k+1)}\right) \cdot s^{(k)} = 0. \tag{4.12}$$

The aforementioned criterion can be used to check the termination of the unidirectional search method.

4.7.1 Cauchy's (steepest descent) method

The search direction used in Cauchy's method is the negative of the gradient at any particular point $x^{(t)}$.

$$s^{(k)} = -\nabla f\left(x^{(k)}\right). \tag{4.13}$$

Since this direction gives maximum descent in function values, it is also known as the steepest descent method. At every iteration, the derivative is computed at the current point, and a unidirectional search is performed in the negative to this derivative direction to find the minimum point along that direction. The minimum point becomes the current point, and the search is continued from this point. The algorithm continues until a point having a small enough gradient vector is found. This algorithm guarantees improvement in the function value at every iteration.

Algorithm 4.6.

Step 1: Choose a maximum number of iterations M to be performed, an initial point $x^{(0)}$, two termination parameters ϵ_1, ϵ_2, and set $k = 0$.
Step 2: Calculate $\nabla f\left(x^{(k)}\right)$, the first derivative at the point $x^{(k)}$.
Step 3: If $\left\|\nabla f\left(x^{(k)}\right)\right\| \leq \epsilon_1$, terminate; Else if $k \geq M$; terminate; Else go to Step 4.
Step 4: Perform a unidirectional search to find $\alpha^{(k)}$ using ϵ_2 such that $f\left(x^{(k+1)}\right) = f\left(x^{(k)} - \alpha^{(k)}\nabla f\left(x^{(k)}\right)\right)$ is minimum. One criterion for termination is when $\left|\nabla f\left(x^{(k+1)}\right) \cdot \nabla f\left(x^{(k)}\right)\right| \leq \epsilon_2$.
Step 5: Is $\frac{\left\|x^{(k+1)} - x^{(k)}\right\|}{\left\|x^{(k)}\right\|} \leq \epsilon_1$? If yes, terminate; Else set $k = k + 1$ and go to Step 2.

Since the direction $s^{(k)} = -\nabla f\left(x^{(k)}\right)$ is a descent direction, the function value $f\left(x^{(k+1)}\right)$ is always smaller than $f\left(x^{(k)}\right)$ for positive values of $\alpha^{(k)}$. Cauchy's method works well when $x^{(0)}$ is far away from x^*. When the current point is very close to the minimum, the change in the gradient vector is small. Thus, the new point created by the unidirectional search is also close to the current point. This slows the convergence process near the true minimum. Convergence can be made faster by using second-order derivatives.

Example 4.8. Minimize $f(X) = x_1^2 + x_2^2 - 11x_1 - 9x_2$ starting from $X_1 = (6, 5)^T$ by the steepest descent method.

Solution: Here, the function is $f(X) = x_1^2 + x_2^2 - 11x_1 - 9x_2$. Partial derivatives of $f(X)$ are as follows:

$$\frac{\partial f(X)}{\partial x_1} = 2x_1 - 11 \text{ and } \frac{\partial f(X)}{\partial x_2} = 2x_2 - 9.$$

Therefore, $\nabla f = \begin{Bmatrix} 2x_1 - 11 \\ 2x_2 - 9 \end{Bmatrix}$. We know the initial vector $X^{(0)} = \begin{Bmatrix} 6 \\ 5 \end{Bmatrix}$.

Hence, $\nabla f\left(X^{(0)}\right) = \begin{Bmatrix} 2 \times 6 - 11 \\ 2 \times 5 - 9 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$.

The steepest direction $s^{(0)}$ is $-\nabla f\left(X^{(0)}\right) = \begin{Bmatrix} -1 \\ -1 \end{Bmatrix}$. The new vector $X^{(1)}$ is

$$X^{(0)} + \alpha s^{(0)} = \begin{Bmatrix} 6 \\ 5 \end{Bmatrix} + \alpha \begin{Bmatrix} -1 \\ -1 \end{Bmatrix} = \begin{Bmatrix} 6 - \alpha \\ 5 - \alpha \end{Bmatrix}.$$

Now,
$f(X) = (6-\alpha)^2 + (5-\alpha)^2 - 11(6-\alpha) - 9(5-\alpha) = 2\alpha^2 - 2\alpha - 50.$

The necessary condition tells,

$$\frac{df}{d\alpha} = 0 \Rightarrow 4\alpha - 2 = 0 \Rightarrow \alpha = 0.5.$$

So, $X^{(1)} = \begin{Bmatrix} 6 - 0.5 \\ 5 - 0.5 \end{Bmatrix} = \begin{Bmatrix} 5.5 \\ 4.5 \end{Bmatrix}$.

This completes the first iteration.

Next, $\nabla f\left(X^{(1)}\right) = \begin{Bmatrix} 2 \times 5.5 - 11 \\ 2 \times 4.5 - 9 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$. As $\nabla f\left(X^{(1)}\right)$ is a zero vector, we have to terminate the process and the optimum vector X is $(5.5, 4.5)^T$. The corresponding optimum function value is $f = 5.5^2 + 4.5^2 - 11 \times 5.5 - 9 \times 4.5 = 50.5$.

4.8 Newton's method

Newton's method uses second-order derivatives to create search directions. This allows faster convergence to the minimum point. Considering the first three terms in Taylor's series expansion of a multivariable function, it can be shown that the first-order optimality condition will be satisfied if a search direction

$$s^{(k)} = -\left[\nabla^2 f\left(x^{(k)}\right)\right]^{-1} \nabla f\left(x^{(k)}\right) \tag{4.14}$$

is used. It can also be shown that if the matrix $\left[\nabla^2 f\left(x^{(k)}\right)\right]^{-1}$ is positive semidefinite, the direction $s^{(k)}$ must be descent. But if the matrix $\left[\nabla^2 f\left(x^{(k)}\right)\right]^{-1}$ is not positive definite, the direction $s^{(k)}$ may or may not be descent, depending on whether the quantity $\nabla f\left(x^{(k)}\right)^T \left[\nabla^2 f\left(x^{(k)}\right)\right]^{-1} \nabla f\left(x^{(k)}\right)$ is positive. Thus, the aforementioned search direction may not always guarantee a decrease in the function value in the vicinity of the current point. But the second-order optimality condition suggests that $\nabla^2 f(x^*)$ be positive definite for the minimum point. Thus, it can be assumed that the matrix $\nabla^2 f(x^*)$ is positive definite in the vicinity of the minimum point, and the aforementioned search direction becomes descent near the minimum point. Thus, this method is suitable and efficient when the initial point is close to the optimum point. Since the function value is not guaranteed to reduce at every iteration, the occasional restart of the algorithm from a different point is often necessary.

Algorithm 4.7.

The algorithm is the same as Cauchy's method except Step 4 is modified as follows. *Step 4*: Perform a unidirectional search to find $\alpha^{(k)}$ using ϵ_2, such that $f\left(x^{(k+1)}\right) = f\left(x^{(k)} - \alpha^{(k)}\left[\nabla^2 f\left(x^{(k)}\right)\right]^{-1} \nabla f\left(x^{(k)}\right)\right)$ is minimum.

4.9 Marquardt's method

Cauchy's method works well when the initial point is far away from the minimum point, and Newton's method works well when the initial point is near the minimum point. In any given problem, it is usually not known whether the chosen initial point is away from the minimum or close to the minimum, but wherever be the minimum point, a method can be devised to take advantage of both these methods. In Marquardt's method, Cauchy's method is initially followed. Thereafter, Newton's method is adopted. The transition from Cauchy's method to Newton's method is adaptive and depends on the history of the obtained intermediate solutions, as outlined in Algorithm 4.8.

Algorithm 4.8.

```
Step 1: Choose a starting point $x^{(0)}$, the maximum number of iterations $M$, and a termination parameter $\epsilon$. Set $k = 0$ and $\lambda^{(0)} = 10^4$ (a large number).
Step 2: Calculate $\nabla f\left(x^{(k)}\right)$.
Step 3: If $\left\|\nabla f\left(x^{(k)}\right)\right\| \leq \epsilon$ or $k \geq M$? Terminate; Else go to Step 4.
Step 4: Calculate $s\left(x^{(k)}\right) = -\left[H^{(k)} + \lambda^{(k)} I\right]^{-1} \nabla f\left(x^{(k)}\right)$. Set $x^{(k+1)} = x^{(k)} + s\left(x^{(k)}\right)$.
Step 5: Is $f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$? If yes, go to Step 6; Else go to Step 7.
Step 6: Set $\lambda^{(k+1)} = \frac{1}{2}\lambda^{(k)}$, $k = k + 1$, and go to Step 2.
Step 7: Set $\lambda^{(k)} = 2\lambda^{(k)}$ and go to Step 4.
```

A large value of the parameter λ is used initially. Thus, the Hessian matrix has little effect on the determination of the search direction (see Step 4 of Algorithm 4.8). Initially, the search is similar to that in Cauchy's method. After several iterations (when hopefully the current solution has converged close to the minimum), the value of λ becomes small, and the effect is more like that in Newton's method. The algorithm can be made faster by performing a unidirectional search while finding the new point Step 4 $x^{(k+1)} = x^{(k)} + \alpha^{(k)} s\left(x^{(k)}\right)$. Since the computations of the Hessian matrix and its inverse are computationally expensive, the unidirectional search along $s^{(k)}$ is not usually performed. For simpler objective functions, however, a unidirectional search in Step 4 can be achieved to find the new point $x^{(k+1)}$.

Practice set

1. Identify the stationary points and the nature of the following functions.
   a. $f(x_1, x_2) = x_1^2 + 2x_2^2$
   b. $f(x, y) = (x-1)^2 + 10(y-2)^2$

c. $f(y_1, y_2) = y_1^2 + 2y_2^2 - 4y_1 - 2y_1y_2$
d. $f(x,y) = \left(x^2+y-11\right)^2 + \left(x+y^2-7\right)^2$
e. $f(x,y,z) = -x^2 + 4xy - 9y^2 + 2xz + 8yz - 4z^2$

2. Find the extreme points of the function $f(x_1, x_2) = x_1^3 + x_2^3 + 2x_1^2 + 4x_2^2 + 6$.

3. State the necessary and sufficient conditions for the extreme of a multivariable function with no constraints.

4. Determine whether the following matrices are positive definite, negative definite or indefinite

a. $M = \begin{bmatrix} 4 & -3 & 0 \\ -3 & 0 & 4 \\ 0 & 4 & 2 \end{bmatrix}$

b. $M = \begin{bmatrix} -14 & 3 & 0 \\ 3 & -1 & 4 \\ 0 & 4 & 2 \end{bmatrix}$

c. $M = \begin{bmatrix} 5 & 2 & 2 \\ 2 & 5 & 2 \\ 2 & 2 & 5 \end{bmatrix}$

5. Bracket the interval in which the minimum of the function $f(x,y) = (x-10)^2 + (y-10)^2$ lies, while it is considered along the direction $(2,5)^T$ starting from the current point $(2,1)^T$ using the bounding phase method with $\Delta = (0.5, 0.5)^T$ and improve your search by using three iterations of the golden section search algorithm.

6. Use Box's evolutionary optimization method, perform three iterations to find the minimum for the function $f(x,y) = x^3 - 2xy^2 + 4x + 10$ with size reduction parameter $\Delta = (1,1)^T$ and initial point $(1,1)^T$.

7. Minimize $f(x,y) = (x-1)^2 + 10(y-2)^2$, where $X = (x,y)$, evolutionary search algorithm. Choose $X^{(0)} = (1,1)^T$ and reduction parameter $\Delta = (2,2)^T$. Perform three iterations.

8. Perform two iterations of the simplex search method to minimize $f(x_1, x_2) = x_1^2 + x_2^2 - 16x_1 - 12x_2$, where the expansion factor $\gamma = 1.5$ and contraction factor $\beta = 0.5$. Take initial simplex as $(6,3), (4,4\big)$, and $(2,3)$.

9. In trying to minimize the function $f(x,y) = 10 - x + xy + y^2$, use $x^{(1)} = (0,2)^T, x^{(2)} = (0,0)^T$, and $x^{(3)} = (1,1)^T$ as the initial simplex of three points. Complete three iterations of the simplex search algorithm to find new simplex. Assume $\beta = 0.5$ and $\gamma = 2$.

10. Perform two iterations to minimize $f(x_1, x_2) = 3{x_1}^2 + {x_2}^2 - 12x_1 - 8x_2$ using the Hook and Jeeves exploratory pattern search method starting from $x^{(0)} = (1, 1)^T$ and perturbation vector $\Delta = (0.5, 0.5)^T$.

11. Perform three iterations of the Cauchy steepest descent method to minimize the function $f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$ starting from the point $X = \left(0/0\right)$.

12. Use Newton's method of multivariable optimization to minimize the function

$f(x_1, x_2) = x_1 - x_2 + 2x_1^2 + 2x_1x_2 + x_2^2$

by taking the starting point as $x^{(0)} = (0, 0)^T$.

13. Minimize $f(x_1, x_2) = 2x_1^2 + x_2^2$ by using the Cauchy steepest descent method with the starting point $X_1 = (1, 2)^T$.

14. Minimize $f(X) = 3x_1^2 + 3x_2^2 - 33x_1 - 27x_2$ starting from $X_1 = (6, 5)^T$ by the steepest descent method.

Further reading

Bazaraa, M., Sherali, H., & Shetty, C. (1993). *Nonlinear programming: Theory and algorithms*. New York, NY: Wiley.

Bhavikatti, S. S. (2010). *Fundamentals of optimum design in engineering*. New Delhi: New Age International (P) Limited Publishers.

Box, G., & Draper, N. (1969). *Evolutionary operation*. New York, NY: Wiley.

Deb, K. (2004). *Optimization for engineering design algorithms and examples*. New Delhi: Prentice Hall of India.

Gonzaga, C. (1992). Path following methods for linear programming. *SIAM Review*, 167–224.

Kolman, B., & Trench, W. (1971). *Elementary multivariable calculus*. New York, NY: Academic Press.

Nash, S., & Sofer, A. (1996). *Linear and nonlinear programming*. New York, NY: McGraw-Hill.

Nelder, J., & Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 308–313.

Powell, M. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 155–162.

Rao, S. S. (1995). *Optimization theory and applications*. New Delhi: New Age International (P) Limited Publishers.

Strang, G. (1980). *Linear algebra and its applications*. Orlando: Academic Press.

Thomas, G. (1967). *Calculus and analytic geometry*. Boston, MA: Addison-Wesley.

Wright, J. N. (2006). *Numerical optimization*. New York, NY: Springer.

CHAPTER FIVE

Multivariable constrained nonlinear optimization

5.1 Classical methods for equality constrained optimization

This section dedicates the classical method to investigate optimal solutions for equality constrained optimization problems. We consider the optimization of multivariable continuous function subjected to equality constraints. In the following, we present the general form of multivariable equality constrained optimization problem.

$$\text{Minimize}\, f = f(X)$$

subject to

$$g_j(X) = 0,\ j = 1,\ 2,\ \cdots,\ m \tag{5.1}$$

where $X = (x_1,\ x_2,\ \cdots,\ x_n)^T$.

Here, m represents the number of constraints and n is the number of independent variables; the value of m is less than equal to n. If $m > n$, the problem becomes over defined and, in general, there will be no solution. In this context, several exact methods are available to obtain a solution. These are method of direct substitution, constrained variation, and Lagrange multipliers, which are discussed in the following paragraphs.

5.1.1 Solution by direct substitution

Consider an optimization problem having n variables and m equality constraints. Then, theoretically it is possible to investigate simultaneously the m equality constraints and express any set of m variables in terms of the remaining $n - m$ variables. We convert the same to the unconstrained optimization problem by substituting these expressions into the original objective function. Hence, we get a new objective function involving only $n - m$ variables. As such, the new objective function is provided with no constraints, and the optimum of such problems can be obtained

© 2020 Elsevier Inc.
All rights reserved.

by using unconstrained optimization techniques. We consider the direct substitution method and solved an example problem.

Example 5.1. Using a direct substitution method, find the dimension of a box of largest volume that can be inscribed in a sphere of unit radius.

Solution: Take the origin of the Cartesian coordinate system $x_1x_2x_3$ be at the center of the sphere, and the sides of the box be $2x_1, 2x_2$, and $2x_3$. Then, the volume of the box can be written as follows:

$$f(x_1, x_2, x_3) = 2x_1 2x_2 2x_3 = 8x_1x_2x_3 \tag{5.2}$$

Since the box will be inscribed in a unit sphere, the corners of the box must lie on the surface of the sphere. Hence, the unit radius x_1, x_2, and x_3 should satisfy the constraint:

$$x_1^2 + x_2^2 + x_3^2 = 1 \tag{5.3}$$

The equality constrained optimization problem can be stated as follows:

$$\text{Maximize}\, f(x_1, x_2, x_3) = 8x_1x_2x_3$$

subjected to

$$x_1^2 + x_2^2 + x_3^2 = 1 \tag{5.4}$$

This optimization problem possess three design variables x_1, x_2, and x_3 and one equality constraint. So, the equality constraint (Eq. (5.3)) can be used to eliminate any one of the design variables from the objective function. We choose to eliminate x_3, and Eq. (5.3) gives

$$x_3 = \left(1 - x_1^2 - x_2^2\right)^{\frac{1}{2}} \tag{5.5}$$

Thus, the objective function becomes

$$f(x_1, x_2) = 8x_1x_2\left(1 - x_1^2 - x_2^2\right)^{\frac{1}{2}}. \tag{5.6}$$

Eq. (5.6) can be maximized by using an unconstrained optimization technique for an unconstrained function in two variables.

The necessary conditions for a maximum of f are as follows:

$$\frac{\partial f}{\partial x_1} = 8x_2\left[\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}} - \frac{x_1^2}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}}\right] = 0 \tag{5.7}$$

$$\frac{\partial f}{\partial x_2} = 8x_1\left[\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}} - \frac{x_2^2}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}}\right] = 0 \tag{5.8}$$

Eqs. (5.7) and (5.8) can be simplified to obtain

$$\begin{aligned} 1 - 2x_1^2 - x_2^2 &= 0 \\ 1 - x_1^2 - 2x_2^2 &= 0. \end{aligned} \tag{5.9}$$

From Eq. (5.9), it follows that the optimum variables are $x_1^* = x_2^* = 1/\sqrt{3}$, and hence, $x_3^* = \frac{1}{\sqrt{3}}$. This solution provides the maximum volume of the box as $f_{max} = \frac{8}{3\sqrt{3}}$.

The next task is to check whether the solution corresponds to a maximum or a minimum.

We apply the sufficiency conditions to $f(x_1, x_2)$ of Eq. (5.6). As such, we need the second-order partial derivatives at (x_1^*, x_2^*) and these are given by

$$\frac{\partial^2 f}{\partial x_1^2} = -\frac{8x_1x_2}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}} - \frac{8x_2}{1-x_1^2-x_2^2}\left[\frac{x_1^3}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}} + 2x_1\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}\right]$$
$$= -\frac{32}{\sqrt{3}}$$

$$\frac{\partial^2 f}{\partial x_2^2} = -\frac{8x_1x_2}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}} - \frac{8x_1}{1-x_1^2-x_2^2}\left[\frac{x_2^3}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}} + 2x_1\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}\right]$$
$$= -\frac{32}{\sqrt{3}}$$

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} = 8\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}} - \frac{8x_2^2}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}} - \frac{8x_1^2}{1-x_1^2-x_2^2}$$
$$\left[\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}} + 0\frac{x_2^2}{\left(1-x_1^2-x_2^2\right)^{\frac{1}{2}}}\right] = -\frac{16}{\sqrt{3}}$$

Since $\frac{\partial^2 f}{\partial x_1^2} < 0$ and $\frac{\partial^2 f}{\partial x_1^2}\frac{\partial^2 f}{\partial x_2^2} - \left(\frac{\partial^2 f}{\partial x_1 \partial x_2}\right)^2 > 0$, by using the principal minors of the Hessian matrix of f, we can conclude that the second-order partial derivative or Hessian matrix is negative definite at (x_1^*, x_2^*). Therefore, the point (x_1^*, x_2^*) corresponds to the maximum of f.

Here, we can conclude that this method appears to be simple theoretically, but it is not convenient practically. The main reason for inconvenience is that the constraint equations will be nonlinear for most practical problems, and often, it becomes very difficult to solve them.

5.1.2 Solution by the method of constrained variation

The basic idea used in the method of constrained variation is to find a closed-form expression for the first-order differential of f, that is, df, at all the points at which the constraints $g_j(X) = 0$, $j = 1, 2, \cdots, m$ are satisfied. The desired optimum points are then obtained by setting the differential df equal to zero.

The important features can be identified through the following simple problem with $n = 2$ (number of independent variables) and $m = 1$ (number of constraints).

$$\text{Minimize} f(x_1, x_2)$$

subject to

$$g(x_1, x_2) = 0$$

A necessary condition for $f(x_1, x_2)$ to have a minimum at some point (x_1^*, x_2^*) is that the total derivative of $f(x_1, x_2)$ with respect to x_1 must be zero at (x_1^*, x_2^*). By setting the total differential of $f(x_1, x_2)$ equal to zero, we obtain

$$df = \frac{\partial f}{\partial x_1} dx_1 + \frac{\partial f}{\partial x_2} dx_2 = 0 \tag{5.10}$$

Since $g\left(x_1^*, x_2^*\right) = 0$ at the minimum point, any variations dx_1 and dx_2 taken about the point (x_1^*, x_2^*) are called admissible variations provided that the new point lies on the constraint:

$$g\left(x_1^* + dx_1, x_2^* + dx_2\right) = 0. \tag{5.11}$$

Taylor's series expansion of the function in Eq. (5.11) about the point (x_1^*, x_2^*) gives

$$g\left(x_1^* + dx_1, x_2^* + dx_2\right) = g\left(x_1^*, x_2^*\right) + \frac{\partial g}{\partial x_1}\left(x_1^*, x_2^*\right)dx_1 + \frac{\partial g}{\partial x_2}\left(x_1^*, x_2^*\right)dx_2 = 0 \tag{5.12}$$

where dx_1 and dx_2 are assumed to be small. Since $g\left(x_1^*, x_2^*\right) = 0$, Eq. (5.12) reduces to

$$dg = \frac{\partial g}{\partial x_1}dx_1 + \frac{\partial g}{\partial x_2}dx_2 = 0 \text{ at } \left(x_1^*, x_2^*\right). \tag{5.13}$$

Thus, Eq. (5.13) has to be satisfied by all admissible variations. The same concept can be generalized for n number of independent variables with m number of constraints.

Example 5.2. Using a constrained optimization method, find the dimension of the largest section in flexure that can be cut from a circular log of radius r.

Solution: Consider the width of the section is $2x$, and the depth of the section is $2y$. Then, the optimization problem can be stated as follows:

$$\text{Maximize } f(X) = \frac{4}{3}xy^2$$

subject to

$$x^2 + y^2 = r^2.$$

We can represent the given constraint in the following way

$$g(X) = x^2 + y^2 - r^2 = 0.$$

Therefore,

$$\frac{\partial f}{\partial x} = \frac{4}{3}y^2, \quad \frac{\partial f}{\partial y} = \frac{8}{3}xy, \quad \frac{\partial g}{\partial x} = 2x, \text{ and } \frac{\partial g}{\partial y} = 2y.$$

The necessary condition for (x^*, y^*) to be optimum subject to the function $g(X) = 0$ is as follows:

$$\frac{\partial f}{\partial x}\frac{\partial g}{\partial y} - \frac{\partial f}{\partial y}\frac{\partial g}{\partial x} = 0 \Rightarrow \frac{8}{3}y^3 - \frac{16}{3}x^2y = 0.$$

Further solving the above relations, we get $y = \sqrt{2}x$. Substituting the value of y in the constraint $g(X) = 0$, we have $x^2 + 2x^2 - r^2 = 0 \Rightarrow x = r/\sqrt{3}$. Hence, $y = \left(\sqrt{2/3}\right)r$. So, the largest section is $f_{max} = (8/9)r^3$.

5.1.3 Solution by the method of Lagrange multipliers

The fundamental idea of this method is to convert the equality constrained optimization problems to unconstrained optimization problems using the Lagrange multiplier. Let us consider a two-variable optimization problem with equality constrained to understand this method.

Two-variable equality constrained problem can be stated as follows:

$$\text{Minimize}\, f(x_1, x_2) \tag{5.14}$$

subject to

$$g(x_1, x_2) = 0.$$

The necessary condition for the existence of an optimum point at $X = X^*$ is obtained to be

$$\left(\frac{\partial f}{\partial x_1} - \frac{\frac{\partial f}{\partial x_2}}{\frac{\partial g}{\partial x_2}}\frac{\partial g}{\partial x_1}\right)\Big|_{(x_1^*, x_2^*)} = 0. \tag{5.15}$$

For Eq. (5.15), we may define a quantity Lagrange multiplier λ, which is given as follows:

$$\lambda = -\left(\frac{\frac{\partial f}{\partial x_2}}{\frac{\partial g}{\partial x_2}}\right)\Big|_{(x_1^*, x_2^*)} \tag{5.16}$$

Eq. (5.15) can be rewritten as follows:

$$\left(\frac{\partial f}{\partial x_1} + \lambda\frac{\partial g}{\partial x_1}\right)\Big|_{(x_1^*, x_2^*)} = 0. \tag{5.17}$$

and Eq. (5.16) can be written as follows:

$$\left(\frac{\partial f}{\partial x_2} + \lambda\frac{\partial g}{\partial x_2}\right)\Big|_{(x_1^*, x_2^*)} = 0. \tag{5.18}$$

In addition, the equality constraints also satisfy the extreme point. So, we get

$$g(x_1, x_2)\Big|_{(x_1^*, x_2^*)} = 0. \tag{5.19}$$

Here, Eqs. (5.17)–(5.19) represent the necessary conditions for the point (x_1^*, x_2^*) to be an optimal point.

It is noted that the partial derivative $\frac{\partial g}{\partial x_2}\Big|_{(x_1^*,x_2^*)}$ should be nonzero to define λ value in Eq. (5.16). This is because of the variation dx_2 was expressed in terms of dx_1 in the derivation of Eq. (5.15). Another way, if we choose to express dx_1 in terms of dx_2, we would have obtained the requirement that $\frac{\partial g}{\partial x_1}\Big|_{(x_1^*,x_2^*)}$ be nonzero to define λ. Thus, the derivation of the necessary conditions by the method of Lagrange multipliers requires that at least one of the partial derivatives of $g(x_1, x_2)$ be nonzero at an extreme point.

The necessary conditions given by Eqs. (5.17)–(5.19) are more commonly generated by constructing a function L, known as the Lagrange function, as follows:

$$L(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda g(x_1, x_2) \tag{5.20}$$

By treating L as a function of the three variables x_1, x_2, and λ, the necessary conditions for its extreme are given by

$$\begin{aligned}
\frac{\partial L}{\partial x_1}(x_1, x_2, \lambda) &= \frac{\partial f}{\partial x_1}(x_1, x_2) + \lambda \frac{\partial g}{\partial x_1}(x_1, x_2) = 0 \\
\frac{\partial L}{\partial x_2}(x_1, x_2, \lambda) &= \frac{\partial f}{\partial x_2}(x_1, x_2) + \lambda \frac{\partial g}{\partial x_2}(x_1, x_2) = 0 \\
\frac{\partial L}{\partial \lambda}(x_1, x_2, \lambda) &= g(x_1, x_2) = 0
\end{aligned} \tag{5.21}$$

Eq. (5.21) can be seen to be the same as Eqs. (5.17)–(5.20).

In the following, the extension of this method to a general problem of n variables with m constraints is given.

5.1.3.1 Necessary conditions

Consider the case of a general optimization problem with n variables and m equality constraints

$$\text{Minimize}\, f(X) \tag{5.22}$$

subject to

$$g_j(X) = 0, j = 1, 2, \cdots, m.$$

In this case, the Lagrange function L can be defined by introducing Lagrange multiplier λ_j for each constraint $g_j(X)$ as follows:

$$L(x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_m) = f(X) + \lambda_1 g_1(X) + \lambda_2 g_2(X) + \cdots + \lambda_m g_m(X). \tag{5.23}$$

Hence, L becomes a function of $n+m$ unknowns, and the unknowns are $x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_m$. The necessary conditions for the optimum of function L, correspond to the solution of the original problem stated in Eq. (5.22), are given by

$$\frac{\partial L}{\partial x_i} = \frac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i} = 0, i = 1, 2, \cdots, n \tag{5.24}$$

$$\frac{\partial L}{\partial \lambda_j} = g_j(X) = 0, j = 1, 2, \cdots, m. \tag{5.25}$$

Eqs. (5.24) and (5.25) represent $n+m$ equations in terms of the $n+m$ unknowns. The solution of Eqs. (5.24) and (5.25) gives

$$X^* = \begin{Bmatrix} x_1^* \\ x_2^* \\ \vdots \\ x_n^* \end{Bmatrix} \text{ and } \lambda^* = \begin{Bmatrix} \lambda_1^* \\ \lambda_2^* \\ \vdots \\ \lambda_m^* \end{Bmatrix}.$$

The vector X^* corresponds to the relative constrained minimum of $f(X)$ (sufficient conditions are to be verified), while the vector λ^* provides the sensitivity information.

5.1.3.2 Sufficient condition

A sufficient condition for $f(X)$ to have a constrained relative minimum at X^* is given by the following theorem.

Theorem: A sufficient condition for $f(X)$ to have a relative minimum at X^* is that the quadratic, Q, defined by

$$Q = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 L}{\partial x_i \partial x_j} dx_i dx_j \tag{5.26}$$

evaluated at $X = X^*$, must be positive definite for all values of dX for which the constraints are satisfied.

From this theorem, the following points can be noted.

1. If $Q = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 L}{\partial x_i \partial x_j}(X^*, \lambda^*) dx_i dx_j$ is negative for all choices of the admissible variations dx_i, X^* will be a constrained maximum of $f(X)$.
2. It has been noted that a necessary condition for the quadratic form Q, defined by Eq. (5.26) to be positive (negative) definite for all admissible variations dX is that each root of the polynomial z_i, defined by the following determinant equation, be positive (negative).

$$\begin{bmatrix} L_{11}-z & L_{12} & \cdots & L_{1n} & g_{11} & g_{2n} & \cdots & g_{m1} \\ L_{21} & L_{22}-z & \cdots & L_{2n} & g_{12} & g_{22} & \cdots & g_{m2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ L_{n1} & L_{n2} & \cdots & L_{nn}-z & g_{1n} & g_{2n} & \cdots & g_{mn} \\ g_{11} & g_{12} & \cdots & g_{1n} & 0 & 0 & 0 & 0 \\ g_{21} & g_{22} & \cdots & g_{2n} & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ g_{m1} & g_{m2} & \cdots & g_{mn} & 0 & 0 & 0 & 0 \end{bmatrix} = 0 \qquad (5.27)$$

where

$$L_{ij} = \frac{\partial^2 L}{\partial x_i \partial x_j}(X^*, \lambda^*) g_{ij} = \frac{\partial g_i}{\partial x_j}(X^*)$$

Eq. (5.27) on expansion leads to an $(n-m)$th order polynomial in z. If some of the roots of this polynomial are positive while the others are negative, the point X^* is not an optimum point.

5.2 Classical methods for inequality constrained optimization

This section deals with the study of multivariable optimization problems with inequality constraints. The basic idea of finding the optimum solution to these problems is to convert inequality constrained problems to equality constrained, and then it can be converted into an unconstrained optimization problem. The same can be solved in an unconstrained optimization technique. In the following, we will see the general statement of the problem and methodology to investigate the optimality.

The multivariable optimization problem with inequality constraints can be stated as follows:

$$\text{Minimize } f(X) \tag{5.28}$$

subject to

$$g_j(X) \le 0, \; j = 1,\; 2,\; \cdots,\; m$$

The inequality constraints in Eq. (5.28) can be transformed into equality constraints by adding nonnegative slack variables, y_j^2, as follows:

$$g_j(X) + y_j^2 = 0, \; j = 1,\; 2,\; \cdots,\; m \tag{5.29}$$

where the values of the slack variables are unknown.

The problem (Eq. (5.28)) becomes

$$\text{Minimize } f(X)$$

subject to

$$G_j(X, Y) = g_j(X) + y_j^2 = 0, \; j = 1,\; 2,\; \cdots,\; m \tag{5.30}$$

where $Y = (y_1, y_2, \cdots, y_m)^T$ is the vector of slack variables $y_1, y_2, \cdots, y_m$.

This problem can be solved conveniently by the method of Lagrange multipliers. Construct the Lagrange function L as follows:

$$L(X, Y, \lambda) = f(X) + \sum_{j=1}^{m} \lambda_i G_j(X, Y) \tag{5.31}$$

where $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m)^T$ is the vector of Lagrange multipliers $\lambda_1, \lambda_2, \cdots,$ λ_m. The stationary points of the Lagrange function can be found by solving the following equations using necessary conditions for Eq. (5.31).

$$\frac{\partial L}{\partial x_i}(X, Y, \lambda) = \frac{\partial f}{\partial x_i}(X) + \sum_{j=1}^{m} \lambda_j \frac{\partial g_j}{\partial x_i}(X) = 0, \; i = 1,\; 2,\; \cdots,\; n \tag{5.32}$$

$$\frac{\partial L}{\partial \lambda_j}(X, Y, \lambda) = G_j(X, Y) = g_j(X) + y_j^2 = 0, \; j = 1,\; 2,\; \cdots,\; m \tag{5.33}$$

$$\frac{\partial L}{\partial y_j}(X, Y, \lambda) = 2\lambda_j y_j = 0, \; j = 1,\; 2,\; \cdots,\; m \tag{5.34}$$

It can be seen that Eqs. (5.32)–(5.34) represent $(n + 2m)$ equations in the $(n + 2m)$ unknowns, X, λ, and Y. The solution of Eqs. (5.32)–(5.34) thus gives the optimum solution vector X^*, the Lagrange multiplier vector, λ^*, and the slack variable vector, Y^*.

Eq. (5.33) ensures that the constraints $g_j(X) \le 0, j = 1, 2, \cdots, m$, are satisfied, while Eq. (5.31) implies that either $\lambda_j = 0$ or $y_j = 0$. It means that the jth constant is inactive and hence can be ignored. On the other hand, if $y_j = 0$, it means that the constraint is active $(g_j = 0)$ at the optimum point. Consider the division of the constraints into two subsets, J_1 and J_2, where $J_1 + J_2$ represent the optimum point and J_2 include the indices of all the inactive constraints.

Thus, for $j \in J_1$, $y_j = 0$ (constraints are active), for $j \in J_2$, $\lambda_j = 0$ (constraints are inactive), and Eq. (5.31) can be simplified as follows:

$$\frac{\partial f}{\partial x_i} + \sum_{j \in J_1} \lambda_j \frac{\partial g_j}{\partial x_i} = 0, i = 1, 2, \cdots, n \tag{5.35}$$

Similarly, Eq. (5.33) can be written as follows:

$$g_j(X) = 0, j \in J_1 \tag{5.36}$$

$$g_j(X) + y_j^2 = 0, j \in J_2 \tag{5.37}$$

Eqs. (5.35)–(5.37) represent $n + p + (m - p) = n + m$ equations in the $n + m$ unknowns $x_i (i = 1, 2, \cdots, n)$, $\lambda_j (j \in J_1)$, $\lambda_j (j \in J_2)$, where p denotes the number of active constraints.

Assuming that the first p constraints are active, Eq. (5.35) can be expressed as follows:

$$-\frac{\partial f}{\partial x_i} = \lambda_1 \frac{\partial g_1}{\partial x_i} + \lambda_2 \frac{\partial g_2}{\partial x_i} + \cdots + \lambda_p \frac{\partial g_p}{\partial x_i}, i = 1, 2, \cdots, n. \tag{5.38}$$

These equations can be written collectively as follows:

$$-\nabla f = \lambda_1 \nabla g_1 + \lambda_2 \nabla g_2 + \cdots + \lambda_p \nabla g_p \tag{5.39}$$

where ∇f and ∇g_j are the gradients of the objective function and the jth constraint, respectively

$$\nabla f = \begin{Bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{Bmatrix} \text{ and } \nabla g_j = \begin{Bmatrix} \dfrac{\partial g}{\partial x_1} \\ \dfrac{\partial g}{\partial x_2} \\ \vdots \\ \dfrac{\partial g}{\partial x_n} \end{Bmatrix} \tag{5.40}$$

Eq. (5.39) indicates that the negative of the gradient of the objective function can be expressed as a linear combination of the gradients of the active constraints at the optimum point.

Furthermore, it can be shown that in the case of a minimization problem, the λ_j values ($j \in J_1$) have to be positive. For easy illustration, suppose that only two constraints are active ($p = 2$) at the optimum point. Then, Eq. (5.39) reduces to

$$-\nabla f = \lambda_1 \nabla g_1 + \lambda_2 \nabla g_2 \tag{5.41}$$

Kuhn–Tucker condition

$$\frac{\partial f}{\partial x_i} + \sum_{j \in J_1} \lambda_j \frac{\partial g_j}{\partial x_i} = 0, i = 1, 2, \cdots, n \tag{5.42}$$

$$\lambda_j > 0, j \in J_1 \tag{5.43}$$

These are called the Kuhn–Tucker conditions after the mathematicians who derived them as the necessary conditions to be satisfied at a relative minimum of $f(X)$. These conditions are, in general, not sufficient to ensure a relative minimum. However, there is a class of problems, called convex programming problems, for which the Kuhn–Tucker conditions are necessary and sufficient for a global minimum.

If the set of active constraints is not known, the Kuhn–Tucker conditions can be stated as follows:

$$\frac{\partial f}{\partial x_i} + \sum_{j \in J_1} \lambda_j \frac{\partial g_j}{\partial x_i} = 0, i = 1, 2, \cdots, n$$

$$\lambda_j g_i = 0, j = 1, 2, \cdots, m \tag{5.44}$$

$$g_j \le 0, j = 1, 2, \cdots, m$$

$$\lambda_j \ge 0, j = 1, 2, \cdots, m$$

Note that if the problem is one of maximization or if the constraints are of the type $g_j \ge 0$, the λ_j have to be nonpositive in Eq. (5.44). On the other hand, if the problem is one of maximization with constraints in the form $g_j \ge 0$, the λ_j have to be nonnegative Eq. (5.44).

5.3 Random search method

Random generation of numbers and minor modifications in unconstrained constraints minimization may be useful to obtain the optimal solution. This idea is used to develop random search methods to solve a constrained optimization problem. The procedure for this method can be outlined in the following steps.

1. Using a random number generates a trial design vector for each design variable.
2. Verify whether the constraints are satisfied at the trial design vector. Within a particular tolerance, often, the equality constraints are satisfied. If any constraint is violated, continue generating new trial vectors until a trial vector that satisfies all the constraints is found.
3. If all the constraints are satisfied, then keep the current trial vector as the best design if it gives a reduced objective function value compared to the previous best available design. Otherwise, discard the currently feasible trial vector and proceed to Step 1 to generate a new trial design vector.
4. The best design available at the end of generating a specified maximum number of trial design vectors is taken as the solution to the constrained optimization problem.

It can be seen that several modifications can be made to the basic procedure indicated earlier. For example, after finding a feasible trial design vector, a feasible direction can be generated (using random numbers) and a one-dimensional search can be conducted along the feasible direction to find an improved feasible design vector.

Another procedure involves constructing an unconstrained function, $F(X)$, by adding a penalty for violating any constrained as follows:

$$F(X) = f(X) + a\sum_{j=1}^{m}\left[G_j(X)\right]^2 + b\sum_{k=1}^{p}[H_k(X)]^2 \tag{5.45}$$

where

$$\left[G_j(X)\right]^2 = \left[\max\left(0, g_j(X)\right)\right]^2 \tag{5.46}$$

$$[H_k(X)]^2 = h_k^2(X) \tag{5.47}$$

indicate the squares of violations of inequality and equality constraints, respectively, where a and b are the constants. Eq. (5.45) demonstrates that while minimizing the objective function $f(X)$, a positive penalty is added whenever a constraint is violated. The penalty is proportional to the square of the amount of violation. The values of the constants a and b can be adjusted to change the contributions of the penalty terms relative to the magnitude of the objective function. It can be summarized that the random search methods are not efficient compared to the other methods described in this chapter. However, they are very simple to program and usually are reliable in finding a nearly optimal solution with a sufficiently large number of trial vectors. Also, this method is useful to find near global optimal solution even if the nonconvex feasible region is provided.

5.4 Complex method

The complex method uses the concepts of the simplex method of unconstrained minimization to solve constrained minimization problems. The problem statement can be stated as follows:

$$\text{Minimize } f(X) \tag{5.48}$$

subject to

$$g_j(X) \leq 0, \ j = 1, \ 2, \ \cdots, \ m \tag{5.49}$$

$$x_i^{(l)} \leq x_i \leq x_i^{(u)}, \ i = 1, \ 2, \ \cdots, \ n \tag{5.50}$$

Generally, the satisfaction of the side constraints (lower and upper bounds on the variables x_i) may not correspond to the satisfaction of the constraints $g_j(X) \leq 0$. As such, this method may not manage nonlinear equality constraints. The formation of a sequence of geometric figures (simplex) each having $k = n + 1$ vertices in an n-dimensional space is the basic idea in the simplex method. Similarly, in the complex method, a sequence of geometric figures each having $k \geq n + 1$ vertices is formed to find the constrained minimum point. The method assumes that an initial feasible point X_1 (which satisfies all the m constraints) is available.

5.4.1 Iterative procedure

1. Obtain $k \geq n+1$ points, each of which satisfies all m constraints. In actual practice, we start with only one feasible point X_1, and the remaining $k-1$ points are found one at a time by the use of random numbers generated in the range 0 and 1, as follows:

$$x_{i,j} = x_i^{(l)} + r_{i,j}\left(x_i^{(u)} - x_i^{(l)}\right), i = 1, 2, \cdots, n, j = 1, 2, \cdots, m \qquad (5.51)$$

where $x_{i,j}$ is the ith component of the X_j and $r_{i,j}$ is a random number lying in the interval $(0, 1)$. It can be pointed out that the points $X_2, X_3, \cdots, X_k$ generated according to Eq. (5.51) satisfy the side constraints, Eq. (5.50), but may not satisfy the constraints given by Eq. (5.49).

As soon as a new point X_j is generated $(j = 2, 3, \cdots, k)$, we find whether it satisfies all the constraints, Eq. (5.49). If X_j violates any of the constraints stated in Eq. (5.49), the trial point X_j is moved halfway toward the centroid of the remaining. The centroid X_c of accepted points is given by

$$X_c = \frac{1}{j-1}\sum_{l=1}^{j-1} X_l. \qquad (5.52)$$

If the trial point X_j so found still violates some of the constraints, Eq. (5.49), the process of moving halfway in toward the centroid X_c is continued until a feasible point X_j is found. Ultimately, we will be able to find a feasible point X_j by this procedure provided that the feasible region is convex. By proceeding in this way, we will ultimately be able to find the required feasible points $X_2, X_3, \cdots, X_k$.

2. The objective function is evaluated at each of the k points (vertices). If the vertex X_h corresponds to the largest function value, the process of reflection is used to find a new point X_r as follows:

$$X_r = (1+\alpha)X_c - \alpha X_h \qquad (5.53)$$

where $\alpha \geq 1$ (to start with), and X_c is the centroid of all vertices except X_h

$$X_c = \frac{1}{k-1}\sum_{\substack{l=1 \\ l \neq k}}^{k} X_l \qquad (5.54)$$

3. Since the problem is a constrained one, the point X_r has to be tested for feasibility. If the point X_r is feasible and $f(X_r) < f(X_h)$, the point X_h is replaced by X_r, and go to Step 2. If $f(X_r) \geq f(X_h)$, a new trial point X_r is found by reducing the value of α in Eq. (5.49) by a factor of 2 and is tested for the satisfaction of the relation $f(X_r) < f(X_h)$. If $f(X_r) \geq f(X_h)$, the procedure of finding a new point X_r with a reduced value of α is repeated. This procedure is repeated, if necessary, until the value of α becomes smaller than a prescribed small quantity ϵ, say 10^{-6}. If an improved point X_r, with $f(X_r) < f(X_h)$, cannot be obtained even with that small value of α, the point X_r is discarded and the entire procedure of reflection is restarted by using the point X_p (which has the second-highest function value) instead of X_h.

4. If at any stage, the reflected point X_r (found in Step 3) violates any of the constraints Eq. (5.49), it is moved halfway in toward the centroid until it becomes feasible, that is,

$$(X_r)_{\text{new}} = \frac{1}{2}(X_c + X_r). \tag{5.55}$$

This method will progress toward the optimum point as long as the complex has not collapsed into its centroid.

5. Each time the worst point X_h of the current complex is replaced by a new point, the complex gets modified, and we have to test the convergence of the process. We assume convergence of the process whenever the following two conditions are satisfied:

a. The complex shrinks to a specified small size (i.e., the distance between any two vertices among $X_1, X_2, \cdots, X_k$ becomes smaller than a prescribed small quantity, ϵ_1).

b. The standard deviation of the function value becomes sufficiently small (i.e., when

$$\left\{\frac{1}{k}\sum_{j=1}^{k}\left[f(X)-f\left(X_j\right)\right]^2\right\}^{\frac{1}{2}} \leq \epsilon_2 \tag{5.56}$$

where X is the centroid of all the k vertices of the current complex and $\epsilon_2 > 0$ is a specified small number).

Note:

This method does not require the derivatives of $f(X)$ and $g_j(X)$ to find the minimum point, and hence, it is computationally very simple. The method is very simple from a programming point of view and does not require large computer storage.

1. A value of 1.3 for the initial value of α in Eq. (5.53) is satisfactory by Box.
2. Box recommended value of $k \approx 2n$ (although a lesser value can be chosen if n is greater than, say, 5). If k is not sufficiently large, the complex tends to collapse and flatten along the first constraint boundary encountered.
3. From the aforementioned procedure, it can be observed that the complex rolls over and over, normally expanding. However, if a boundary is encountered, the complex contracts and flattens itself. It can then roll along this constraint boundary and leave it if the contours change. The complex can also accommodate more than one boundary and can turn corners.
4. If the feasible region is nonconvex, there is no guarantee that the centroid of all feasible points is also feasible. If the centroid is not feasible, we cannot apply the aforementioned procedure to find the new points X_r.
5. The method becomes inefficient rapidly as the number of variables increases.
6. It cannot be used to solve problems having equality constraints.
7. This method requires an initial point X_1 that is feasible.

5.5 Sequential linear programming

In the sequential linear programming (SLP) method, the solution of the original nonlinear programming (LP) problem is obtained by solving a series of LP problems. Each LP problem is generated by approximating the nonlinear objective and constraint functions using first-order Taylor series expansions about the current design vector, X_i. The resulting LP problem is solved by using the simplex method to find the new design vector X_{i+1}. If X_{i+1} does not satisfy the stated convergence criteria, the problem is relinearized about the point X_{i+1}, and the procedure is continued until the optimum solution X^* is found. If the problem is a convex programming problem, the linearized constraints always lie entirely outside the feasible region. Therefore, the optimum solution of the approximating LP problem that lies at a vertex of the new feasible region will lie outside the original feasible region. However, by relinearizing the problem about the new point and repeating the process, one can achieve convergence to the solution of the original problem in a few iterations. The SLP method is also known as the cutting plane method.

Algorithm 5.1.

Step 1: Take an initial point X_1 such that it need not be feasible. Set the iteration number as $i = 1$.
Step 2: Using Taylor's series, linearize the objective function and constraints about the point X_i as follows:
$f(X) \approx f(X_i) + \nabla f(X_i)^T (X - X_i)$
$g_j(X) \approx g_j(X_i) + \nabla g_j(X_i)^T (X - X_i)$
$h_k(X) \approx h_k(X_i) + \nabla h_k(X_i)^T (X - X_i)$.
Step 3: Using linearized objective function and constraints, construct the approximating linear programming problem as follows:
Minimize $f(X_i) + \nabla f_i^T (X - X_i)$
subject to
$g_j(X_i) + \nabla g_j(X_i)^T (X - X_i) \leq 0, j = 1, 2, \cdots, m,$
$h_k(X_i) + \nabla h_k(X_i)^T (X - X_i) = 0, k = 1, 2, \cdots, p.$
Step 4: Solve the approximating linear programming problem to find the solution vector X_{i+1}.
Step 5: Evaluate the original constraints at X_{i+1}; that is, find $g_j(X_{i+1}), j = 1, 2, \cdots, m$ and $h_k(X_{i+1}), k = 1, 2, \cdots, p$.
If $g_j(X_{i+1}) \leq \epsilon$ for $j = 1, 2, \cdots, m$, and $|h_k(X_{i+1})| \leq \epsilon, k = 1, 2, \cdots, p$, where ϵ is prescribed small positive tolerance, all the original constraints can be assumed to have been satisfied. Hence, stop the procedure by taking $X_{\text{opt}} \approx X_{i+1}$.
If $g_j(X_{i+1}) > \epsilon$ for some j, or $|h_k(X_{i+1})| > \epsilon$ for some k, find the most violated constraint, for example, as follows:
$g_k(X_{i+1}) = \max_j [g_j(X_{i+1})]$.
Relinearize the constraint $g_k(X) \leq 0$ about the point X_{i+1} as follows:
$g_k(X) \approx g_k(X_{i+1}) + \nabla g_k(X_{i+1})^T (X - X_{i+1}) \leq 0,$
Add this as the $(m + 1)$th inequality constraint to the previous linear programming problem.
Step 6: Set the new iteration number as $i = i + 1$.
The total number of constraints in the new approximating linear programming problem as $m + 1$ inequalities and p equalities. Go to Step 4.

The SLP method has the following advantages.

1. This technique is quite efficient for solving convex programming problems with nearly linear objective and constraint functions.
2. Each of the approximating problems will be a LP problem, and hence, it can be solved efficiently. Moreover, any two consecutive approximating LP problems differ by only one constraint, and hence, the dual simplex method can be used to solve the sequence of approximating LP problems much more efficiently.
3. This method can be easily extended to solve integer programming problems. In this case, one integer LP problem has to be investigated in each stage.

5.6 Zoutendijk's method of feasible directions

In this method, we choose a starting point satisfying all the constraints and move to a better point according to the iterative scheme:

$$X_{i+1} = X_i + \lambda S_i. \tag{5.57}$$

where X_i is the starting point for the ith iteration, S_i is the direction of movement, λ is the distance of movement (step length), and X_{i+1} is the final point obtained at the end of the ith iteration.

The value of λ is always chosen in such a way that X_{i+1} lies in the feasible region. The search direction S_i is found such that

1. a small move in that direction violates no constraint, and
2. the value of the objective function can be reduced in that direction.

The new point X_{i+1} is taken as the starting point for the next iteration, and the entire procedure is repeated until a point is obtained that has no direction satisfying both properties 1 and 2 can be found. In general, such a point denotes the constrained local minimum of the problem. This local minimum need not be a global one unless the problem is a convex programming problem. A direction satisfying property 1 is called feasible, while a direction satisfying both properties 1 and 2 is called a usable feasible direction. There are many ways of choosing usable feasible directions, and hence, there are many different methods of feasible directions. As seen earlier, a direction S is feasible at a point X_i if it satisfies the relation:

$$\left(\frac{d}{d\lambda} g_j(X_i + \lambda S)\right)\Big|_{\lambda=0} = S^T \nabla g_j(X_i) \le 0 \tag{5.58}$$

where the equality sign holds only if a constraint is linear or strictly concave.

A vector S will be a usable feasible direction if it satisfies the relations:

$$\frac{d}{d\lambda} f(X_i + \lambda S)\Big|_{\lambda=0} = S^T \nabla f(X_i) < 0, \tag{5.59}$$

$$\frac{d}{d\lambda} g_j(X_i + \lambda S)\Big|_{\lambda=0} = S^T \nabla g_j(X_i) \le 0. \tag{5.60}$$

It is possible to reduce the value of the objective function at least by a small amount by taking a step length $\lambda > 0$ along such a direction.

The detailed iterative procedure of the methods of feasible directions will be considered in terms of two well-known methods: Zoutendijk's method of feasible directions and Rosen's gradient projection method.

In the feasible direction method, a linear approximation of the objective function and constraints are made to determine locally good search directions. The desired search direction at any point should be such that the points generated locally along that directions require two different considerations: feasibility and descent properties of non-LP problems. The condition for descent direction d to be descent if $\nabla f\left(x^{(t)}\right).d \le 0$. The condition for feasibility can be derived from similar considerations on active constraints. By including only the first-order term in Taylor's series expansion of an active constraint $g_{\bar{j}}(x)$ at any point $x^{(t)}$ ($\bar{j}$ represents all active constraints at the point $x^{(t)}$).

Algorithm 5.2. begins with a random point. At this point, the set of active constraints is found. If none of the constraints are active, the current point is an intermediate point in the search space, and the steepest descent search direction is used. But if one or more search directions are active, the current point is on at least one constraint boundary. Thus, any arbitrary search direction (or the steepest descent direction) may not find a feasible point. Thus, a search direction that is maximally feasible, and descent is found by solving an artificial LP problem. Once a search direction is found, a unidirectional search is performed along that direction to find the minimum point. This completes one iteration of Zoutendijk's feasible direction method.

Algorithm 5.2.

```
Step 1: Set an iteration counter $t = 0$. Choose an initial feasible point $x^{(0)}$ and
a parameter for checking the constraint violation $\epsilon$.
Step 2: At the current point $x^{(t)}$, let $I^{(t)}$ be the set of indices of active
constraints, which can be represented as follows:
$I^{(t)} = \{j : 0 \leq g_j(x^{(t)}) \leq \epsilon, j = 1, 2, \cdots, J\}$.
If $I^{(t)}$ is empty, use $d^{(t)} = \theta^{(t)} = -\nabla f(x^{(t)})$, normalize $d^{(t)}$ and go to Step 4.
Step 3: Solve the following linear programming problem
Maximize $\theta$
subject to
$\nabla f(x^{(t)}) d \leq -\theta$
$\nabla g_j(x^{(t)}) d \geq \theta, j \in I^{(t)}$;
$-1 \leq d_i \leq 1, i = 1, 2, \cdots, N$.
Label the solution $d^{(t)}$ and $\theta^{(t)}$.
Step 4: If $\theta^{(t)} \leq 0$, then stop the process.
Else $\overline{\alpha} = \min[\alpha : g_j(x^{(t)} + \alpha d^{(t)}) = 0, j = 1, 2, \cdots, J; \alpha > 0]$.
If no $\overline{\alpha} > 0$ exists, set $\overline{\alpha} = \infty$.
Step 5: Find $\alpha^{(t)}$ such that
$f(x^{(t)} + \alpha^{(t)} d^{(t)}) = \min[f(x^{(t)} + \alpha d^{(t)}) : 0 \leq \alpha \leq \alpha]$.
Set $x^{(t+1)} = x^{(t)} + \alpha^{(t)} d^{(t)}$, $t = t + 1$ and go to Step 2.
```

If none of the intermediate points is a boundary point, this method is similar to the steepest descent method (Cauchy's method), except that a modification is needed to take care of the constraint violations. However, in Zoutendijk's method, only a subset of constraints is used to define a subproblem. Thus, the LP problem is smaller than other linearization methods. One difficulty with this method is that since only active constraints are used to determine the search directions, the algorithm may result in zigzag iteration patterns, thereby making the convergence slower.

5.7 Sequential quadratic programming

Sequential quadratic programming is one of the newly developed methods that has a wide range of application and perhaps one of the best methods of optimization. The method has a theoretical basis that is related to

1. the solution of a set of nonlinear equations using Newton's method, and

2. the derivation of simultaneous nonlinear equations using the Kuhn–Tucker conditions to the Lagrangian of the constrained optimization problem.

In this section, the derivation of both the equations and the solution procedure of the sequential quadratic programming approach is discussed.

5.7.1 Derivation

Consider a nonlinear optimization problem with only equality constraints as follows:

$$\text{Minimize } f(X)$$

subject to

$$h_k(X) = 0, k = 1, 2, \cdots, p \tag{5.61}$$

The extension to include inequality constraints will be considered at a later stage. The Lagrange function, $L(X, \lambda)$, corresponding to the problem of Eq. (5.61) is given by

$$L = f(X) + \sum_{k=1}^{p} \lambda_k h_k(x) \tag{5.62}$$

where λ_k is the Lagrange multiplier for the kth equality constraint.

The Kuhn–Tucker necessary conditions can be stated as follows:

$$\nabla L = 0 \text{ or } \nabla f + \sum_{k=1}^{p} \lambda_k \nabla h_k = 0 \text{ or } \nabla f + [A]^T \lambda = 0, \tag{5.63}$$

$$h_k(X) = 0, \ k = 1, \ 2, \ \cdots, \ p. \tag{5.64}$$

Here, $[A]$ is an $n \times p$ matrix whose kth column denotes the gradient of the function h_k. Eqs. (5.63) and (5.64) represent a set of $n + p$ nonlinear equations in $n + p$ unknowns ($x_i, i = 1, 2, \cdots, n$ and $\lambda_k, k = 1, 2, \cdots, p$). These nonlinear equations can be solved using Newton's method. For convenience, we rewrite Eqs. (5.63) and (5.64) as follows:

$$F(Y) = 0, \tag{5.65}$$

where

$$F = \begin{Bmatrix} \nabla L \\ h \end{Bmatrix}_{(n+p)\times 1}, \ Y = \begin{Bmatrix} X \\ \lambda \end{Bmatrix}_{(n+p)\times 1}, \ 0 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}_{(n+p)\times 1} \tag{5.66}$$

According to Newton's method, the solution of Eq. (5.65) can be found iteratively as follows:

$$Y_{j+1} = Y_j + \Delta Y_j \tag{5.67}$$

with

$$[\nabla F]_j^T \Delta Y_j = -F(Y_j) \tag{5.68}$$

where Y_j is the current solution of the jth iteration and ΔY_j is the change in Y_j necessary to generate the improved solution, Y_{j+1}, and $[\nabla F]_j = [\nabla F(Y_j)]$ is the $(n+p)\times(n+p)$ Jacobian matrix of the nonlinear equations whose ith column denotes the gradient of the function $F_i(Y)$ with respect to the vector Y. By substituting Eqs. (5.65) and (5.66) into Eq. (5.68), we obtain

$$\begin{bmatrix} [\nabla^2 L] & [H] \\ [H]^T & [0] \end{bmatrix}_j \begin{Bmatrix} \Delta X \\ \Delta \lambda \end{Bmatrix}_j = -\begin{Bmatrix} \nabla L \\ h \end{Bmatrix}_j, \tag{5.69}$$

$$\Delta X_j = X_{j+1} - X_j, \tag{5.70}$$

$$\Delta \lambda_j = \lambda_{j+1} - \lambda_j, \tag{5.71}$$

where $[\nabla^2 L]_{n\times n}$ denotes the Hessian matrix of the Lagrange function.

The first set of Eq. (5.69) can be written separately as follows:

$$[\nabla^2 L]_j \Delta X_j + [H]_j \Delta \lambda_j = -\nabla L_j \tag{5.72}$$

By using Eq. (5.71) for $\Delta\lambda_j$ and Eq. (5.73) for ∇L_j, Eq. (5.72) can be expressed as follows:

$$[\nabla^2 L]_j \Delta X_j + [H]_j (\lambda_{j+1} - \lambda_j) = -\nabla f_j - [H]_j^T \lambda_j, \tag{5.73}$$

which can be simplified to obtain

$$[\nabla^2 L]_j \Delta X_j + [H]_j \lambda_{j+1} = -\nabla f_j. \tag{5.74}$$

Eq. (5.74) and the second set of equations in Eq. (5.69) can now be combined as follows:

$$\begin{bmatrix} [\nabla^2 L] & [H] \\ [H]^T & [0] \end{bmatrix}_j \begin{Bmatrix} \Delta X_j \\ \Delta \lambda_{j+1} \end{Bmatrix} = -\begin{Bmatrix} \nabla f_j \\ h_j \end{Bmatrix}. \tag{5.75}$$

Eq. (5.75) can be solved to find the change in the design vector ΔX_j and the new values of the Lagrange multipliers, λ_{j+1}. The iterative process indicated by Eq. (5.75) can be continued until convergence is achieved.

Consider the following programming problem.

Find ΔX that minimizes the quadratic objective function

$$Q = \nabla f^T \Delta X + \frac{1}{2} \Delta X^T \left[\nabla^2 L\right] \Delta X \tag{5.76}$$

subject to the linear equality constraints $h_k + \nabla h_k^T \Delta X = 0, k = 1, 2, \cdots, p$ or $h + [H]^T \Delta X = 0$.

The Lagrange function $\tilde{L}$, corresponding to the problem of Eq. (5.76) is given by

$$\tilde{L} = \nabla f^T \Delta X + \frac{1}{2} \Delta X^T \left[\nabla^2 L\right] \Delta X + \sum_{k=1}^{p} \lambda_k \left(h_k + \nabla h_k^T \Delta X\right), \tag{5.77}$$

where λ_k is the Lagrange multiplier associated with the kth equality constraint.

The Kuhn–Tucker necessary condition can be stated as follows:

$$\nabla f + \left[\nabla^2 L\right] \Delta X + [H] \lambda = 0, \tag{5.78}$$

$$h_k + \nabla h_k^T \Delta X = 0, k = 1, 2, \cdots, p. \tag{5.79}$$

Eqs. (5.78) and (5.79) can be identified to be the same as Eq. (5.75) in the matrix form. This shows that the original problem of Eq. (5.61) can be solved iteratively by solving the quadratic programming problem defined by Eq. (5.76). When inequality constraints are added to the original problem, the quadratic programming problem of Eq. (5.76) becomes

$$\text{Minimize } Q = \nabla f^T \Delta X + \frac{1}{2} \Delta X^T \left[\nabla^2 L\right] \Delta X$$

subject to

$$g_j + \nabla g_j^T \Delta X \le 0, \; j = 1, \; 2, \; \cdots, \; m, \tag{5.80}$$

$$h_k + \nabla h_k^T \Delta X = 0, \; k = 1, \; 2, \; \cdots, \; p,$$

with the Lagrange function given by

$$\tilde{L} = f(X) + \sum_{j=1}^{m} \lambda_j g_j(X) + \sum_{k=1}^{p} \lambda_{m+k} h_k(X) \tag{5.81}$$

Since the minimum of the augmented Lagrange function is involved, the sequential quadratic programming method is also known as the projected Lagrangian method.

5.7.2 Solution procedure

As in the case of Newton's method of unconstrained minimization, the solution vector ΔX in Eq. (5.80) is treated as search direction S, and the quadratic programming subproblem (in terms of the design vector S) is restated as follows:

$$\text{Minimize } Q(S) = \nabla f(X)^T S + \frac{1}{2} S^T [H] S$$

subjected to

$$\beta_j g_j(X) + \nabla g_j(X)^T S \leq 0, \; j = 1,\; 2,\; \cdots,\; m,$$

$$\overline{\beta}\, h_k(X) + \nabla h_j(X)^T S = 0,\; k = 1,\; 2,\; \cdots,\; p, \tag{5.82}$$

where $[H]$ is a positive definite matrix that is taken initially as the identity matrix and is updated in subsequent iterations to converge to the Hessian matrix of the Lagrange function of Eq. (5.81), and β_j and $\overline{\beta}$ are constants used to ensure that the linearized constraints do not cut off the feasible space completely. Typical values of these constants are given by

$$\overline{\beta} \approx 0.9;\; \beta_j = \begin{cases} 1 \text{if} g_j(X) \leq 0 \\ \overline{\beta} \text{ if} g_j(X) \geq 0 \end{cases}. \tag{5.83}$$

The subproblem of Eq. (5.82) is a quadratic programming problem. The problem can be solved by any of the methods discussed in this chapter as the gradients of the function involved can be evaluated easily. Since the Lagrange multipliers associated with the solution of the problem, Eq. (5.82), are needed, they can be evaluated as such. Once the search direction, S, is found by solving the problem in Eq. (5.82), the design vector is updated as follows:

$$X_{j+1} = X_j + \alpha^* S, \tag{5.84}$$

where α^* is the optimal step length along the direction S found by minimizing the function (using an exterior penalty function approach)

$$\phi = f(X) + \sum_{j=1}^{m} \lambda_j \left(max\left[0, g_j(X)\right]\right) + \sum_{k=1}^{p} \lambda_{m+k} \left| h_k(X) \right|, \tag{5.85}$$

with

$$\lambda_j = \begin{cases} \left|\lambda_j\right|,\; j = 1, 2, \cdots, m + p \text{ in the first iteration} \\ max\left\{ \left|\lambda_j\right|, \frac{1}{2}\left(\tilde{\lambda}_j, \left|\lambda_j\right| \right) \right\} \text{ in subsequent iterations} \end{cases} \tag{5.86}$$

and $\tilde{\lambda}_j = \lambda_j$ of the previous iteration.

The one-dimension step length α^* can be found by any of the standard methods.

Once X_{j+1} is found from Eq. (5.84), for the next iteration, the Hessian matrix $[H]$ is updated to improve the quadratic approximation in Eq. (5.82). Usually, a modified Broyden–Fletcher–Goldfarb–Shanno algorithm formula is used for this purpose

$$[H_{i+1}] = [H_i] - \frac{[H_i]P_iP_i^T[H_i]}{P_i^T[H_i]P_i} + \frac{\gamma\gamma^T}{P_i^TP_i} \tag{5.87}$$

$$P_i = X_{i+1} - X_i \tag{5.88}$$

$$\gamma = \theta Q_i + (1-\theta)[H_i]P_i \tag{5.89}$$

$$Q_i = \nabla_x \tilde{L}(X_{i+1}, \lambda_{i+1}) - \nabla_x \tilde{L}(X_i, \lambda_i) \tag{5.90}$$

$$\theta = \begin{cases} 1.0 \quad \text{if} \quad P_i^TQ_i \geq 0.2P_i^T[H_i]P_i \\ \dfrac{0.8P_i^T[H_i]P_i}{P_i^T[H_i]P_i - P_i^TQ_i} \quad \text{if} \quad P_i^TQ_i < 0.2P_i^T[H_i]P_i \end{cases} \tag{5.91}$$

where $\tilde{L}$ is given by Eq. (5.91) and the constants 0.2 and 0.8 in Eq. (5.91) can be changed, based on numerical experience.

5.8 Penalty function method

The penalty function method transforms the basic optimization problem into alternative formulations such that numerical solutions are sought by solving a sequence of unconstrained minimization problems. Consider the basic optimization problem, with inequality constraints, be of the form

$$\text{Minimize}\, f(X)$$

subject to

$$g_j(X) \leq 0, \; j = 1, \; 2, \; \cdots, \; m. \tag{5.92}$$

This optimization problem is converted into an unconstrained minimization problem by constructing a function of the form

$$\phi_k = \phi(X, r_k) = f(X) + r_k \sum_{j=1}^{m} G_j\left[g_j(X)\right] \tag{5.93}$$

where G_j is some function of the constraint g_j and r_k is a positive constant known as the penalty parameter. The significance of the second term on the right side of Eq. (5.93) is called penalty term. If the unconstrained minimization of the ϕ function is repeated for a sequence of values of the penalty parameter $r_k(k = 1, 2, \cdots)$, the solution may be brought to converge to that of the original problem stated in Eq. (5.92). This is the reason why the penalty function methods are also known as sequential unconstrained minimization techniques.

The penalty function formulations for inequality constrained problems can be divided into two categories: interior and exterior methods. In the interior formulations, some popularly used forms of G_j are given by

$$G_j = -\frac{1}{g_j(X)}. \tag{5.94}$$

$$G_j = log\left[-g_j(X)\right]. \tag{5.95}$$

Some commonly used forms of the function G_j in the case of the exterior penalty, function formulations are as follows:

$$G_j = \max\left[0, g_j(X)\right]. \tag{5.96}$$

$$G_j = \left\{\max\left[0, g_j(X)\right]\right\}^2. \tag{5.97}$$

In the interior methods, the unconstrained minima of ϕ_k all lie in the feasible region and converge to the solution of Eq. (5.92) as r_k is varied in a particular manner. In the exterior methods, the unconstrained minima of ϕ_k all lie in the infeasible region and converge to the desired solution from the outside as r_k is changed in a specified manner. The convergence of the unconstrained minima of ϕ_k is illustrated for a simple problem:

$$\text{Minimize}\, f(X) = \alpha x_1$$

subject to

$$g_1(X) = \beta - x_1 \leq 0. \tag{5.98}$$

It can be observed that the unconstrained minima of $\phi(X, r_k)$ converges to the optimum point X^* as the parameter r_k is increased sequentially. Conversely, the interior method gives convergence as the parameter r_k is decreased sequentially.

There are several reasons for the appeal of the penalty function formulations. One main reason, which can be observed, is that the sequential nature of the method allows a gradual or sequential approach to the criticality of the constraints. Besides, the sequential process permits a graded approximation to be used in the analysis of the system. This means that if the evaluation of f and g_j [and hence $\phi(X, r_k)$] for any specified design vector X is computationally very difficult, we can use coarse approximations during the early stages of optimization (when the unconstrained minima of ϕ_k are far away from the optimum) and finer or more detailed analysis approximation during the final stages of optimization. Another reason is that the algorithms for the unconstrained minimization of rather arbitrary functions are well studied and generally are quite reliable.

5.9 Interior penalty function method

This is well known in the fields of science and engineering design. In this method, the initial point is selected in such a way that it should lie in the feasible region. For a design engineer, selecting a point in the feasible region is not at all a problem, since the usual design procedure gives a point in the feasible region.

Penalized objective function ϕ is formed as follows:

$$\phi(X, r_k) = f(X) - r_k \sum_{j=1}^{m} \frac{1}{g_j(X)}. \tag{5.99}$$

The minimization of ϕ is taken up with an assumed initial value of r_1 and the problem is treated as an unconstrained optimization problem. Like the design, the point is in the feasible region if $g_j(X)$ is a negative quantity. Hence, ϕ is always greater than $f(X)$. As the design point approaches closer to constraint, $g_j(X)$ reduces and hence ϕ increases. Therefore, the search never crosses the boundary. Thus, the search for ϕ is always within the feasible region.

In subsequent iterations, r_k is decreased, and hence, the optimum point of ϕ comes closer to the constraints but never allowed to cross. Since the penalty term acts as a barrier to design points entering the infeasible regions, this method is also known as the barrier method. This method is called an interior penalty method since the design point is always inside the feasible region.

The interior penalty function method algorithm may also be stated as follows.

Algorithm 5.3.

```
Step 1: Start with a feasible design point $X_1$ and select the initial value of $r_1$
which is more than 1. Set $K = 1$.
Step 2: Find the vector $X_k^*$ that minimizes the modified objective function
$\phi(X, r_k) = f(X) - r_k \sum_{j=1}^{m} \frac{1}{g_j(X)}$. For this use any unconstrained optimization
technique.
Step 3: Test whether the point $X_k^*$ is the optimum of the original problem. If it
satisfies termination criteria stop the iterative process. Otherwise, go to Step 4.
Step 4: Find the value of the next penalty parameter $r_{k+1} = Cr_k$, where
$0 < C < 1$ usually it is taken as 0.1, 0.2, or 0.5.
Step 5: Set $k = k + 1$ and go to Step 2.
```

5.10 Convex programming problem

A function $f(X)$ is said to be convex if for any pair of points $U = (u_1, u_2, \cdots, u_n)^T$, $V = (v_1, v_2, \cdots, v_n)^T \in X$, and $0 \le \lambda \le 1$,

$$f(\lambda V + (1-\lambda)U) \le \lambda f(V) + (1-\lambda)f(U) \tag{5.100}$$

That is if the segment joining the two points lies entirely above or on the graph of $f(X)$. It can be seen that a convex function is always bending upward, and hence, it is apparent that the local minimum of a convex function is also a global minimum.

A function $f(X)$ is said to be concave if for any two points U, $V \in X$ and $0 \le \lambda \le 1$,

$$f(\lambda V + (1-\lambda)U) \ge \lambda f(V) + (1-\lambda)f(U) \tag{5.101}$$

That is if the line segment joining two points lies entirely below or on the graph of $f(X)$. It can be seen that a concave function bends downward, and hence, the local maximum will also be its global maximum. It can be observed that the negative of a convex function is a concave function, and vice versa. Also, note that the sum of convex functions is a convex function, and the sum of the concave functions is a concave function. A function $f(X)$ is strictly convex or concave if the strict inequality holds in Eqs. (5.100) and (5.101) for any $U \neq V$. That is, a strictly convex function should satisfy the condition, $f(\lambda V + (1-\lambda)U) < \lambda f(V) + (1-\lambda)f(U)$. However, a strictly concave function should satisfy the condition, $f(\lambda V + (1-\lambda)U) > \lambda f(V) + (1-\lambda)f(U)$.

Theorem: A function $f(X)$ is convex if for any two points U and V, we have

$$f(V) \geq f(U) + \nabla f^T(U)(V-U). \tag{5.102}$$

Proof:. If $f(X)$ is convex, we have by definition

$$f(\lambda V + (1-\lambda)U) \leq \lambda f(V) + (1-\lambda)f(U)$$

that is, $f(U + \lambda(V-U)) \leq f(U) + \lambda(f(V) - f(U))$.

The inequality can be rewritten as follows:

$$f(V) - f(U) \geq \left\{\frac{f(U + \lambda(V-U)) - f(U)}{\lambda(V-U)}\right\}(V-U) \tag{5.103}$$

By defining $\Delta X = \lambda(V-U)$, the inequality Eq. (5.103) can be written as follows:

$$f(V) - f(U) \geq \left\{\frac{f(U + \Delta X) - f(U)}{\Delta X}\right\}(V-U) \tag{5.104}$$

By taking the limit as $\Delta X \to 0$, inequality Eq. (5.104) becomes

$$f(V) - f(U) \geq \nabla f^T(U)(V-U), \tag{5.105}$$

which can be seen to be the desired result.

Similarly, it can be noted that for a function $f(X)$ is concave if for any two points U and V, we have

$$f(V) \leq f(U) + \nabla f^T(U)(V-U). \tag{5.106}$$

By using the definition of a convex function and Taylor's theorem, we get the following conclusions.

1. A function $f(X)$ is convex if the Hessian matrix $H(X) = \left[\frac{\partial^2 f(X)}{\partial x_i \partial x_j}\right]$, for $i,j = 1, 2, \cdots, n$ is positive semidefinite.
2. A function $f(X)$ is strictly convex if the Hessian matrix $H(X) = \left[\frac{\partial^2 f(X)}{\partial x_i \partial x_j}\right]$, for $i,j = 1, 2, \cdots, n$ is positive definite.
3. A function $f(X)$ is concave if the Hessian matrix $H(X) = \left[\frac{\partial^2 f(X)}{\partial x_i \partial x_j}\right]$, for $i,j = 1, 2, \cdots, n$ is negative semidefinite.

4. A function $f(X)$ is strictly concave if the Hessian matrix $H(X) = \left[\frac{\partial^2 f(X)}{\partial x_i \partial x_j}\right]$, for $i, j = 1, 2, \cdots, n$ is negative definite.
5. Any local minimum of a convex function $f(X)$ is a global minimum.
6. Any local maximum of a concave function $f(X)$ is a global maximum.

In Section 5.9, we see that the sequential minimization of

$$\phi(X, r_k) = f(X) - r_k \sum_{j=1}^{m} \frac{1}{g_j(X)}, r_k > 0 \tag{5.107}$$

For a decreasing sequence of values of r_k gives the minima X_k^*. As $k \to \infty$, these points X_k^* converge to the minimum of the constrained problem:

$$\text{Minimize } f(X)$$

subject to

$$g_j(X) \leq 0, \ j = 1, \ 2, \ \cdots, \ m. \tag{5.108}$$

To ensure the existence of a global minimum of $\phi(X, r_k)$ for every positive value of r_k, ϕ has to be a strictly convex function of X. The following theorem gives sufficient conditions for the ϕ function to be strictly convex. If ϕ is convex, for every $r_k > 0$, there exists a unique minimum of $\phi(X, r_k)$.

Theorem: If $f(X)$ and $g_j(X)$ are convex and at least one of $f(X)$ and $g_j(X)$ is strictly convex, the function $\phi(X, r_k)$ defined by Eq. (5.107) will be a strictly convex function of X.

Proof:. This theorem can be proved in two steps. In the first step, we prove that if a function $g_k(X)$ is convex, $1/(g_k(X))$ will be concave. In the second step, we prove that a positive combination of a convex function is convex and strictly convex if at least one of the function is strictly convex.

By using the above property "Any local minimum of a convex function $f(X)$ is a global minimum," it can ensure that the sequential minimization of $\phi(X, r_k)$ for a decreasing sequence of values of r_k leads to the global minimum of the original constrained problem. When the convexity conditions are not satisfied, or when the functions are so complex that we do not know beforehand whether the convexity conditions are satisfied, it will not be possible to prove that the minimum found by the sequential unconstrained minimization technique is a global one. In such cases, one has to satisfy a local minimum only. However, one can always reapply the sequential unconstrained minimization technique from different feasible starting points and try to find a better local minimum point if the problem has several local minima. Of course, this procedure requires more computational effort.

5.11 Exterior penalty function method

In the exterior penalty function method, the ϕ function is generally taken as follows:

$$\phi(X, r_k) = f(X) + r_k \sum_{j=1}^{m} \langle g_j(X) \rangle^q, \tag{5.109}$$

where r_k is a positive penalty parameter, the exponent q is a nonnegative constant, and the bracket function $\langle g_j(X) \rangle$ is defined as follows:

$$\langle g_j(X) \rangle = max\langle g_j(X), 0 \rangle = \begin{cases} g_j(X) & \text{if } g_j(X) > 0 \text{ (constraint is violated)} \\ 0 & \text{if } g_j(X) \le 0 \text{ (constraint is satisfied)} \end{cases} \tag{5.110}$$

It is seen from Eq. (5.109) that the effect of the second term on the right side is to increase $\phi(X, r_k)$ in proportion to the qth power of the amount by which the constraints are violated. Thus there will be a penalty for violating the constraints, and the amount of penalty will increase at a faster rate than the amount of violation of a constraint (for $q > 1$). This is the reason why the formulation is called the penalty function method. Usually, the function $\phi(X, r_k)$ possesses a minimum as a function of X in the infeasible region. The unconstrained minima X_k^* converge to the optimal solution of the original problem as $k \to \infty$ and $r_k \to \infty$. Thus the unconstrained minima approach the feasible domain gradually, and as $k \to \infty$, the X_k^* eventually lies in the feasible region. Let us consider Eq. (5.109) for various values of q.

1. When $q = 0$. Here, the ϕ function is given by

$$\phi(X, r_k) = f(X) + r_k \sum_{j=1}^{m} \langle g_j(X) \rangle^0 = \begin{cases} f(X) + m_k & \text{if } \forall g_j(X) > 0 \\ f(X) & \text{if } \forall g_j(X) \le 0 \end{cases} \tag{5.111}$$

This function is discontinuous on the boundary of the acceptable region, and hence, it would be very difficult to minimize this function.

2. When $0 < q < 1$. Here, the ϕ function will be continuous, but the penalty for violating a constraint may be too small. Also, the derivatives of the function are discontinuous along the boundary. Thus it will be difficult to minimize the ϕ function.
3. When $q = 1$. In this case, under certain restrictions, it has been shown that there exists an r_0 so large that the minimum of $\phi(X, r_k)$ is exactly the constrained minimum of the original problem for all $r_k > r_0$. However, the contours of the ϕ function look similar and possess discontinuous first derivatives along the boundary. Hence despite the convenience of choosing a single r_k that yields the constrained minimum in one unconstrained minimization, the method is not very attractive from the computational point of view.
4. When $q > 1$. The ϕ function will have continuous first derivatives in this case. These derivatives are given by

$$\frac{\partial \phi}{\partial x_i} = \frac{\partial f}{\partial x_i} + r_k \sum_{j=1}^{m} q\left[g_j(X)\right]^{q-1} \frac{\partial g_j(X)}{\partial x_i}. \tag{5.112}$$

Generally, the value of q is chosen as 2 in practical computation.

Algorithm 5.4.

Step 1: Start from any design X_1 and a suitable value of r_1. Set $k = 1$.
Step 2: Find the vector X_k^* that minimizes the function
$\phi(X, r_k) = f(X) + r_k \sum_{j=1}^{m} \left[g_j(X)\right]^q$.
Step 3: Test whether the point X_k^* satisfies all the constraints. If X_k^* is feasible, it is the desired optimum and hence terminate the procedure. Otherwise, go to Step 4.
Step 4: Choose the next value of the penalty parameter that satisfies the relation
$r_{k+1} > r_k$
and set the new value of k as original k plus 1 and go to Step 2.

Usually, the value of r_{k+1} is chosen according to the relation $r_{k+1} = cr_k$, where c is a constant greater than 1.

Practice set

1. Determine the dimensions of the longest rectangular section in flexure that can be cut from a log having a section of radius r.
2. Show that the cone of the greatest volume, which can be inscribed in a given sphere, has an altitude equal to 4/3 of the radius of the sphere. Prove that the curved surface of the cone is a maximum for the same.
3. Find the dimension of a cylinder with top and bottom made up of sheet metal to maximize its volume such that the total surface area is equal to 24π.
4. Find the points closest to the origin on $xy^2 = 54$ and its distance from the origin.
5. Use the Lagrange multiplier method to find the greatest and least distances from the point $(2, 1, -2)$ to the sphere with the equation $x_1^2 + x_2^2 + x_3^2 = 1$.
6. Use the Lagrange multiplier method to find the least distance from the origin to the plane $x_1 + 2x_2 + 2x_3 = 3$.
7. Minimize $f(x_1, x_2) = x_1^2 + x_2^2 + 60x_1$ subject to the constraints $g_1 = x_1 - 80 \geq 0, g_2 = x_1 + x_2 - 120 \geq 0$ using the Kuhn–Tucker conditions.
8. Minimize $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 3x_3^2$ subject to the constraints $g_1 = x_1 - x_2 - 2x_3 \leq 12, g_2 = x_1 + 2x_2 - 3x_3 \leq 8$ using the Kuhn–Tucker conditions.
9. Maximize $f(x, y) = 8x + 4y + xy - x^2 - y^2$ subject to $2x + 3y \leq 24$, $-5x + 12y \leq 24$, and $y \leq 5$ using the Kuhn–Tucker conditions.
10. Minimize $F = 9 - 8x - 6y - 4z + 2x^2 + 2y^2 + z^2 + 2xy + 2xz$ subject to $x + y + 2z = 3$ using (1) constrained variation method and (2) Lagrange multiplier method.
11. Find the extreme or optimum point of the function $Z = x^3 - y^3 - 2x^2 + 4y^2 + 12$ and identify whether they are the relative maximum, relative minimum or points of inflection.
12. Write down the Kuhn–Tucker conditions for the following problem

$$\text{Maximize } 3x^2 - 2y$$

subject to

$$2x + y = 4$$

$$x^2 + y^2 \leq 19.4$$

$$x \geq 0.$$

Find out whether points $(0,4)^T$ and $(3.4,-2.8)^T$ are Kuhn–Tucker points. How would the maximum function value change if the equality constraint is changed to the following:

$$2x + y = 6.$$

13. Using the Lagrange method, find the point on the ellipse defined by the intersection of the surfaces $x_1 + x_2 = 1$ and $x_1^2 + 2x_2^2 + x_3 = 1$ and is nearest to the origin.

14. The intersection of the planar surface $x + y + 4z = 2$ and a cone $x^2 + 2y^2 + z^2 = 1$ creates an ellipse.

   a. Formulate an optimization problem to locate a point on the ellipse, which is nearest to the origin.

   b. Write and solve the resulting Karush–Kuhn–Tucker conditions to find the nearest point.

15. Using Zoutendijk's feasible direction method, solve the following problem:

$$\text{Minimize}\, f(x) = (x-5)^2$$

subject to

$$2 \le x \le 3.$$

Consider a starting point $x^{(0)} = 2$.

16. Using the feasible direction method at the point $x = (1,1)^T$ and a direction vector $d = (1, 0.14)^T$, solve the following problem:

$$\text{Maximize } z = (x-1.5)^2 + (y-4)^2$$

subject to

$$4.5x + y^2 \le 18$$

$$2x - y \ge 1$$

$$x, y \ge 0.$$

Consider two iterations of the golden section search method to bracket the minimum point along the direction d. Assume the mid-point of that interval as the new point of x. Create a search direction at the new point of x.

Further reading

Bazaraa, M., & Shetty, C. (1979). *Nonlinear programming: Theory and algorithms*. New York, NY: Wiley.

Bazaraa, M., Sherali, H., & Shetty, C. (1993). *Nonlinear programming: Theory and algorithms*. New York, NY: Wiley.

Beveridge, G., & Schechter, R. (1970). *Optimization: Theory and practice*. New York, NY: McGraw-Hill.

Bhatti, M. A. (2013). *Practical optimization methods: With Mathematica® applications*. New York, NY: Springer Verlag.

Bhavikatti, S. S. (2010). *Fundamentals of optimum design in engineering*. New Delhi: New Age International (P) Limited Publishers.

Cassis, J., & Schmit, L. (1976). On implementation of the extended interior penalty function. *International Journal for Numerical Methods in Engineering*, *10*(1), 3–23.

Deb, K. (2004). *Optimization for engineering design algorithms and examples*. New Delhi: Prentice-Hall of India.

Fox, R. (1971). *Optimization methods for engineering design*. Reading, MA: Addison-Wesley.

Gabriele, G., & Ragsdell, K. (1977). The generalized reduced gradient method: A reliable tool for optimal design. *ASME Journal of Engineering for Industry*, *99*(2), 384–400.

Geromel, J., & Baptistella, L. (1981). Feasible direction method for large-scale nonconvex programs: Decomposition approach. *Journal of Optimization Theory and Applications*, 231–249.

Haftka, R., & Starnes, J. (1976). Application of a quadratic extended interior penalty function for structural optimization. *AIAA Journal*, 718–728.

Kamat, M. P. (1993). *Structural optimization: Status and promise*. Washington, DC: AIAA.

Nash, S., & Sofer, A. (1996). *Linear and nonlinear programming*. New York: McGraw-Hill.

Nayak, S., & Chakraverty, S. (2013). Non-probabilistic approach to investigate uncertain conjugate heat transfer in an imprecisely defined plate. *International Journal of Heat and Mass Transfer*, *67*, 445–454.

Panik, M. (1976). Classical optimization: Foundations and extensions. Amsterdam: North-Holland.

Powell, M. (1978). *A fast algorithm for nonlinearity constrained optimization calculations*. Berlin: Springer Verlag.

Prasad, B. (1981). A class of generalized variable penalty methods for nonlinear programming. *Journal of Optimization Theory and Applications*, *35*(2), 159–182.

Rao, S. S. (1995). *Optimization theory and applications*. New Delhi: New Age International (P) Limited Publishers.

Reklaitis, G., Ravindran, A., & Ragsdell, K. (1983). *Engineering optimization—Methods and applications*. New York, NY: Wiley.

Rosen, J. (1960). The gradient projection method of nonlinear programming, Part I: Linear constraints. *SIAM Journal*, *8*, 181–217.

Sharma, J. K. (2006). *Operations research theory & applications*. New Delhi: Macmillan India Ltd.

Simmons, D. (1975). *Nonlinear programming for operations research*. Englewood Cliffs, NJ: Prentice-Hall.

Wright, J. N. (2006). *Numerical optimization*. New York, NY: Springer.

Gurdal, R. T. (1998). *Design and optimization of laminated composite*. New York, NY: Wiley.

Zoutendijk, G. (1960). *Methods of feasible directions*. Amsterdam: Elsevier.

Zoutendijk, G. (1966). Nonlinear programming: A numerical survey. *SIAM Journal of Control Theory and Applications*, *4*(1), 194–210.

CHAPTER SIX

Geometric programming

The concept of geometric programming can be started with an interesting example problem of molecular geometry. The geometry of a molecule plays a crucial role to determine many of its chemical and physical properties. The following three measurements are carried out to study molecular geometry.

1. Bond angles: The angles between two bonds or two bonded electron pairs in a compound. It is measured in degrees.
2. Bond distances: The average distance between the nuclei of two bonded atoms in a molecule. It is measured in angstroms.
3. Dihedral angles: The angle between two intersecting planes. In other words these are the angles between planes through two sets of three atoms, having two atoms in common. It is measured in degrees.

In this context the objective of geometric optimization is to investigate molecular stability by using the atomic arrangement. As we know that molecules are more stable when they possess low energy, to optimize a molecular geometry, we need to check various possibilities to see which one has the lowest energy value. This can be observed by creating a potential energy surface (PES). The PES can be defined as a mathematical relationship between different molecular geometries and their corresponding single-point energies. The PES values are generally shown in a three-dimensional graph. The three dimensions represent the bond angle, bond distance, and Hartree–Fock energy values. The PES can be characterized by the following distinct points:

1. The point at which the lowest value in a particular section or region of the PES exists is called a local minimum point.
2. The point at which the lowest value in the entire PES exists is called the global minimum point.
3. The point at which the highest value in a particular section or region of the PES exists is called a local maximum point.
4. The point at which the highest value in the entire PES exists is called the global maximum point.
5. The point on the PES at which a maximum in one direction and a minimum in the other direction is called a saddle point. Saddle points show a transition connecting two equilibrium structures.

© 2020 Elsevier Inc.
All rights reserved.

Many optimization techniques in which emphasis is on finding variables corresponding to optimum design. After finding such variables optimum objective function is determined. But in geometric programming emphasis is to find relative magnitudes of the terms of the objective function first and then determine the corresponding variables. The main advantage of this method is that it requires the objective function and the constraints in the form of posynomials.

6.1 Posynomial

A posynomial is defined as

$$f(X) = \sum_{j=1}^{N} U_j(X) \tag{6.1}$$

where

$$U_j(X) = C_j x_1^{a_{1j}} x_2^{a_{2j}} \cdots x_n^{a_{nj}} = C_j \prod_{i=1}^{n} x_i^{a_{ij}}. \tag{6.2}$$

Here the coefficient C_j is a positive constant; design parameter x_i is also a positive variable; exponents a_{ij} are real constants (positive, negative, or zero). For example,

$$f(X) = x_1x_2x_3 + x_1^3x_2 + 4x_3^2 + \frac{2}{x_1x_2} + 3x_3^{3/2}, \tag{6.3}$$

subject to $x_i \geq 0$ for $i = 1, 2, 3$ is a posynomial.

Eq. (6.3) can be written as

$$f(X) = U_1 + U_2 + U_3 + U_4 + U_5 \tag{6.4}$$

The coefficients are $C_1 = C_2 = 1, C_3 = 4, C_4 = 2, C_5 = 3,$ x_i are positive variables and

$$a_{11} = 1, a_{21} = 1, a_{31} = 1,$$

$$a_{12} = 3, a_{22} = 1, a_{32} = 0,$$

$$a_{13} = 0, a_{23} = 0, a_{33} = 2,$$

$$a_{14} = -1, a_{24} = -1, a_{34} = 0,$$

$$a_{15} = 0, a_{25} = 0, a_{35} = \frac{3}{2},$$

are real constants. Here the number of terms $N = 5$ and design parameters $n = 3$.

6.2 Unconstrained geometric programming program

The unconstrained geometric optimization problem can be stated as

$$\text{Minimize } f(X) = \sum_{j=1}^{N} U_j(X) = \sum_{j=1}^{N} C_j \prod_{i=1}^{n} x_i^{a_{ij}} = \sum_{j=1}^{N} C_j x_1^{a_{1j}} x_2^{a_{2j}} \cdots x_n^{a_{nj}} \quad (6.5)$$

where $X = (x_1, x_2, \cdots, x_n)^T$, $C_j > 0$, $x_j > 0$, and a_{ij} are real numbers.

The necessary condition for minimization gives $\partial f(X)/\partial x_k = 0$ for $k = 1, 2, \cdots, n$, that is

$$\frac{\partial}{\partial x_k}\left(\sum_{j=1}^{N} C_j x_1^{a_{1j}} x_2^{a_{2j}} \cdots x_n^{a_{nj}}\right) = 0$$

$$\Rightarrow \sum_{j=1}^{N} C_j x_1^{a_{1j}} x_2^{a_{2j}} \cdots a_{kj} x_k^{a_{kj}-1} \cdots x_n^{a_{nj}} = 0. \quad (6.6)$$

Multiplying x_k with Eq. (6.6) we have

$$x_k \frac{\partial f}{\partial x_k} = \sum_{j=1}^{N} a_{kj} U_j(X) = 0 \text{ for } k = 1, 2, \cdots, n. \quad (6.7)$$

Let $X^* = \left(x_1^*, x_2^*, \cdots, x_3^*\right)^T$ be the minimum for corresponding problem Eq. (6.5). Then we may write

$$\sum_{j=1}^{N} a_{kj} U_j(X^*) = 0 \text{ for } k = 1, 2, \cdots, n. \quad (6.8)$$

Now, $\frac{1}{f(X^*)} \sum_{j=1}^{N} a_{kj} U_j(X^*) = 0$ can be represented as

$$\sum_{j=1}^{N} a_{kj} \frac{U_j(X^*)}{f(X^*)} = 0. \quad (6.9)$$

Assume, $\delta_j^* = \frac{U_j(X^*)}{f(X^*)}$, then Eq. (6.9) becomes

$$\sum_{j=1}^{N} a_{kj}\delta_j^* = 0. \tag{6.10}$$

It may be noted that δ_j^* represents the relative contribution to the optimum objective function by each of N terms. Therefore we get

$$\sum_{j=1}^{N} \delta_j^* = \delta_1^* + \delta_2^* + \cdots + \delta_N^*$$

$$= \frac{1}{f(X^*)}[U_1(X^*) + U_2(X^*) + \cdots + U_N(X^*)] = \frac{f(X^*)}{f(X^*)} = 1. \tag{6.11}$$

In compact form Eq. (6.11) can be rewritten as $\sum_{j=1}^{N} \delta_j^* = 1$. For a better understanding of Eq. (6.11) we need to know δ_j^*. So, in the following, we will obtain δ_j^*.

Find $\boldsymbol{\delta_j^*}$

Based on the linear combinations of δ_j^*, $j = 1, 2, \cdots, N$, we have

1. $\sum_{j=1}^{N} \delta_j^* a_{kj} = 0$ is called orthogonality condition and
2. $\sum_{j=1}^{N} \delta_j^* = 1$ is called normality condition.

Hence there are n number of orthogonality equations and one normality equation.

If $N = n + 1$, then there are as many equations as there are unknowns. Therefore the simultaneous equations may be solved to get δ_j^* values uniquely. In this case it is called the degree of difficulty zero.

If $N > n + 1$, then we have more unknowns and less number of equations. Then the degree of difficulty is $N - (n + 1)$. For such case the linear programming solution is possible to find δ_j^* values to minimize $f(x)$.

If $N < n + 1$, then there is no solution.

Find $\boldsymbol{f(X^*)}$

The objective function at the optimal point X^* can be represented as

$$f(X^*) = (f(X^*))^1 = (f(X^*))^{\delta_1^* + \delta_2^* + \cdots + \delta_N^*} = (f(X^*))^{\delta_1^*}(f(X^*))^{\delta_2^*} \cdots (f(X^*))^{\delta_N^*}. \tag{6.12}$$

It is known from Eq. (6.9) that $\delta_j^* = \frac{U_j(X^*)}{f(X^*)} = \frac{U_j^*}{f^*}$. So we can write

$$f^* = \frac{U_1^*}{\delta_1^*} = \frac{U_2^*}{\delta_2^*} = \cdots = \frac{U_N^*}{\delta_N^*}. \tag{6.13}$$

Using Eq. (6.12) in Eq. (6.13) we have

$$f^* = \left(\frac{U_1^*}{\delta_1^*}\right)^{\delta_1^*}\left(\frac{U_2^*}{\delta_2^*}\right)^{\delta_2^*}\cdots\left(\frac{U_N^*}{\delta_N^*}\right)^{\delta_N^*}. \tag{6.14}$$

But $U_j^* = C_j\prod_{i=1}^{n}\left(x_i^*\right)^{a_{ij}}, i = 1, 2, \cdots, n$. Therefore

$$f^* = \left(\frac{C_1}{\delta_1^*}\right)^{\delta_1^*}\prod_{i=1}^{n}\left(x_i^*\right)^{a_{i1}\delta_1^*}\left(\frac{C_2}{\delta_2^*}\right)^{\delta_2^*}\prod_{i=1}^{n}\left(x_i^*\right)^{a_{i2}\delta_2^*}\cdots\left(\frac{C_N}{\delta_N^*}\right)^{\delta_1^*}\prod_{i=1}^{n}\left(x_i^*\right)^{a_{iN}\delta_N^*}$$

$$= \prod_{j=1}^{N}\left(\frac{C_j}{\delta_j^*}\right)^{\delta_j^*}\left(x_1^*\right)^{\sum_{j=1}^{N}a_{1j}\delta_j^*}\left(x_2^*\right)^{\sum_{j=1}^{N}a_{2j}\delta_j^*}\cdots\left(x_n^*\right)^{\sum_{j=1}^{N}a_{nj}\delta_j^*}. \tag{6.15}$$

Using orthogonality condition in Eq. (6.15), we get

$$f^* = \prod_{j=1}^{N}\left(\frac{C_j}{\delta_j^*}\right)^{\delta_j^*}. \tag{6.16}$$

Let us illustrate this concept using an example.

Example 6.1. Optimize the following unconstrained geometric optimization problem.

$$\text{Minimize} f(x) = x_1 + x_2 + \frac{1}{x_1 x_2}, x_1, x_2 > 0.$$

Solution: The given objective function can be written as

$$f(x) = x_1 + x_2 + \frac{1}{x_1 x_2} = U_1 + U_2 + U_3. \tag{6.17}$$

Here the coefficients are $C_1 = 1, C_2 = 1, C_3 = 1$; and $N = 3, n = 2$. The $a_{ij},\ i = 1, 2$ and $j = 1,\ 2,\ 3$ values are

$$a_{11} = 1, a_{12} = 0, a_{13} = -1$$

$$a_{21} = 0, a_{22} = 1, a_{23} = -1$$

Since, $N = n + 1$, the degree of difficulty is zero. As such we can write the objective function at the optimum point in the following way:

$$f^* = \left(\frac{C_1}{\delta_1^*}\right)^{\delta_1^*}\left(\frac{C_2}{\delta_2^*}\right)^{\delta_2^*}\left(\frac{C_3}{\delta_3^*}\right)^{\delta_3^*}, \tag{6.18}$$

where, $\delta_1^* = \frac{U_1(X^*)}{f(X^*)}, \delta_2^* = \frac{U_2(X^*)}{f(X^*)}, \delta_3^* = \frac{U_3(X^*)}{f(X^*)}$.

Using orthogonality and normality conditions, we have

$$\begin{aligned} a_{11}\delta_1^* + a_{12}\delta_2^* + a_{13}\delta_3^* &= 0 \\ a_{21}\delta_1^* + a_{22}\delta_2^* + a_{23}\delta_3^* &= 0 \\ \delta_1^* + \delta_2^* + \delta_3^* &= 1 \end{aligned} \tag{6.19}$$

In matrix form, Eq. (6.19) becomes

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 1 & 1 & 1 \end{pmatrix}\begin{pmatrix} \delta_1^* \\ \delta_2^* \\ \delta_3^* \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{6.20}$$

From Eq. (6.20), the following is obtained

$$\delta_1^* - \delta_3^* = 0 \Longrightarrow \delta_1^* = \delta_3^* \text{ and } \delta_2^* - \delta_3^* = 0 \Longrightarrow \delta_2^* = \delta_3^*.$$

Therefore $\delta_1^* = \delta_2^* = \delta_3^*$ and $\delta_1^* = \delta_2^* = \delta_3^* = \frac{1}{3}$.

Using the δ_1^*, δ_2^*, and δ_3^* values, the optimal function value is

$$f^* = \left(\frac{1}{\frac{1}{3}}\right)^{1/3}\left(\frac{1}{\frac{1}{3}}\right)^{1/3}\left(\frac{1}{\frac{1}{3}}\right)^{1/3} = (3)^{1/3}(3)^{1/3}(3)^{1/3} = 3.$$

The next task is to investigate the unknown x_1 and x_2 values.

From Eq. (6.20), we get $\delta_1^* = \delta_2^* = \delta_3^*$. As such

$$\frac{U_1}{f^*} = \frac{U_2}{f^*} = \frac{U_3}{f^*} \Longrightarrow \frac{x_1}{3} = \frac{x_2}{3} = \frac{\frac{1}{x_1 x_2}}{3}. \tag{6.21}$$

Solving Eq. (6.21), we have

$$x_1 = x_2$$

and

$$\frac{x_2}{3} = \frac{\frac{1}{x_1 x_2}}{3} \Longrightarrow x_2 = \frac{1}{x_1 x_2} \Longrightarrow x_1 x_2^2 = 1 \Longrightarrow x_1^3 = 1 \Longrightarrow x_1 = 1.$$

Hence $x_1 = x_2 = 1$ and we conclude that at $x_1 = 1$ and $x_2 = 1$; the minimum value of the given function is $f^* = 1 + 1 + 1 = 3$.

Example 6.2. An open cylindrical vessel is to be constructed to transport 100m^3 of oil from a store to a factory. The metal sheet used for the bottom costs ₹1000.00 and that used for cylindrical wall costs ₹500.00 per square meter. If it costs ₹100.00 for each round trip of the vessel, find the dimension of the vessel for minimizing the transportation cost. Assume that the vessel has no salvage upon completion of the operation.

Solution: In this problem, the objective function is the cost of the transportation of 100m^3 oil, that is $f(X)$ which is the sum of the cost of the bottom vessel, cost of cylindrical wall, and cost of transporting oil.

Let the radius of the cylindrical vessel be x and height be y. Then the volume of the cylinder is $\pi x^2 y$.

Number of trips for transporting 100m^3 oil is $100/\pi x^2 y$. Hence we can write the objective function

$$f(X) = \pi x^2 \times 1000 + 2\pi xy \times 500 + \frac{100}{\pi x^2 y} \times 100 = \sum_{j=1}^{3} C_j x^{a_{1j}} y^{a_{2j}}, \quad (6.22)$$

where $C_1 = 1000\pi$, $C_2 = 1000\pi$, and $C_3 = \frac{10{,}000}{\pi}$; $N = 3$, $n = 2$.

$$a_{11} = 2, a_{21} = 0$$

$$a_{12} = 1, a_{22} = 1$$

$$a_{13} = -2, a_{23} = -1$$

Consider the weights δ_1^*, δ_2^*, and δ_3^* to be assigned to the three terms in $f(X)$. Then from orthogonality condition $\delta_j^* a_{kj} = 0$ and from normality condition we have $\delta_1^* + \delta_2^* + \delta_3^* = 1$. Therefore

$$2\delta_1^* + \delta_2^* - 2\delta_3^* = 0 \quad (6.23)$$

$$\delta_2^* - \delta_3^* = 0 \quad (6.24)$$

$$\delta_1^* + \delta_2^* + \delta_3^* = 1 \quad (6.25)$$

Eqs. (6.23)–(6.25) can be represented in the following matrix form

$$\begin{bmatrix} 2 & 1 & -2 \\ 0 & 1 & -1 \\ 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} \delta_1^* \\ \delta_2^* \\ \delta_3^* \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}. \quad (6.26)$$

From Eq. (6.24) we get $\delta_2^* = \delta_3^*$. Then substituting the value of δ_2^* in Eq. (6.23), we obtain

$$2\delta_1^* - \delta_3^* = 0 \Rightarrow \delta_1^* = \frac{\delta_3^*}{2}.$$

Hence Eq. (6.25) becomes

$$\frac{\delta_3^*}{2} + \delta_3^* + \delta_3^* = 1 \Rightarrow \delta_3^* = 0.4.$$

Therefore $\delta_2^* = 0.4$ and $\delta_1^* = 0.2$.

Using the obtained weights δ_1^*, δ_2^*, and δ_3^*, the minimum cost is

$$f^* = \left(\frac{C_1}{\delta_1^*}\right)^{\delta_1^*}\left(\frac{C_2}{\delta_2^*}\right)^{\delta_2^*}\left(\frac{C_3}{\delta_3^*}\right)^{\delta_3^*} = \left(\frac{1000\pi}{0.2}\right)^{0.2}\left(\frac{1000\pi}{0.4}\right)^{0.4}\left(\frac{10,000}{\pi\times 0.4}\right)^{0.4} = 9069.35.$$

From the relations, $f^* = U_1/\delta_1^* = U_2/\delta_2^* = U_3/\delta_3^*$, we have

$$U_1 = f^*\delta_1^* \Rightarrow 1000\pi x^2 = 9069.35\times 0.2 \Rightarrow x = 0.76\text{m}$$

$$U_2 = f^*\delta_2^* \Rightarrow 1000\pi xy = 9069.35\times 0.4 \Rightarrow y = 1.519\text{m}.$$

So the ideal dimension for the cylindrical vessel is a radius of 0.76m and a height of 1.519m. The total cost of transportation is ₹9069.35.

6.2.1 Arithmetic–geometric inequality

As the conditions for optimum values are the same as those for the arithmetic mean to be equal to the geometric mean, the above optimization method for posynomials is called geometric programming. The general arithmetic mean inequality (Cauchy's inequality) for any n non-negative numbers $x_1, x_2, \cdots, x_n$ can be written as

$$\frac{x_1 + x_2 + \cdots + x_n}{n} \geq (x_1 x_2 \cdots x_n)^{1/n}. \tag{6.27}$$

In Eq. (6.27), the equality sign holds good if and only if $x_1 = x_2 = \cdots = x_n$.

For example, if we take $x_1 = 2.0$, $x_2 = 3.0$, $x_3 = 4.0$, and $x_4 = 7.0$, the arithmetic mean is $(2+3+4+7)/4 = 4.0$ and geometric mean is $(2\times 3\times 4\times 7)^{1/4} = 3.6$.

But if we consider $x_1 = x_2 = x_3 = x_4 = 1$, the arithmetic mean is $4/4 = 1$ and geometric mean is $(1\times 1\times 1\times 1)^{1/4} = 1$.

If a general case is taken then we may note that out of n values of u_i, $i = 1, 2, \cdots, N$

m_1 have the value of u_1.

m_2 have the value of u_2.

$\vdots$

m_N have the value of u_N.

Then $n = m_1 + m_2 + \cdots + m_N$.

The inequality Eq. (6.27) is given by

$$\frac{m_1 u_1 + m_2 u_2 + \cdots + m_N u_N}{n} \geq \left(u_1^{m_1} u_2^{m_2} \cdots u_N^{m_N}\right)^{1/n}. \tag{6.28}$$

Taking $\delta_i = m_i/n$, the above inequality Eq. (6.28) is

$$\delta_1 u_1 + \delta_2 u_2 + \cdots + \delta_N u_N \geq u_1^{\delta_1} u_2^{\delta_2} \cdots u_N^{\delta_N}. \tag{6.29}$$

Substituting $U_i = \delta_i u_i$ we have

$$U_1 + U_2 + \cdots + U_N \geq \left(\frac{U_1}{\delta_1}\right)^{\delta_1} \left(\frac{U_2}{\delta_2}\right)^{\delta_2} \cdots \left(\frac{U_N}{\delta_N}\right)^{\delta_N}. \tag{6.30}$$

In the case of optimization of polynomials, we know L.H.S. of Eq. (6.30) represents objective function and

$$U_j = C_j \prod_{i=1}^{n} x_i^{a_{ij}} = C_j x_1^{a_{1j}} x_2^{a_{2j}} \cdots x_n^{a_{nj}}. \tag{6.31}$$

R.H.S. of Eq. (6.30) becomes

$$\left(\frac{C_1 x_1^{a_{11}} x_2^{a_{21}} \cdots x_n^{a_{n1}}}{\delta_1}\right)^{\delta_1} \left(\frac{C_2 x_1^{a_{12}} x_2^{a_{22}} \cdots x_n^{a_{n2}}}{\delta_2}\right)^{\delta_2} \cdots \left(\frac{C_N x_1^{a_{1N}} x_2^{a_{2N}} \cdots x_1^{a_{nN}}}{\delta_N}\right)^{\delta_N}$$

$$= \left(\frac{C_1}{\delta_1}\right)^{\delta_1} \left(\frac{C_2}{\delta_2}\right)^{\delta_2} \cdots \left(\frac{C_N}{\delta_N}\right)^{\delta_N} x_1^{\sum_{j=1}^{N} a_{1j}\delta_j} x_2^{\sum_{j=1}^{N} a_{2j}\delta_j} \cdots x_n^{\sum_{j=1}^{N} a_{nj}\delta_j}. \tag{6.32}$$

According to arithmetic–geometric inequality, Eq. (6.30) is equal to Eq. (6.32), if and only if $x_1 = x_2 = \cdots = x_n = 1$. Hence $\sum_{j=1}^{N} a_{ij}\delta_j = 0$ for $i = 1, 2, \cdots, n$. Therefore it has to satisfy orthogonality conditions.

Thus $m_1 + m_2 + \cdots + m_N = n$, that is

$$\frac{m_1}{n} + \frac{m_2}{n} + \cdots + \frac{m_N}{n} = 1,$$

$$\delta_1 + \delta_2 + \cdots + \delta_N = 1.$$

In other words, the normality condition should also be satisfied.

As such, the conditions for optimum values of δ (weights) for a posynomial are the same as the conditions for the arithmetic mean to be equal to the geometric mean. Hence this optimization technique is called geometric programming.

6.2.2 Primal–dual relationship and sufficiency conditions in the unconstrained case

In inequality Eq. (6.30), the right side is called the dual function, $\nu(\delta_1, \delta_2, \cdots, \delta_N)$. In compact form, the inequality Eq. (6.30) can be written as

$$f \geq \nu. \tag{6.33}$$

The fundamental result can be noted that the minimum of the dual function equals the maximum of the primal function. Also it can be said that the minimization of the dual function subject to the orthogonality and normality conditions is a sufficient condition for f, the primal function, to be a global maximum.

If f^* indicates the minimum of the primal function and ν^* denotes the maximum of the dual function, Eq. (6.33) states that

$$f \geq f^* \geq \nu^* \geq \nu \tag{6.34}$$

In this section, we prove that $f^* = \nu^*$ and f^* corresponds to the global minimum of $f(X)$. For the convenience of notation, let us denote the objective function $f(X)$ by x_0 and make the exponential transformation

$$e^{w_i} = x_i \text{ or } w_i = ln\, x_i, i = 0, 1, 2, \cdots, n, \tag{6.35}$$

where the variables w_i are unconstrained in sign. Define the new variable δ_j also termed weights as

$$\delta_j = \frac{U_j}{x_0} = \frac{c_j \prod_{i=1}^{n} x_i^{a_{ij}}}{x_0}, j = 1, 2, \cdots, N. \tag{6.36}$$

It can be observed that Eq. (6.36) is positive and satisfy the relation

$$\sum_{j=1}^{N} \delta_j = 1. \tag{6.37}$$

By considering logarithms on both sides of Eq. (6.36), we have

$$ln\, \delta_j = ln\, c_j + \prod_{i=1}^{N} a_{ij}\, ln\, x_i - ln\, x_0. \tag{6.38}$$

or

$$ln\frac{\delta_j}{c_j} = \sum_{i=1}^{N} a_{ij}w_i - w_0, j = 1, 2, \cdots, N. \tag{6.39}$$

Thus the original problem of minimizing $f(X)$ with no constraints can be replaced by one of the minimizing w_0 subject to the equality constraints given by Eqs. (6.37) and (6.39). The objective function x_0 is given by

$$x_0 = e^{w_0} = \sum_{j=1}^{N} c_j \prod_{i=1}^{n} e^{a_{ij}w_i} = \sum_{j=1}^{N} c_j e^{\sum_{i=1}^{n} a_{ij}w_i}. \tag{6.40}$$

Since the exponential function $e^{a_{ij}w_i}$ is convex for w_i, the objective function x_0, which is a positive combination of exponential functions, is also convex. Hence there is only one stationary point for x_0 and it must be the global minimum. The global minimum point of w_0 can be obtained by constructing the following Lagrangian function and finding its stationary point

$$L(w, \delta, \lambda) = w_0 + \lambda_0\left(\sum_{i=1}^{N} \delta_i - 1\right) + \sum_{j=1}^{N} \lambda_j\left(\sum_{i=1}^{n} a_{ij}w_i - w_0 - ln\frac{\delta_j}{c_j}\right), \tag{6.41}$$

where

$$w = \begin{Bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{Bmatrix}, \delta = \begin{Bmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_N \end{Bmatrix}, \lambda = \begin{Bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_N \end{Bmatrix}, \tag{6.42}$$

with λ denoting the vector of Lagrange multipliers. At the stationary point of L, we have

$$\frac{\partial L}{\partial w_i} = 0, \ i = 0, \ 1, \ 2, \cdots, \ n$$

$$\frac{\partial L}{\partial \delta_j} = 0, \ j = 1, \ 2, \ \cdots, \ N$$

$$\frac{\partial L}{\partial \lambda_i} = 0, \ \ i = 0, \ 1, \ 2, \ \cdots, \ N \tag{6.43}$$

Eq. (6.43) gives the following relations

$$1-\sum_{j=1}^{N}\lambda_j=0 \text{ or } \sum_{j=1}^{N}\lambda_j=1 \tag{6.44}$$

$$\sum_{j=1}^{N}\lambda_j a_{ij}=0,\ i=1,\ 2,\ \cdots,\ n \tag{6.45}$$

$$\lambda_0-\frac{\lambda_j}{\delta_j}=0 \text{ or } \lambda_0=\frac{\lambda_j}{\delta_j},\ j=1,\ 2,\ \cdots,\ n \tag{6.46}$$

$$\sum_{j=1}^{N}\delta_j-1=0 \text{ or } \sum_{j=1}^{N}\delta_j=1 \tag{6.47}$$

$$-ln\frac{\delta_j}{c_j}+\sum_{i=1}^{n}a_{ij}w_i-w_0=0,\ j=1,\ 2,\ \cdots,\ N \tag{6.48}$$

Eqs. (6.44), (6.46), and (6.47) give the relation

$$\sum_{j=1}^{N}\lambda_j=1=\sum_{j=1}^{N}\lambda_0\delta_j=\lambda_0\sum_{j=1}^{N}\delta_j=\lambda_0. \tag{6.49}$$

Thus the values of the Lagrange multipliers are given by

$$\lambda_j=\begin{cases}1, & \text{for } j=0\\ \delta_j, & \text{for } j=1,\ 2,\ \cdots,\ N\end{cases} \tag{6.50}$$

By substituting Eq. (6.50) into Eq. (6.51), we get

$$L(\delta,w)=-\sum_{j=1}^{N}\delta_j\, ln\frac{\delta_j}{c_j}+(1-w_0)\left(\sum_{j=1}^{N}\delta_j-1\right)+\sum_{i=1}^{n}w_i\left(\sum_{j=1}^{N}a_{ij}\delta_j\right). \tag{6.51}$$

The function given in Eq. (6.51) can be considered as the Lagrangian function corresponding to a new optimization problem whose objective function $\tilde{\nu}(\delta)$ is given by

$$\tilde{\nu}(\delta)=-\sum_{j=1}^{N}\delta_j\, ln\frac{\delta_j}{c_j}=ln\left[\prod_{j=1}^{N}\left(\frac{c_j}{\delta_j}\right)^{\delta_j}\right], \tag{6.52}$$

and the constraints are

$$\sum_{j=1}^{N} \delta_j - 1 = 0, \tag{6.53}$$

$$\sum_{j=1}^{N} a_{ij}\delta_j = 0, i = 1, 2, \cdots, n. \tag{6.54}$$

This problem will be the dual for the original problem. The quantity $(1 - w_0)$, w_1, w_2, $\cdots$, w_n can be regarded as the Lagrange multipliers for the constraints given by Eqs. (6.53) and (6.54).

Now it is evident that the vector δ which makes the Lagrangian of Eq. (6.51) stationary will automatically give a stationary point for Eq. (6.41). It can be proved that the function

$$\delta_j \, ln\frac{\delta_j}{c_j}, \; j = 1, \; 2, \; \cdots, \; N \tag{6.55}$$

is convex since δ_j is positive. Since the function $\tilde{\nu}(\delta)$ is given by the negative sum of convex functions, it will be a concave function. Hence the function $\tilde{\nu}(\delta)$ will have a unique stationary point which will be its global maximum point. Hence the minimum of the original primal function is the same as the maximum of the function given by Eq. (6.52) subject to the normality and orthogonality conditions given by Eqs. (6.53) and (6.54) with the variables δ_j constrained to be positive.

By substituting the optimal solution δ^*, the optimal value of the objective function becomes

$$\tilde{\nu} = \tilde{\nu}(\delta^*) = L(w^*, \delta^*) = w_0^* = L(w^*, \delta^*, \lambda^*) = -\sum_{j=1}^{N} \delta_j^* \, ln\frac{\delta^*}{c_j}. \tag{6.56}$$

By taking the exponentials and using the transformation relation Eq. (6.35), we get

$$f^* = \prod_{j=1}^{N} \left(\frac{c_j}{\delta_j^*}\right)^{\delta_j^*}. \tag{6.57}$$

6.2.3 Primal and dual problems

We saw that geometric programming treats the problem of minimizing posynomials and maximizing product functions. The minimization problems are called primal programs and the maximization problems are called dual programs.

6.2.4 Computational procedure

To solve a given unconstrained minimization problem, we construct the dual function $\nu(\delta)$ and maximize $\nu(\delta)$ or $ln\ \nu(\delta)$, whichever is convenient, subject to the constraints by Eqs. (6.53) and (6.54). If the degree of difficulty of the problem is zero, there will be a unique solution for the values of δ_j^*.

For problems with a degree of difficulty greater than zero, there will be more variables $\delta_j (j = 1, 2, \cdots, N)$ then the number of equations $(n + 1)$. Sometimes it will be possible for us to express any $(n + 1)$ number of δ_j in terms of the remaining $(N - n - 1)$ number of δ_j. In such cases our problem will be to maximize $\nu(\delta)$ or $ln\ \nu(\delta)$ for the $(N - n - 1)$ independent δ_j.

6.3 Constrained optimization

Most engineering optimization problems are subject to constraints. If the objective function and all the constraints are expressible in the form of posynomials, geometric programming can be used most conveniently to solve the optimization problem. Let the constrained minimization problem be stated as

Find $X = (x_1, x_2, \cdots, x_n)^T$, that minimizes the objective function

$$f(X) = \sum_{j=1}^{N_0} c_{0j} \prod_{i=1}^{n} x_i^{a_{0ij}} \tag{6.58}$$

and satisfies the constraints

$$g_k(X) = \sum_{j=1}^{N_k} c_{kj} \prod_{i=1}^{n} x_i^{a_{kij}} \leq 1,\ k = 1,\ 2,\ \cdots,\ m$$

or

$$g_k(X) = \sum_{j=1}^{N_k} c_{kj} \prod_{i=1}^{n} x_i^{a_{kij}} \leq 1,\ k = 1,\ 2,\ \cdots,\ m \tag{6.59}$$

where the coefficients c_{0j} ($j = 1, 2, \cdots, N_0$) and c_{kj} ($k = 1, 2, \cdots, m$; $j = 1, 2, \cdots, N_k$) are positive numbers, the exponents a_{0ij} ($i = 1, 2, \cdots, n$; $j = 1, 2, \cdots, N_0$) and a_{kij}($k = 1, 2, \cdots, m; i = 1, 2, \cdots, n; j = 1, 2, \cdots, N_k$) are real numbers, m indicates the total number of constraints, N_0 represents the number of terms in the objective function, and N_k denotes the number of terms in the kth constraint. The design variables $x_1, x_2, \cdots, x_n$ are assumed to take only positive values in Eqs. (6.58) and (6.59).

6.3.1 Solution of a constrained geometric programming problem

For simplicity of notation, let us denote the objective function as

$$x_0 = g_0(X) = f(X) = \sum_{i=1}^{N_0} c_{0j} \prod_{j=1}^{n} x_i^{a_{0ij}}. \tag{6.60}$$

The constraints given in Eq. (6.59) can be rewritten as

$$f_k = \sigma_k \left[1 - g_k(X)\right] \geq 0, \ k = 1, 2, \cdots, m, \tag{6.61}$$

where σ_k is the signum function that introduced for the kth constraint so that it takes on the value $+1$ or -1, depending on whether $g_k(X)$ is ≤ 1 or ≥ 1, respectively. The problem is to minimize the objective function, Eq. (6.60), subject to the inequality constraints given by Eq. (6.61). This problem is called the primal problem and can be replaced by an equivalent problem (known as the dual problem) with linear constraints, which is often easier to solve. The dual problem involves the maximization of the function, $\nu(\lambda)$, given by

$$\nu(\lambda) = \prod_{k=0}^{m} \prod_{j=1}^{N_k} \left(\frac{c_{kj}}{\lambda_{kj}} \sum_{l=1}^{N_k} \lambda_{kl} \right)^{\sigma_k \lambda_{kj}} \tag{6.62}$$

subject to the normality and orthogonality conditions

$$\sum_{j=1}^{N_0} \lambda_0 j = 1, \tag{6.63}$$

$$\sum_{k=0}^{m} \sum_{j=1}^{N_k} \sigma_k a_{kij} \lambda_{kj} = 0, \ i = 1, 2, \cdots, n. \tag{6.64}$$

If the problem has zero degrees of difficulty, the normality and orthogonality conditions Eqs. (6.63) and (6.64) yield a unique solution λ^* from which the stationary value of the original objective function can be obtained as

$$f^* = x_0^* = \nu(\lambda^*) = \prod_{k=0}^{m} \prod_{j=}^{N_k} \left(\frac{c_{kj}}{\lambda_{kj}^*} \sum_{l=1}^{N_k} \lambda_{kl}^* \right)^{\sigma_k \lambda_{kj}^*}. \tag{6.65}$$

If the function $f(X)$ is known to possess a minimum, the stationary value f^* given by Eq. (6.65) will be the global minimum of f since, in this case, there is a unique solution for λ^*.

The degree of difficulty of the problem D is defined as

$$D = N - n - 1, \tag{6.66}$$

where N denotes the total number of posynomial terms in the problem

$$N = \sum_{k=0}^{m} N_k. \tag{6.67}$$

If the problem has a positive degree of difficulty, the linear Eqs. (6.63) and (6.64) can be used to express any $(n+1)$ of the λ_{kj} in terms of the remaining D of the λ_{kj}. By using these relations, ν can be expressed as a function of the D independent λ_{kj}. Now the stationary points of ν can be found by using any of the unconstrained optimization techniques.

If calculus techniques are used, the first derivatives of the function ν for the independent dual variables are set as zero. This results in as many simultaneous nonlinear equations as there are degrees of difficulty (i.e., $N - n - 1$). The solution of these simultaneous nonlinear equations yields the best values of the dual variables λ^*. Hence this approach is occasionally impractical due to the computations required. However, if the set of nonlinear equations can be solved, geometric programming provides an elegant approach.

6.3.2 Optimum design variables

For problems with a zero degree of difficulty, the solution of λ^* is unique. Once the optimum values of λ_{kj} are obtained, the maximum of the dual function ν^* can be obtained from Eq. (6.65), which is also the minimum of the primal function, f^*. Once the optimum value of the objective function $f^* = x_0^*$ is known, the next step is to determine the values of the design variables x_i^* ($i = 1, 2, \cdots, n$). This can be achieved by simultaneously solving the following equations:

$$\delta_{0j}^* = \lambda_{0j}^* \equiv \frac{c_{0j} \prod_{i=1}^{n} \left(x_i^*\right)^{a0ij}}{x_0^*}, \; j = 1, 2, \cdots, N_0, \tag{6.68}$$

$$\delta_{kj}^* = \frac{\lambda_{kj}^*}{\sum_{l=1}^{N_k} \lambda_{kl}^*} = c_{kj} \prod_{i=1}^{n} \left(x_i^*\right)^{akij}, \; j = 1, 2, \cdots, N_k; \; k = 1, 2, \cdots, m. \tag{6.69}$$

6.3.3 Primal and dual programs in the case of less-than inequalities

If the original problem has a zero degree of difficulty, the minimum of the primal problem can be obtained by maximizing the corresponding dual function. Unfortunately this cannot be done in the general case where there are some greater than the type of inequality constraints. However, if the problem has all the constraints in the form of $g_k(X) \leq 1$, the signum σ_k are all equal to $+1$, and the objective function $g_0(X)$ will be a strictly convex function of the transformed variables $w_1, w_2, \cdots, w_n$, where

$$x_i = e^{wi}, \; i = 0, 1, 2, \cdots, n \tag{6.70}$$

In this case, the following primal—dual relationship can be valid

$$f(X) \geq f^* \equiv \nu^* \geq \nu(\lambda). \tag{6.71}$$

From the earlier discussion, the following characteristics can be observed.

1. The factors c_{kj} appearing in the dual function $\nu(\lambda)$ are the coefficients of the posynomials $g_k(X), \; k = 0, 1, 2, \cdots, m$.
2. The number of components in the vector λ is equal to the number of terms involved in the posynomials $g_0, g_1, \cdots, g_m$. Associated with every term in $g_k(X)$, there is a corresponding δ_{kj}.
3. Each factor $\left(\sum_{l=}^{N_k} \lambda_{kl}\right)^{\lambda_{kj}}$ of $\nu(\lambda)$ comes from an inequality constraint $g_k(X) \leq 1$. No such factor appears from the primal function $g_0(X)$ as the normality condition forces $\sum_{l=1}^{N_k} \lambda_{0j}$ to be unity.

4. The coefficient matrix $\left[a_{kij}\right]$ appearing in the orthogonality condition is the same as the exponent matrix appearing in the posynomials of the primal programs.

6.4 Geometric programming with mixed inequality constraints

In this case, the geometric problem contains at least one signum function with a value of $\sigma_k = -1$ among $k = 1, 2, \cdots, m$. (Note that $\sigma_0 = +1$ corresponds to the objective function.) Here no general statement can be made about the convexity or concavity of the constraint set. However, since the objective function is continuous and is bounded below by zero, it must have a constrained minimum with exist points satisfying the constraints.

Practice set

1. Minimize the following unconstrained geometric optimization problem $f = 7xy^{-1} + 3yz^{-2} + 5x^{-3}yz + xyz,\ x, y, z > 0.$
2. Minimize $f(X) = \frac{1}{2}x_1^{1/3}x_2^{-1/3}x_3 + \frac{2}{3}x_1^{1/2}x_2^{-2/3},\ x_1, x_2, x_3 > 0.$
3. Minimize $f(X) = 5x_1 + 20x_2 + \frac{10}{x_1x_2},\ x_1, x_2 > 0.$
4. Find the optimum value of the following function and the optimality
   $f(X) = 15x_1^{-1}x_2^{-1} + 10x_1x_2x_3^{-1} + 25x_2x_3 + x_1x_3,\ x_1, x_2, x_3 > 0.$
5. An open cylindrical vessel is to be constructed to transport 80m^3 of grain from a warehouse to a factory. The sheet metal used for the bottom and sides cost ₹800 and ₹100 per square meter, respectively. If it costs ₹10 for each round trip of the vessel, find the dimensions of the vessel for minimizing the transportation cost. Assume that the vessel has no salvage upon completion of the operation.
6. Minimize $f(X) = \frac{40}{x_1x_2x_3} + 40x_1x_3,$
   subject to
   $4x_1x_2 + 2x_2x_3 = 8,$
   $x_1, x_2, x_3 > 0.$

Further reading

Aggarwal, V., & O'Reilly, U. (2006). Design of posynomial models for MOSFETs: Symbolic regression using genetic algorithms. In R. Riolo, T. Soule, & B. Worzel (Eds.), *Genetic programming theory and practice. Genetic and evolutionary computation*. Ann Arbor: Springer. (Chapter 7).

Avriel, M., & Barrett, J. (1980). Optimal design of pitched laminated wood beams. In M. Avriel (Ed.), *Advances in geometric programming* (pp. 407–419). New York: Plenum Press.

Bhavikatti, S. S. (2010). *Fundamentals of optimum design in engineering*. New Delhi: New Age International (P) Limited Publishers.

Duffin, R., Peterson, E., & Zener, C. (1967). *Geometric programming—Theory and application*. New York: Wiley.

Federowicz, A., & Rajgopal, J. (1999). Robustness of posynomial geometric programming optima. *Mathematical Programming*, 421–431.

Maranas, C., & Floudas, C. (1997). Global optimization in generalized geometric programming. *Computers & Chemical Engineering*, 351–369.

Peterson, E. (2001). The origins of geometric programming. *Annals of Operations Research*, 15–19.

Rao, S. S. (1995). *Optimization theory and applications*. New Delhi: New Age International (P) Limited Publishers.

Salomone, H., & Iribarren, O. (1993). Posynomial modeling of batch plants: a procedure to include process decision variables. *Computers & Chemical Engineering*, 173–184.

Satish, N., Ravindran, K., Moskewicz, M., Chinnery, D., & Keutzer, K. (2005). *Evaluating the effectiveness of statistical gate sizing for power optimization*. Berkeley: University of California.

Shigley, J., & Mischke, C. (1989). *Mechanical engineering design*. New York: McGraw-Hill.

Wilde, D., & Beightler, C. (1967). *Foundations of optimization*. Englewood Cliffs: Prentice-Hall.

Zener, C. (1971). *Engineering design by geometric programming*. New York: Wiley.

This page intentionally left blank

CHAPTER SEVEN

Dynamic programming

Dynamic programming (DP) is an algorithmic approach for investigating an optimization problem by splitting into several simpler subproblems. It is noted that the overall problem depends on the optimal solution to its subproblems. Hence, the very essential feature of DP is the proper structuring of optimization problems into multiple levels, which are solved sequentially one level at a time. By using ordinary optimization problem techniques, each one level is solved, and its solution helps to define the characteristics of the next level problem in the sequence. Commonly, the levels represent different time periods in the outlook of the overall problem. Understand the basic concept of DP by using a very basic example of the Fibonacci series.

Example 7.1. Fibonacci series is a series of number, which starts with 0 and 1, where each number is the sum of the previous two numbers. The first few Fibonacci numbers are 0, 1, 1, 2, 3, 5, 8, 13, and 21. Now, if it asked to calculate *n*th Fibonacci Number, then we may calculate it by the following generalized formula:

$$\mathrm{Fib}(n) = \mathrm{Fib}(n-1) + \mathrm{Fib}(n-2), \text{ for } n > 1. \tag{7.1}$$

From Eq. (7.1), it is observed that to solve the overall problem (i.e., Fib(n)), we broke it down into two smaller subproblems (which are Fib($n-1$) and Fib($n-2$)). This shows that we can use DP to solve this problem.

Example 7.2. The problem of determining the level of inventory of a single commodity can be stated as a dynamic program.

Here, the decision variable is the amount to order at the beginning of each month; the objective is to minimize the total ordering and inventory-carrying costs; the basic constraint requires that the demand for the product be satisfied. If we can order only at the beginning of each month and we want an optimal ordering policy for the coming year, we could decompose the problem into 12 levels, each representing the ordering decision at the

© 2020 Elsevier Inc.
All rights reserved.

beginning of the corresponding month. Sometimes the levels do not have time implications. Problems that can be formulated as dynamic programs with levels that do not have time implications are often difficult to recognize.

7.1 Characteristics of dynamic programming

Before discussing different methods of solving a DP problem, first, take a look at what are the characteristics of a problem that tells us, we can apply DP to solve it.

1. Overlapping subproblems

   Subproblems are smaller versions of the original problem. Any problem has overlapping subproblems if finding its solution involves solving the same subproblem multiple times. Take the example of the Fibonacci numbers; to find the Fib (4), we need to break it down into the following subproblems given in Fig. 7.1.

   We can see the overlapping subproblem pattern here, as Fib(2) has been called twice and Fib(1) has been called three times.

2. Optimal substructure property

   Any problem has optimal substructure property if its overall optimal solution can be constructed from the optimal solutions of its subproblems. For Fibonacci numbers, as we know in Eq. (7.1), this clearly shows that a problem of size "*n*" has been reduced to subproblems of size "*n*-1" and "*n*-2." Therefore, Fibonacci numbers have

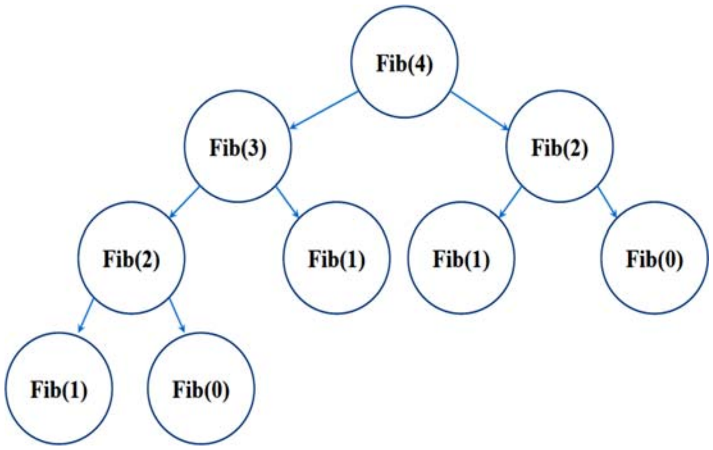


Figure 7.1 The network of Fibonacci numbers.

the optimal substructure property. DP methods offer two methods to solve the problem.

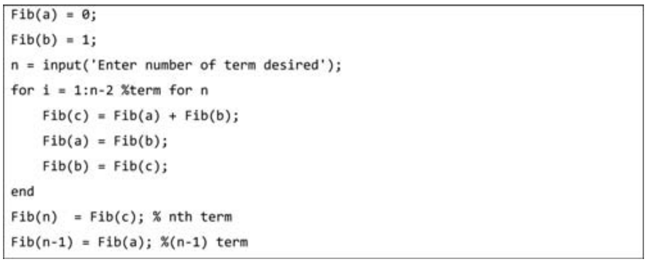
a. Top-down with Memoization: In this approach, we solve the bigger problem by recursively finding the solution to smaller subproblems. Whenever we solve a subproblem, we check its result so that we do not end up solving it repeatedly if it is called multiple times. Instead, we can just return the saved result. This technique of storing the results of already solved subproblems is called Memoization. We shall see this technique in our example of Fibonacci numbers. First, let us see the non-DP recursive solution for finding the *n*th Fibonacci number:

As we saw earlier, this problem shows the overlapping subproblems pattern, so make use of memoization here. We can use an array to store the already solved subproblems.

b. Bottom-up with Tabulation: This approach is the opposite of the top-down approach and avoids recursion. In this approach, we solve the problem "bottom-up" (i.e., by solving all the related subproblems first). This is typically done by filling up an *n*-dimensional table. Based on the results in the table, the solution to the top/original problem is then computed.

Tabulation is the opposite of Memoization, as in Memoization we solve the problem and maintain a map of already solved subproblems. In other words, in memoization, we do it top-down in the sense that we solve the top problem first (which typically recurses down to solve the subproblems). Let us apply tabulation to our example of Fibonacci numbers. Since we know that every Fibonacci number is the sum of the two preceding numbers, we can use this fact to populate our table. Here is the MATLAB code for our bottom-up DP approach:

MATLAB code



```
Fib(a) = 0;
Fib(b) = 1;
n = input('Enter number of term desired');
for i = 1:n-2 %term for n
    Fib(c) = Fib(a) + Fib(b);
    Fib(a) = Fib(b);
    Fib(b) = Fib(c);
end
Fib(n)  = Fib(c); % nth term
Fib(n-1) = Fib(a); %(n-1) term
```

Example 7.3. Consider a set S of numbers whose only prime factors are 2, 3, or 5. Then the sequence of first 11 numbers is 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, and 15. By convention, 1 is included. Given a number n, then find nth element of S.

Solution: This problem can be solved by splitting the sequence into the following three groups.

1. 1×2, 2×2, 3×2, 4×2, 5×2, ... (multiply sequence of natural numbers with 2 only)
2. 1×3, 2×3, 3×3, 4×3, 5×3, ... (multiply sequence of natural numbers with 3 only)
3. 1×5, 2×5, 3×5, 4×5, 5×5, ... (multiply sequence of natural numbers with 5 only)

We can observe that every subsequence is in set the S. Then we can merge and sort all the entries of these three sequences to get every element of S. Every step we choose the smallest one and move one step after.

7.2 Terminologies

Regardless of the type or size of a decision problem, certain terms and concepts are commonly used in a DP approach to solve such problems.

1. Level

The essential feature of the dynamic-programming approach is the structuring of optimization problems into multiple levels, which are solved sequentially one level at a time. Although each one-level problem is solved as an ordinary optimization problem, its solution helps to define the characteristics of the next one-level problem in the sequence. Often, the levels represent different periods in the problem's planning domain. For example, the problem of determining the level of inventory of a single commodity can be stated as a dynamic program. The decision variable is the amount to order at the beginning of each month; the objective is to minimize the total ordering and inventory-carrying costs; the basic constraint requires that the demand for the product be satisfied. If we can order only at the beginning of each month and we want an optimal ordering policy for the coming year, we could decompose the problem into 12 levels, each representing the ordering decision at the beginning of the corresponding month. Sometimes the levels do not have time implications. For example, the problem of determining the routes of a minimum delay from the homes of the commuters to the downtown parking lots is formulated as a dynamic program. The decision variable is whether to choose up or down in an intersection, and the levels of the process are defined to be the number of intersections to go. Problems that can be formulated as dynamic programs with levels that do not have time implications are often difficult to recognize.

2. States

States are connected with each level of the optimization process. It gives the information required to fully assess the consequences that the current decision has upon future actions. For example, in the inventory problem, each level has only one variable describing the state: the inventory level on hand of the single commodity. The minimum-delay problem also has one state variable: the intersection a commuter is in at a particular level. The specification of the states of the system is perhaps the most important design parameter of the DP model. There are no set rules for assigning states. In fact, for the most part, this is an art often requiring creativity and subtle insight about the problem being studied. The essential properties that can be useful in the selection of states are as follows:

a. The states should tell enough information to make future decisions without regard to how the process reached the current state.

b. The number of state variables should be small.

Since the computational effort associated with the DP approach is prohibitively expensive when there are more than two, or possibly three, state variables involved in the model formulation. This last feature considerably limits the applicability of DP in practice.

3. Recursive optimization

This is the final general characteristic of a DP process. By using recursive optimization, we build to a solution to the overall *N*-level problem by first solving a one-level problem. Then sequentially including one level at a time and solving one-level problems until the overall optimum has been found. This procedure can be based on a backward induction process, where the first level to be analyzed is the final level of the problem and problems are solved moving back one level at a time until all levels are included. Alternatively, the recursive procedure can be based on a forward induction process, where the first level to be solved is the initial level of the problem and problems are solved moving forward one level at a time until all levels are included. In certain problems, only one of these induction processes can be applied. The basis of the recursive optimization procedure is the so-called principle of optimality, which has already been stated: an optimal policy has the property that, whatever the current state and decision, the remaining decisions must constitute an optimal policy concerning the state resulting from the current decision.

A block diagram representation of the DP problem is shown in Fig. 7.2. Each block of the diagram of Fig. 7.2 represents a single-level decision problem (Fig. 7.3).

4. Return function

At each level, a decision is made that can affect the state of the system at the next level and help in arriving at the optimal solution at the current level. Every decision that is made has its own worth or benefit associated and can be described in an algebraic equation form called a return function. In general, this return function depends on both the state variable and the decision made at a particular level. An optimal policy or decision at a level yields optimal (maximum or minimum) return for a given value of the state variable.

For a multilevel decision process, the functional relationship among state, level, and decision may be described as shown in Fig. 7.4.

In this discussion,

n is the level number.

s_n is the state input to level n from level $n + 1$. Its value is the status of the system resulting from the previous $(n + 1)$ level decision.

d_n is the decision variable at level n (independent of previous levels). This represents the range of alternatives available when making a decision at level n.

$f_n = r_n(s_n, d_n)$ is the return (or objective) function for level n.

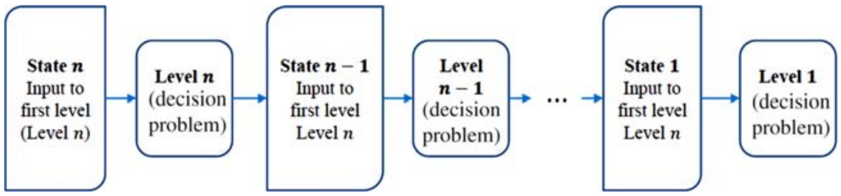

State n Input to first level (Level n)
Level n (decision problem)
State $n-1$ Input to first level Level n
Level $n-1$ (decision problem)
...
State 1 Input to first level Level n
Level 1 (decision problem)

Figure 7.2 Flow diagram of DP problem between levels.

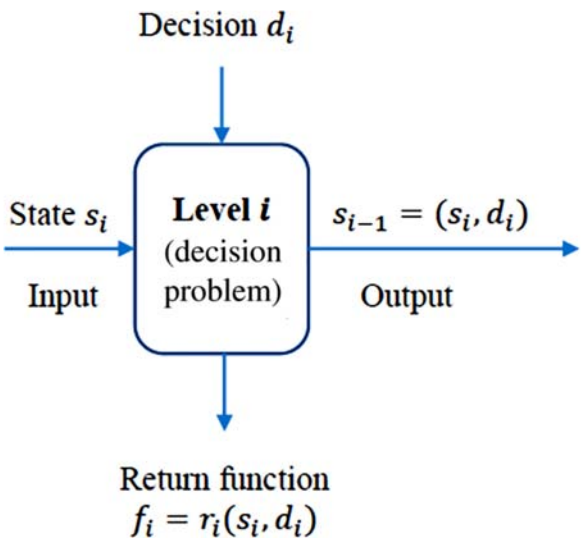


Figure 7.3 The *i*th level decision problem.

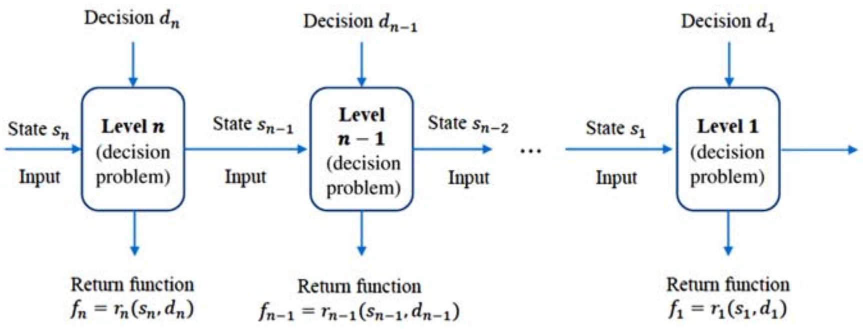

Decision d_n
Decision d_{n-1}
Decision d_1
State s_n
Input
Level n (decision problem)
State s_{n-1}
Input
Level $n-1$ (decision problem)
State s_{n-2}
Input
...
State s_1
Input
Level 1 (decision problem)
Return function $f_n = r_n(s_n, d_n)$
Return function $f_{n-1} = r_{n-1}(s_{n-1}, d_{n-1})$
Return function $f_1 = r_1(s_1, d_1)$

Figure 7.4 Multilevel decision problem.

Furthermore, suppose that there are n levels at which a decision is to be made. These n levels are all interconnected by a relationship (called transition function), that is,

$$\text{Output at level } n = (\text{Input to state } n) \times (\text{Decision at level } n) \tag{7.2}$$

$$s_{n-1} = s_n \times d_n \tag{7.2}$$

where $\times$ represents any mathematical operation, namely, addition, subtraction, division, or multiplication. The units of s_n, d_n, and s_{n-1} must be homogeneous.

It can be seen that at each level of the problem, there are two input variables such as state variable s_n and decision variable d_n. The state variable (state-input) relates the present level to the previous level. For example, the current state s_n provides complete information about various possible conditions in which the problem is to be solved when there are n stages to go. The decision d_n is made at level n for optimizing the total return over the remaining $n-1$ levels. The decision d_n which optimizes the output at level n, produces two outputs:

a. the return function $r_n(s_n, d_n)$; and

b. the new state variable s_{n-1}.

The return function is expressed as a function of the state variable s_n. The decision (variable) d_n indicates the state of the process at the beginning of the next level (level $n-1$) and is denoted by transition function (state transformation):

$$s_{n-1} = t_n(s_n, d_n) \tag{7.3}$$

where t_n represents a state transformation function and its form depends on the particular problem to be solved. This formula allows the transition from one level to another.

5. Condition of separability

Condition of separability means the objective function should be the composition of the individual levels. Thus

$$\begin{aligned} &f = \textstyle\sum_{i=1}^{N} f_i;\\ &\text{Or}\\ &f = \textstyle\prod_{i=1}^{N} f_i. \end{aligned} \tag{7.4}$$

Condition of monotonicity: This requirement of return function is satisfied if for all values of a and b that make,

$$r_i(s_i, d_i = a) \geq r_i(s_i, d_i = b). \tag{7.5}$$

7.3 Developing optimal decision policy

DP is an approach in which the problem is broken down into several smaller subproblems called levels. These subproblems are then solved sequentially until the original problem is finally solved. A particular sequence of alternatives (course of action) adopted by the decision-maker in a multilevel decision problem is called a policy. The optimal policy is, therefore, is the sequence of alternatives that achieves the decision-maker's objective. The solution of a DP problem is based on Bellman's principle of optimality (recursive optimization technique), which states:

The optimal policy must be one such that, regardless of how a particular state is reached, all later decision (choices) proceeding from that state must be optimal.

Based on this principle of optimality, an optimal policy is derived by solving one level at a time and then sequentially adding a series of one level problems that are solved until the optimal solution of the initial problem is obtained. The solution procedure is based on a backward induction process and forward induction process. In the first process, the problem is solved by solving the problem at the last level and working backward toward the first level, making optimal decisions at each level of the problem. In the forward process, it is used to solve a problem by first solving the initial level of the problem and working toward the last level, making an optimal decision at each level of the problem.

The exact recursion relationship depends on the nature of the problem to be solved by DP. The one-level return is given by

$$f_1 = r_1(s_1, d_1), \tag{7.6}$$

and the optimal value of f_1 under the state variable s_1 can be obtained by selecting a suitable decision variable d_1. That is,

$$f_1^{\text{opt}}(s_1) = \text{Opt.}\{r_1(s_1, d_1)\}. \tag{7.7}$$

The range of d_1 is determined by s_1, but s_1 is determined by what has happened in level 2. Then in level 2, the return function would take the form

$$f_2^{\text{opt}}(s_2) = \text{Opt.}\{r_2(s_2, d_2) \times f_1^{\text{opt}}(s_1)); \; s_1 = t_2(s_2, d_2). \tag{7.8}$$

By continuing the aforementioned logic recursively for a general n level problem, we have

$$f_n^{\text{opt}}(s_n) = \text{Opt.}\{r_n(s_n, d_n) \times f_{n-1}^{\text{opt}}(s_{n-1})); \; s_{n-1} = t_n(s_n, d_n). \tag{7.9}$$

Here, the symbol $\times$ denotes any mathematical relationship between s_n and d_n, including addition, subtraction, multiplication.

Algorithm 7.1.

Step 1: Identify the problem decision variables and specify the objective function to be optimized under certain limitations, if any.
Step 2: Decompose (or divide) the given problem into several smaller subproblems (or levels). Identify the state variables at each level and note the transformation function as a function of the state variable and decision variable at the next level.
Step 3: Note a general recursive relationship for computing the optimal policy. Decide whether to follow the forward or the backward method for solving the problem.
Step 4: Prepare appropriate tables to show the required values of the return function at each level.
Step 5: Determine the overall optimal policy or decisions and its value at each level. There may be more than one optimal policy.

Example 7.4. A salesman located in a Coimbatore city decided to travel to Hyderabad city. He knew the distances of alternative routes from Coimbatore to Hyderabad. He then drew a highway network map as shown in Fig. 7.5. The city of origin Coimbatore is city 1. The destination city Hyderabad is city 10. Other cities through which the salesman will have to pass through are numbered 2–9. The arrow representing routes between cities and distances in kilometers are indicated on each route. The salesman's problem to find the shortest route that covers all the selected cities from Coimbatore to Hyderabad.

Solution: To solve the problem we need to define the problem levels: decision variables, state variables, return function, and a transition

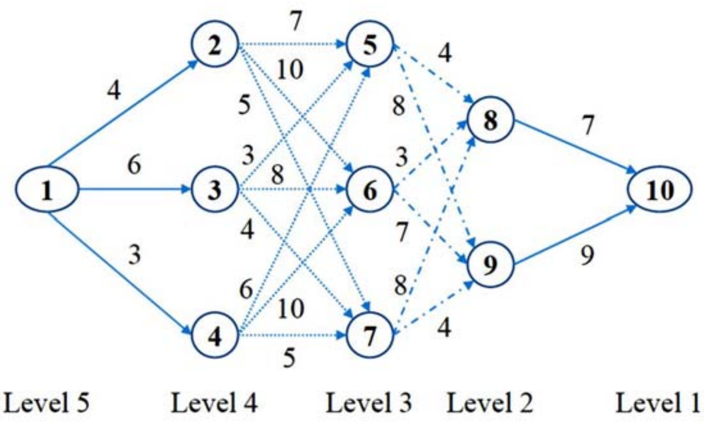


Figure 7.5 Network map.

function. For this particular problem, the following definitions will be used to denote various state and decision variables.

d_n = Decision variables that define the immediate destinations when there are n ($n = 1, 2, 3, 4$) levels to go.

s_n = State variables describe a specific city at any level.

$D(s_n, d_n)$ = Distance associated with the state variable s_n and the decision variables d_n for the current nth level.

$f_n(s_n, d_n)$ = Minimum total distance for the last n levels, given that the salesman is in state s_n and select d_n as an immediate destination.

$f_n^{\text{opt}}(s_n)$ = Optimal path (minimum distance) when the salesman is in state s_n with n more levels to go for reaching the final level (destination).

We start calculating distances between a pair of cities from destination city 10 and work backward level $L_5 \rightarrow L_4 \rightarrow L_3 \rightarrow L_2 \rightarrow L_1$ to find the optimal path. The recursion relationship for this problem can be stated as follows:

$$f_n^{\text{opt}}(s_n) = \text{Min}\left\{D(s_n, d_n), f_{n-1}^{\text{opt}}(d_n)\right); \; n = 1, 2, 3, 4$$

where $f_{n-1}^{\text{opt}}(d_n)$ is the optimal distance for the previous levels.

Working backward in levels from Hyderabad to Coimbatore, we determine the shortest distance to Hyderabad (node 10) in level 1, from state $s_1 = 8$ (node 8) and state $s_1 = 9$ (node 9) in level 2. Since the distances associated with entering level 2 from state $s_1 = 8$ and $s_1 = 9$ are $D_{8,10} = 7$ and $D_{9,10} = 9$, respectively, the optimal values of $f_1^{\text{opt}}(s_1)$ is the minimum value between $D_{8,10}$ and $D_{9,10}$. These results are presented in Table 7.1.

We move backward to level 3. Suppose that the salesman is at state $s_2 = 5$ (node 5). Here, the salesman has to decide whether he should go to either $d_2 = 8$ (node 8) or $d_2 = 9$ (node 9). For this, the salesman must evaluate two sums:

$$D_{(5,8)} + f_1^{\text{opt}}(8) = 4 + 7 = 11 \text{ (to state } s_1 = 8)$$

Table 7.1 Level 2 computation.

Decision d_1		$f_1(s_1, d_1) = D(s_1, d_1)$	$f_1^{\text{opt}}(s_1)$	**Optimal decision d_1**
		10		
State s_1	8	7	**7**	**10**
	9	9	9	10

$$D_{(5,9)} + f_1^{\text{opt}}(9) = 8 + 9 = 17 \text{ (to state } s_1 = 9)$$

This distance function for traveling from state $s_2 = 5$ is the smallest of these two sums

$$f_2(s_2) = \text{Min}\{11, 17) = 11 \text{ (to state } s_1 = 8)$$

Similarly, the calculation of distance function for traveling from state $s_2 = 6$ and $s_2 = 7$ can be completed as follows:

For state $s_2 = 6$

$$f_2(s_2) = \text{Min}\left\{ \begin{array}{l} D_{6,8} + f_1^{\text{opt}}(8) = 3 + 7 = 10 \\ D_{6,9} + f_1^{\text{opt}}(9) = 7 + 9 = 16 \end{array} \right\} = 10 \text{(to state } s_1 = 8)$$

For state $s_2 = 7$

$$f_2(s_2) = \text{Min}\left\{ \begin{array}{l} D_{7,8} + f_1^{\text{opt}}(8) = 8 + 7 = 15 \\ D_{7,9} + f_1^{\text{opt}}(8) = 4 + 9 = 13 \end{array} \right\} = 13 \text{(to state } s_1 = 9)$$

These results are entered into the two-level table as presented in Table 7.2.

The results that we obtain by continuing the same process for level 4 and 5 are presented in Tables 7.3 and 7.4.

The aforementioned optimal results at various levels can be summarized as follows:

Table 7.2 Level 3 computation.

		$f_1(s_1, d_1) = D(s_1, d_1)$			
Decision d_1		**8**	**9**	$f_1^{\text{opt}}(s_1)$	**Optimal decision** d_1
State s_2	5	11	17	11	8
	6	10	16	**10**	**8**
	7	15	13	13	9

Table 7.3 Level 4 computation.

		$f_1(s_1, d_1) = D(s_1, d_1)$				
Decision d_1		**5**	**6**	**7**	$f_1^{\text{opt}}(s_1)$	**Optimal decision** d_1
State s_3	2	18	20	18	18	5 or 7
	3	14	18	17	**14**	**5**
	4	17	20	18	17	5

Table 7.4 Level 5 computation.

Decision d_1		$f_1(s_1, d_1) = D(s_1, d_1)$ **2**	**3**	**4**	$f_1^{\text{opt}}(s_1)$	**Optimal decision** d_1
State s_4	1	22	20	20	**20**	**3** or **4**

Entering sates (nodes)

$$\text{Sequence or path}\begin{cases} 10 & 8 & 5 & 3 & 1 \\ 10 & 8 & 5 & 4 & 1 \end{cases}$$

$$\text{Distances}\begin{cases} 7 & 4 & 3 & 6 & = & 20 \\ 7 & 4 & 6 & 3 & = & 20 \end{cases}$$

Now it is clear that there are two alternatives shortest routes for this problem, both having a minimum distance of 20 units.

Example 7.5. Given n dice, each with m faces, numbered from 1 to m, find the number of ways to get sum X. X is the summation of values on each face when all the dice are thrown.

Solution: This problem can be efficiently solved using DP in the following process.

Let the function to find X from n dice is Sum (m, n, X). This function can be represented as follows:

$$\begin{aligned} \text{Sum}(m, n, X) &= \text{Finding Sum}(X\text{-}1) \text{ from } (n\text{-}1) \text{ dice plus 1 from } n\text{th dice} \\ &+ \text{Finding Sum}(X\text{-}2) \text{ from } (n\text{-}1) \text{ dice plus 2 from } n\text{th dice} \\ &+ \text{Finding Sum}(X\text{-}3) \text{ from } (n\text{-}1) \text{ dice plus 3 from } n\text{th dice} \\ &\vdots \\ &+ \text{Finding Sum}(X\text{-}m) \text{ from } (n\text{-}1) \text{ dice plus } m \text{ from } n\text{th dice} \end{aligned}$$

So, we may recursively write Sum (m, n, X) as follows:

$$\text{Sum } (m, n, X) = \text{Sum } (m, n-1, X-1) + \text{Sum } (m, n-1, X-2) + \cdots + \text{Sum } (m, n-1, X-m)$$

We can easily illustrate these results by the following consideration.

Let there are three dices, each with six faces and we need to find the number of ways to get sum 8.

$$\text{Sum } (6,3,8) = Sum\ (6,2,7) + Sum\ (6,2,6) + Sum\ (6,2,5) + Sum(6,2,4) + Sum\ (6,2,3) + Sum\ (6,2,2).$$

To evaluate Sum $(6, 3, 8)$, we need to evaluate Sum $(6, 2, 7)$, which can recursively be written as follows:

$$\begin{aligned}\mathrm{Sum}(6,2,7) = {} & \mathit{Sum}(6,1,6) + \mathit{Sum}(6,1,5) + \mathit{Sum}(6,1,4) + \mathit{Sum}(6,1,3) \\ & + \mathit{Sum}(6,1,2) + \mathit{Sum}(6,1,1)\end{aligned}$$

We also need to evaluate Sum $(6, 2, 6)$, which can recursively be written as follows:

$$\begin{aligned}&\mathrm{Sum}(6,2,6) = \mathbf{Sum}(6,1,5) + \mathbf{Sum}(6,1,4) + \mathbf{Sum}(6,1,3) \\ &\qquad\qquad + \mathbf{Sum}(6,1,2) + \mathbf{Sum}(6,1,1) \\ &\cdots \\ &\cdots \\ &\mathrm{Sum}(6,2,2) = \mathbf{Sum}(6,1,1)\end{aligned}$$

Here, the subproblems in the italicized font are solved the first time and subproblems in bold font are solved again. Hence, storing the results of the solved subproblems saves time.

7.4 Multiplicative separable return function and single additive constraint

Consider the general form of the recursive equation involving multiplicative separable return function and single additive constraints are follows.

$$\begin{aligned}&\text{Maximize } Z = \{f_1(d_1)\cdot f_2(d_2)\cdots f_n(d_n)\} \\ &\text{subject to} \\ &a_1d_1 + a_2d_2 + \cdots + a_n\mathrm{d_n} = b \\ &\text{and} \\ &d_j,\ a_j,\ b \geq 0\ \forall j = 1,\ 2,\ \cdots,\ n.\end{aligned} \tag{7.10}$$

where $j = j$th number of level $(j = 1,\ 2,\ \cdots,\ n) d_j =$ decision variable at jth level; $a_j =$ constant.

Defining state variables, $s_1,\ s_2,\ \cdots,\ s_n$ such that

$$\begin{aligned}&s_n = a_1d_1 + a_2d_2 + \cdots + a_nd_n = b \\ &s_{n-1} = a_1d_1 + a_2d_2 + \cdots + a_{n-1}d_{n-1} = s_n - a_nd_n \\ &\vdots \\ &s_{j-1} = s_j - a_jd_j \\ &\vdots \\ &s_1 = s_2 - a_2d_2\end{aligned} \tag{7.11}$$

In general, the state transition function takes the form

$$s_{j-1} = t_j(s_j, d_j);\ j = 1,\ 2,\ \cdots,\ n \tag{7.12}$$

that is a function of the next state and decision variables.

At the nth level, s_n is expressed as the function of the decision variables. Thus, the maximum value of Z, denoted by $f_n^{\text{opt}}(s_n)$ for any feasible value of s_n is given by

$$\begin{aligned} & f_n^{\text{opt}}(s_n) = \underset{d_j > 0}{\text{Max}} \left\{ f_1(d_1) \cdot f_2(d_2) \cdots f_n(d_n) \right\} \\ & \text{subject to the constraint} \\ & s_n = b. \end{aligned} \tag{7.13}$$

By keeping a particular value of d_n constant, the maximum value of Z is given by

$$\begin{aligned} & f_n(d_n) \times \underset{d_j > 0}{\text{Max}} \left\{ f_1(d_1) \cdot f_2(d_2) \cdots f_{n-1}(d_{n-1}) \right\} \\ & = f_n(d_n) \times f_{n-1}^{\text{opt}}(s_{n-1});\ j = 1,\ 2,\ \cdots,\ n-1. \end{aligned} \tag{7.14}$$

The maximum value $f_{n-1}^{\text{opt}}(d_{n-1})$ of Z due to decision variables $d_j (j = 1,\ 2,\ \cdots,\ n-1)$ depends on the state variables $s_{n-1} = t_n(s_n, d_n)$. The maximum of Z for any feasible value of all decision variables will be

$$\begin{aligned} & f_j^{\text{opt}}(s_j) = \underset{d_j > 0}{\text{Max}} \left\{ f_j(d_j) * f_{j-1}^{\text{opt}}(s_{j-1}) \right\};\ j = n,\ n-1,\ \cdots,\ 2 \\ & f_1(s_1) = f_1(d_1) \end{aligned} \tag{7.15}$$

where $s_{j-1} = t_j(s_j, d_j)$.

The value of $f_j^{\text{opt}}(s_j)$ represents the general recursive equation.

Example 7.6.

$$\begin{aligned} & \text{Maximize } Z = x \cdot y \cdot z \\ & \text{subject to} \\ & x + y + z = 10 \\ & \text{and} \\ & x, y, z \geq 0. \end{aligned}$$

Determine the value of x, y, and z.

Solution: Define the variable d_j $(j = 1, 2, 3)$, such that

$$\begin{aligned} & d_3 = x + y + z = 10,\ \text{at level 3} \\ & d_2 = d_3 - z = x + y,\ \text{at level 2} \\ & d_1 = d_2 - y = x \text{ at level 1.} \end{aligned}$$

The maximum value of Z for any feasible value of the state variable is given by

$$f_3(d_3) = \max_z \left\{ z \cdot f_2(d_2) \right\},$$

$$f_2(d_2) = \max_y \left\{ y \cdot f_1(d_1) \right\},$$

$$f_1(d_1) = x = d_2 - y,$$

Thus, $f_2(d_2) = \max_y \left\{ y \cdot (d_2 - y) \right\} = \max_y \left\{ y \cdot d_2 - y^2 \right\}$.

Differentiating $f_2(d_2)$ with respect to y and equating to zero (a necessary condition for a maximum or minimum value of a function), we have $d_2 - 2y = 0$ or $y = d_2/2$.

Now using Bellman's principle of optimality, we obtain

$$f_2(d_2) = \frac{d_2}{2} \cdot d_2 - \left(\frac{d_2}{2}\right)^2 = \frac{d_2^2}{4}, \text{ and}$$

$$f_3(d_3) = \max_z \left\{ z \cdot f_2(d_2) \right\} = \max_z \left\{ z \cdot \frac{d_2^2}{4} \right\} = \max_z \left\{ z \cdot \frac{(d_3 - z)^2}{4} \right\}.$$

Again, differentiating $f_3(d_3)$ with respect to z and equating to zero, we obtain

$$\frac{1}{4}\left\{ z \cdot 2(d_3 - z)(-1) + (d_3 - z)^2 \right\} = 0$$

$$(d_3 - z)(-2z + d_3 - z) = 0$$

$$(d_3 - z)(d_3 - 3z) = 0.$$

Now either $d_3 = z$, which is trivial as $x + y + z = d_3$ or $d_3 = 3z$, or $z = d_3/3 = 10/3$. Therefore,

$$y = \frac{d_2}{2} = \frac{d_3 - z}{2} = \frac{1}{2}\left(10 - \frac{10}{3}\right) = \frac{10}{3},$$

$$x = d_2 - y = \frac{20}{6} - \frac{10}{3} = \frac{10}{3}.$$

Thus, $x = y = z = \frac{10}{3}$, and hence $\mathrm{Max}(x \cdot y \cdot z) = \left(10/3\right)^3 = 1000/27$.

7.5 Additive separable return function and single additive constraint

Consider the problem

$$\begin{aligned}&\text{Minimize } Z = \left[f_1(d_1) + f_2(d_2) + \cdots + f_n(d_n)\right]\\ &\text{subject to the constraint}\\ &a_1 d_1 + a_2 d_2 + \cdots + a_n d_n \geq b, \qquad (7.16)\\ &\text{and}\\ &a_j,\ d_j,\ b \geq 0\ \forall j,\ j = 1,\ 2,\ \cdots,\ n.\end{aligned}$$

Proceed in the same manner as discussed in the previous section. Defining state variables $s_1, s_2, \cdots, s_n$, such that

$$\begin{aligned}&s_n = a_1 d_1 + a_2 d_2 + \cdots + a_n d_n \geq b;\\ &s_{n-1} = s_n - a_n d_n;\\ &\vdots \qquad (7.17)\\ &s_{j-1} = s_j - a_j d_j;\ j = 1,\ 2,\ \cdots,\ n\end{aligned}$$

Let $f_n^{\text{opt}}(s_n) = \min\limits_{d_j > 0} \sum\limits_{j=1}^{n} f_j(d_j)$, such that $s_n \geq b$.

The general recursive equation for obtaining the minimum value of Z, for all decision variables and any feasible value of all decision variables, is given by

$$f_j^{\text{opt}}(s_j) = \underset{d_j > 0}{\text{Min}} \left\{ f_j(d_j) \times f_{j-1}^{\text{opt}}(s_{j-1}) \right\};\ j = 2,\ 3,\ \cdots,\ n;$$

$$f_1^{\text{opt}}(s_1) = f_1(d_1); \qquad (7.18)$$

where $s_{j-1} = t_j(s_j, d_j)$.

Example 7.7. Use dynamic programming to solve the following problem:

$$\begin{aligned}&\text{Minimize } Z = x_1^2 + x_2^2 + x_3^2\\ &\text{subject to the constraint}\\ &x_1 + x_2 + x_3 \geq 15,\\ &\text{and}\\ &x_1, x_2, x_3 \geq 0.\end{aligned}$$

Solution: The given problem is a three-level problem and is defined as follows:

$$s_3 = x_1 + x_2 + x_3 \geq 15;$$

$$s_2 = x_1 + x_2 = s_3 - x_3;$$

$$s_1 = x_1 = s_2 - x_2.$$

The functional (recurrence) relation is expressed as follows:

$$f_1(s_1) = \min_{0 \leq x_1 \leq s_1} \left\{ x_1^2 = (s_2 - x_2)^2 \right\}$$

$$f_2(s_2) = \min_{0 \leq x_2 \leq s_2} \left\{ x_1^2 + x_2^2 \right\} = \min_{0 \leq x_2 \leq s_2} \left\{ x_2^2 + f_1(s_1) \right\}$$

$$f_3(s_3) = \min_{0 \leq x_3 \leq s_3} \left\{ x_1^2 + x_2^2 + x_3^2 \right\} = \min_{0 \leq x_3 \leq s_3} \left\{ x_3^2 + f_2(s_2) \right\}$$

Using the concept of maxima and minima in differential calculus, the minimum value of $f_2(s_2)$ can be obtained as follows.

Differentiating $f_2(s_2)$ with respect to x_2 and equating to zero, we have

$$2x_2 - 2(s_2 - x_2) = 0 \Rightarrow x_2 = \frac{s_2}{2}.$$

Thus, $f_2(s_2) = \min_{0 \leq x_2 \leq s_2} \left\{ x_2^2 + (s_2 - x_2)^2 \right\} = \left(\frac{s_2}{2}\right)^2 + \left(s_2 - \frac{s_2}{2}\right)^2 = \frac{s_2^2}{2};$

$$f_3(s_3) = \min_{0 \leq x_3 \leq s_3} \left\{ x_3^2 + f_2(s_2) \right\} = \min_{0 \leq x_3 \leq s_3} \left\{ x_3^2 + \frac{s_2^2}{2} \right\}$$

$$= \min_{0 \leq x_3 \leq s_3} \left\{ x_3^2 + \frac{(s_3 - x_3)^2}{2} \right\}$$

$$f_3(15) = \min_{0 \leq x_3 \leq s_3} \left\{ x_3^2 + \frac{(15 - x_3)^2}{2} \right\}, \text{ since } x_1 + x_2 + x_3 \geq 15.$$

Using differential calculus, we get the minimum value of the function $f_3(15) = x_3^2 + \frac{(15-x_3)^2}{2}$ at $x_3 = 5$. Hence, $f_3(15) = 5^2 + \frac{(15-5)^2}{2} = 75$.

Thus, $s_3 = 15$ or $x_3 = 5$; $s_2 = s_3 - x_3 = 10$ or $x_2 = \frac{s_2}{2} = 5$; $s_1 = s_2 - x_2 = 5$ or $x_1 = 5$.

The optimal policy is $x_1 = x_2 = x_3 = 5$ with $f_3(15) = 75$.

7.6 Additively separable return function and single multiplicative constraint

Consider the decision problem

$$\begin{aligned}
&\text{Minimize } Z = \left[f_1(d_1) + f_2(d_2) + \cdots + f_n(d_n)\right]\\
&\text{subject to the constraint}\\
&d_1 \cdot d_2 \cdot \cdots \cdot d_n \geq b,\\
&\text{and}\\
&d_j,\ b \geq 0\ \forall j,\ j = 1,\ 2,\ \cdots,\ n.
\end{aligned} \tag{7.19}$$

Proceed in the same manner as discussed in the previous section. Defining state variables $s_1,\ s_2,\ \cdots,\ s_n$, such that

$$\begin{aligned}
&s_n = d_n \cdot d_{n-1} \cdot \cdots \cdot d_2 \cdot d_1 \geq b;\\
&s_{n-1} = d_{n-1} \cdot d_{n-2} \cdot \cdots \cdot d_2 \cdot d_1 = \frac{s_n}{d_n};\\
&\vdots\\
&s_{j-1} = \frac{s_j}{d_j};\ j = 1,\ 2,\ \cdots,\ n
\end{aligned} \tag{7.20}$$

Let $f_n^{\text{opt}}(s_n) = \min\limits_{d_j > 0} \sum\limits_{j=1}^{n} f_j(d_j)$, such that $s_n \geq b$.

The general recursive equation for obtaining the minimum value of Z, for all decision variables and any feasible value of all decision variables, is given by

$$f_j^{\text{opt}}(s_j) = \underset{d_j > 0}{\text{Min}} \left\{f_j(d_j) + f_{j-1}^{\text{opt}}(s_{j-1})\right\};\ j = 2,\ 3,\ \cdots,\ n;$$

$$f_1^{\text{opt}}(s_1) = f_1(d_1); \tag{7.21}$$

where $s_{j-1} = t_j(s_j, d_j)$.

7.7 Dynamic programming approach for solving a linear programming problem

A linear programming (LP) problem in n decision variables and m constraints can be converted into an n level DP problem with m states. Consider a general LP problem:

$$\begin{aligned}
&\text{Maximize } Z = \textstyle\sum_{j=1}^{n} c_j d_j\\
&\text{subject to the constraints}\\
&\sum_{j=1}^{n} a_{ij} d_j \le b_i;\ i = 1,\ 2,\ \cdots,\ m\\
&\text{and}\\
&d_j \ge 0;\ j = 1,\ 2,\ \cdots,\ n.
\end{aligned} \tag{7.22}$$

To solve a LP problem using DP, the value of the decision variable d_j is determined at level $j(j = 1, 2, \cdots, n)$. The value of d_j at several levels can be obtained either by the forward or the backward induction method. The state variables at each level are the number of resources available for allocation to the current level and the succeeding levels.

Let $a_{1j},\ a_{2j},\ \cdots,\ a_{mj}$ be the amount (in units) of resources $i\,(i = 1,\ 2,\ \cdots,\ m)$, respectively, allocated to an activity c_j at jth level, and let $f_n(a_{1j},\ a_{2j},\ \cdots,\ a_{mj})$ be the optimum value of the objective function of a general LP problem for levels $j,\ j+1,\ \cdots,\ n$, and for states $a_{1j},\ a_{2j},\ \cdots,\ a_{mj}$. Thus, the LP problem may be defined by a sequence of functions as follows:

$$f_n(a_{1j},\ a_{2j},\ \cdots,\ a_{mj}) = \text{Max} \sum_{j=1}^{n} c_j d_j.$$

This maximization is taken over the decision variable x_j, such that

$$\textstyle\sum_{j=1}^{n} a_{ij} d_j \le b_i;\ x_j \ge 0.$$

The recursive relations for optimization are as follows:

$$\begin{aligned}
&f_n(a_{1j},\ a_{2j},\ \cdots,\ a_{mj}) \text{ or } f_n(b_1,\ b_2,\ \cdots,\ b_m)\\
&\quad = \max_{0 \le a_{in} x_n \le b_i} \left\{ c_n d_n + f_{n-1}(b_1 - a_{1n} d_n,\ b_2 - a_{2n} d_n,\ \cdots,\ b_m - a_{mn} d_n) \right\}
\end{aligned}$$

The maximum value b that a variable d_n can assume is given as follows:

$$b = \min\left\{ \frac{b_1}{a_{1n}},\ \frac{b_2}{a_{2n}},\ \cdots,\ \frac{b_m}{a_{mn}} \right\},$$

because the minimum value satisfies the set of constraints simultaneously.

7.8 Types of multilevel decision problem

Multilevel decision problems may be classified into the following three categories.

1. The problem in which initial input s_{i+1} is provided is called an initial value problem. This type of problem can be found in the water tank system problem where the capacity of the tank is given.
2. The problem in which the final output variable s_1 is provided is called the final value problem. This type of problem can be found in the design of a cantilever truss where maximum deflection is given.
3. The problem in which both initial input and final output are provided is called boundary value problem.

7.8.1 Concept of suboptimization and the principle of optimality

DP problem may be stated in the following way

Maximize/minimize

$$f(d_1, d_2, \cdots, d_N) = \sum_{i=1}^{N} r_i$$

which satisfy the level transformation equation,

$$s_i = t_i(s_{i+1}, d_i) \text{ for } i = 1,\ 2,\ \cdots,\ N$$

and find decision variables $d_1,\ d_2,\ \cdots,\ d_N$.

DP makes use of the concept of suboptimization and the principle of optimality in solving the problem. If an interior component in a serial system influences all the downstream components, it cannot be suboptimized without a consideration that it has on downstream. The last component in a serial structure influences no other component, and hence, it can be considered together as one large end component. Then that larger component can be suboptimized without the danger of disturbing the contribution of any other component. In fact, any number of end components

Table 7.5 Model table to find f_1^{opt} for a set of values of s_2.

s_2	d_1^{opt}	f_1^{opt}	s_1

Table 7.6 Model table to find f_2^{opt} corresponding to a range of input s_3.

s_3	d_2^{opt}	f_2^{opt}	s_2
⋮	⋮	⋮	⋮

can be systematically grouped in reverse sequential order and suboptimized. This suboptimization and optimal policy are followed in DP.

The steps involved in DP are as follows:

1. Arrange the various levels in the series and number them in the reverse order of the stream.
2. The last level that is numbered one is independent of all other levels. Hence, consider suboptimization or it to find

$$f_1^{\text{opt}} = \text{Opt. } r_1(d_1, s_2) \text{ and } d_1^{\text{opt}}.$$

Since the exact value is known only after upstream components are optimized, find f_1^{opt} for a set of values of s_2 and prepare the following model (Table 7.5).

3. Now take up suboptimization of levels 1 and 2.

   Since $f_2 = r_2 + r_1$,

$$\begin{aligned} f_2^{\text{opt}}(s_3) &= \text{Opt. } [r_2 + r_1] = \text{Opt. } r_2(d_2, s_3) + \text{Opt. } (d_1, s_2) \\ &= \text{Opt. } r_2(d_2 + s_3) + f_1^{\text{opt}}(d_1^{\text{opt}}, s_2). \end{aligned}$$

Thus, suboptimize level 2 to get d_2^{opt} and Opt. $r_2(d, s_3)$ find s_2 and add $f_1^{\text{opt}}(d, s_2)$. Here again s_3 is not a unique value, since upstream components are not yet optimized. Hence, find f_2^{opt} corresponding to a range of input s_3. Prepare the following model (Table 7.6).

4. Continue the aforementioned process to find $f_N^{\text{opt}} = \text{Opt. } R_N(d_N, s_{N+1}) + f_{N-1}^{\text{opt}}$ and d_N^*. Since s_{N+1} is the single input, it needs only one suboptimization.
5. Now start retracing in the downward direction to find.

$$\begin{array}{l} d_N^{\text{opt}},\ s_N,\ r_N^{\text{opt}} \text{ for } \cdots s_{N+1} \\ d_{N-1}^{\text{opt}},\ s_{N-1},\ r_{N-1}^{\text{opt}} \text{ for } \cdots s_N \\ \vdots \\ d_2^{\text{opt}},\ s_2,\ r_2^{\text{opt}} \text{ for } s_3 \\ d_1^{\text{opt}},\ s_1,\ r_1^{\text{opt}} \text{ for } s_2 \end{array}$$

Then $f^{\text{opt}}(d_N, d_{N-1}, \cdots, d_2, d_1, s_{N+1}) = r_N^{\text{opt}} + r_{N-1}^{\text{opt}} + \cdots + r_2^{\text{opt}} + r_1^{\text{opt}}$.

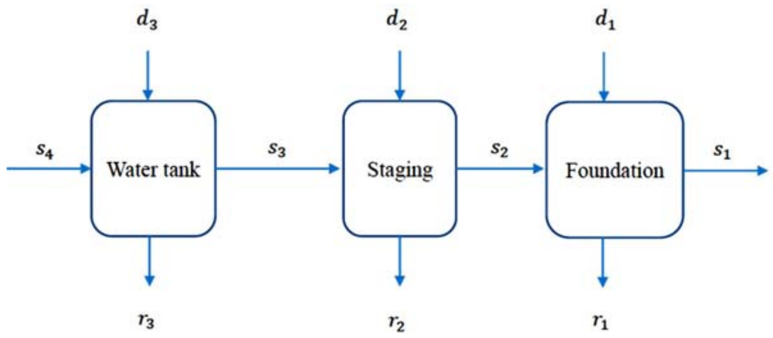


Figure 7.6 Multistage water tank problem.

7.8.2 Formulation of water tank optimization problem into a dynamic programming problem and the solution procedure

The following three levels are to be considered in the optimization.

1. Water tank
2. Staging
3. Foundation

The three levels are arranged as shown in Fig. 7.6. The optimum design is required for a specified value of the quantity of water. Hence initial input for weight (s_n) is known. Thus it is an initial value DP problem involving three-level decisions.

1. Decision variables

   It is possible to use any one type of water tank, decision-based on optimization. Different types of water tanks are as follows:

   A_1—RCC rectangular tank
   A_2—RCC cylindrical tank
   A_3—RCC Intz type tank
   A_4—Steel rectangular tank
   A_5—Steel cylindrical tank
   Here, RCC means reinforced cement concrete.
   Staging decision variables may be written as follows:
   B_1—RCC, 2 level bracings
   B_2—RCC, 3 level bracings
   B_3—Steel, 2 level bracings
   B_4—Steel, 3 level bracings
   Foundation decision variables are given as follows:
   C_1—RCC mat foundation
   C_2—RCC pile foundation
   C_3—Steel pile foundation

2. Input—output variables
   The input—output variables are as follows:
   s_4 is the weight of water
   s_3 is $s_4 +$ self-weight of tank
   s_2 is $s_3 +$ weight of footing
   Footing should be capable of transferring s_1 to ground safely.
3. Return functions
   The return functions are as follows:
   r_3—Cost of the water tank
   r_2—Cost of staging
   r_1—Cost of footing, and
   The total cost will be $r = r_1 + r_2 + r_3$.

Thus the problem is to select the type of water tank (d_1), type of staging (d_2), and type of footing (d_3) for a given capacity of the water tank, so that the cost of the system (r) is minimum.

7.8.3 Procedure

1. For the required capacity, water tanks of all types (A_1—A_2) should be designed (preferably optimum designs obtained) and the cost and weight of each obtained. Then the results are tabulated to show return function (r_3), self-weight, and hence output variable (s_3), which is the sum of the weight of water and that of the water tank.
2. Since self-weight of the tank is different for different types considered, the load transferred to staging (s_3) is different for each case. Design the staging of all types for a set of specific values of (s_3) and obtain cost (r_2), self-weight, and load transferred (s_2) to the foundation.
3. Now the range of (s_2) is known. Select a set of specific values of s_2 and design each type of foundations to get the cost (r_1) and self-weight. Arrange the results in a tabular form.
4. Now suboptimization can be taken up.
   a. Suboptimization of foundation: Looking at the table of foundation design, for the specific values of loads, decide which is the best type of foundation (d_1^{opt}) and note the corresponding cost (f_1^{opt})
   b. Suboptimization of staging and foundation: Now consider each specific weight on staging (s_3). Find the cost of each type for each s_3, which is obtained by adding the cost of staging (r_2) and the corresponding least cost of the foundation. If the least cost (f_1^{opt}) of the foundation is not available for the specific weight, it may be linearly interpolated and used. For each specific load (s_3), the

optimum decision variable of staging (d_2^{opt}) and corresponding least cost (f_2^{opt}) are noted.

c. Next, take up suboptimization of all the three levels: For each type of tank, find the total least cost by adding the cost of tank (r_3) to the corresponding least cost (f_2^{opt}). If for any case (f_2^{opt}) is not directly available, it may be linearly interpolated. After getting total least cost for each type of tank, select the least cost design (d_3^{opt} and f_3^{opt}).

5. Now retrace all tables to collect optimum values of d_3^{opt}, d_2^{opt}, and d_1^{opt}.

This procedure is illustrated with the following example.

Example 7.8. Design the most economical water tank system to store 1250kN of water. Use data of various types of water tank, staging, and foundation presented in Tables 7.7 and 7.8).

Table 7.7 Data about water tanks.

Designation	Type	Input load s_4 (in kN)	Cost ₹ (in lakhs)	Self-weight (in kN)	Output $s_3 = s_4 +$ self-weight
A_1	RCC rectangular	1250	3.4	600	1850
A_2	RCC cylindrical	1250	3.6	520	1770
A_3	RCC Intz	1250	4.5	560	1810
A_4	Steel rectangular	1250	4.8	280	1530
A_5	Steel circular	1250	5.2	220	1470

Table 7.8 Data about staging.

Designation	Type	Input load s_4 (in kN)	Cost ₹ (in lakhs)	Self-weight (in kN)	Output $s_3 = s_4 +$ self-weight
B_1	RCC with bracings at two levels	1450	1.2	400	1850
		1600	1.8	480	2080
		1750	2.4	560	2310
		1900	3.0	600	2500
B_2	RCC with bracings at three levels	1450	1.5	360	1810
		1600	1.9	420	2020
		1750	2.0	460	2210
		1900	2.2	480	2380
B_3	Steel with bracings at two levels	1450	0.8	200	1650
		1600	1.0	220	1820
		1750	1.2	240	1990
		1900	1.3	250	2150
B_4	Steel with bracings at three levels	1450	0.8	180	1630
		1600	0.9	200	1800
		1750	1.0	210	1960
		1900	1.1	220	2120

Solution: Optimization of the water tank system is considered as three-level decision problem as shown in Fig. 7.6. It is an initial value problem with initial input $s_4 = 1250kN$. As optimization of foundation does not affect the optimization of other components, it can be taken up first. From Table 7.9, we can easily prepare Table 7.10 giving optimum values of level 1 (f_1^{opt}) for the inputs 1600, 1900, 2200, and 2500 kN. Table 7.10 gives f_1^{opt} for various s_2.

Suboptimization of levels 2 and 1: Table 7.11 presents the suboptimization process of levels 2 and 1. In this cost of each system of staging, input weights $s_3 = 1450, 1600$, and 1900 are worked out. f_1^{opt} values are worked out by linearly interpolating the values from Table 7.10 corresponding to s_2 values d_1^{opt} are picked up, assuming it to be one which is nearer to s_2 values in Table 7.10.

(In Table 7.11, in the second row under the column s_2, the first entry is 1850 that lies between 1600 and 1900 of Table 7.10. Hence, the f_1^{opt} value is $0.40 - ((0.40 - 0.25)/(1900 - 1600)) \times 50 = 0.375$.)

Table 7.9 Data about the foundation.

Designation	Type	Input load s_4 (in kN)	Cost ₹ (in lakhs)	Self-weight (in kN)	Output $s_3 = s_4 +$ self-weight
C_1	Mat foundation	1600	0.25	500	2100
		1900	0.4	580	2480
		2200	0.8	680	2880
		2500	1.1	800	3300
C_2	RCC pile foundation	1600	0.4	400	2000
		1900	0.5	450	2350
		2200	0.75	540	2740
		2500	0.9	600	3100
C_3	Steel pile foundation	1600	0.5	200	1800
		1900	0.55	250	2150
		2200	0.70	300	2500
		2500	0.8	360	2860

Table 7.10 Suboptimization of level 1 Table 7.11.

s_2	d_1^{opt}	f_1^{opt}	s_1
1600	C_1	0.25	2100
1900	C_1	0.40	2480
2200	C_3	0.70	2500
2500	C_3	0.8	2860

Table 7.11 Suboptimization of levels 2 and 1.

s_3	d_2	R_2	s_2	d_1^{opt}	f_1^{opt}	$f_2 = R_2 + f_1^{opt}$
1450	B_1	1.2	1850	C_1	0.375	1.575
	B_2	1.5	1810	C_1	0.355	1.855
	B_3	0.8	1650	C_1	0.275	1.075
	B_4	0.8	1630	C_1	0.265	**1.065**
1600	B_1	1.8	2080	C_3	0.58	2.380
	B_2	1.9	2020	C_1	0.52	2.420
	B_3	1.0	1820	C_1	0.36	1.360
	B_4	0.9	1800	C_1	0.35	**1.250**
1750	B_1	2.4	2310	C_3	0.737	3.137
	B_2	2.0	2210	C_3	0.703	2.703
	B_3	1.2	1990	C_1	0.490	**1690**
	B_4	1.0	1960	C_1	0.460	2.460
1900	B_1	3.0	2500	C_3	0.8	3800
	B_2	2.2	2380	C_3	0.76	2960
	B_3	1.3	2150	C_1	0.65	1950
	B_4	1.1	2120	C_1	0.62	**1720**

Table 7.12 Optimum values for suboptimized of levels 2 and 1.

s_3	d_2^{opt}	f_2^{opt}	s_2
1450	B_4	1.065	1630
1600	B_4	1.25	1800
1750	B_3	1.690	1990
1900	B_4	1.720	2120

Table 7.13 Values of f_3.

s_2	d_3	R_3	s_3	d_2^{opt}	f_2^{opt}	$f_3 = R_2 + f_1^{opt}$
1250	A_1	3.4	1850	B_4	1.71	**5.11**
	A_2	3.6	1770	B_4	1.694	5.294
	A_3	4.5	1810	B_3	1.702	6.202
	A_4	4.8	1530	B_4	1.164	5.964
	A_5	5.2	1470	B_4	1.089	6.289

From Table 7.11, for each specified load s_3, the best values of f_2 are selected and noted in Table 7.12. It completes the suboptimization of levels 2 and 1.

Suboptimization of levels 3, 2, and 1: for this, Table 7.13 is to be assembled. There is only one input variable $s_4 = 1250kN$. For various variables d_3, R_3, and s_3 are available from Table 7.12. Looking at s_3 values, f_3 values are assembled.

From Table 7.13, it is clear that A_1 type tank is the best. Retracing Tables 7.10–7.13, we get the following best design.

$d_3^{\text{opt}} = A_1$ type tank $s_3 = 1850\text{kN}$

$d_2^{\text{opt}} = B_4$ type staging $s_2 = 2076.7\text{kN}$

$d_1^{\text{opt}} = C_2$ type foundation $s_1 = 2633.1\text{kN}$ and $f_3^{\text{opt}} = 5.11$ lakhs.

Hence, it is recommended for the RCC rectangular tank, steel staging with three-level bracing, and concrete pile foundation. The total cost will be ₹5.11 lakhs.

Practice set

1. Use DP to find the value of

$$\begin{aligned}&\text{Max } Z = x_1 \cdot x_2 \cdot x_3\\&\text{subject to the constraint}\\&x_1 + x_2 + x_3 = 5 \text{ and } x_1, x_2, x_3 \geq 0.\end{aligned}$$

2. Show how the functional equation technique of DP can be used to determine the shortest route when it is constrained to pass through a set of specified nodes, which is a definite subset of the set of nodes of a given network.

3. State Bellman's principle of optimality and apply it to solve the following problem

$$\begin{aligned}&\text{Max } Z = x_1 \cdot x_2 \cdot \cdots \cdot x_n\\&\text{subject to the constraints}\\&x_1 + x_2 + \ldots + x_n = c, \text{ and } x_1, x_2, \ldots, x_n \geq 0.\end{aligned}$$

4. Use Bellman's principle of optimality to solve the problem

$$\begin{aligned}&\text{Min } Z = x_1 + x_2 + \ldots + x_n\\&\text{subject to the constraints}\\&x_1 \cdot x_2 \cdot \cdots \cdot x_n = d;\\&x_1, x_2, \cdots, x_n \geq 0.\end{aligned}$$

5. Solve the following problem using DP.

$$\begin{aligned}&\text{Max } Z = x_1^2 + x_2^2 + x_3^2\\&\text{subject to the constraints}\\&x_1 . x_2 . x_3 \leq 4;\\&x_1, x_2, x_3 \geq 0.\end{aligned}$$

6. Use the principle of optimality to solve the problem

$$\begin{aligned}&\text{Min } Z = \textstyle\sum_{j=1}^{n} x_j^{\alpha}\\ &\text{subject to the constraints}\\ &x_1 \cdot x_2 \cdot \ldots \cdot x_n = r, \text{ and}\\ &x_j \geq 0; j = 1, 2, \ldots, n;\ r \geq 1, \alpha > 0.\end{aligned}$$

7. Solve the following LP problem by the DP approach

a. Max Z $= 3x_1 + 4x_2$

Z $= 3x_1 + 4x_2$

subject to the constraints
$2x_1 + x_2 \leq 40$,
$2x_1 + 5x_2 \leq 180$, and
$x_1, x_2 \geq 0$.

b. Max $Z = 2x_1 + 5x_2$
subject to the constraints
$2x_1 + x_2 \leq 43$,
$2x_2 \leq 46$, and
$x_1, x_2 \geq 0$.

c. Max $Z = x_1 + 9x_2$
subject to the constraints
$2x_1 + x_2 \leq 25$,
$x_2 \leq 11$, and
$x_1, x_2 \geq 0$.

d. Max $Z = 3x_1 + 7x_2$
subject to the constraints
$x_1 + 4x_2 \leq 8$,
$x_2 \leq 2$, and
$x_1, x_2 \geq 0$.

e. Max $Z = 3x_1 + 5x_2$
subject to the constraints
$x_1 \leq 4$,
$x_2 \leq 6$,
$3x_1 + 2x_2 \leq 18$, and
$x_1, x_2 \geq 0$.

f. Max $Z = 50x_1 + 100x_2$
subject to the constraints
$2x_1 + 3x_2 \leq 48$,
$x_1 + 3x_2 \leq 42$,
$x_1 + x_2 \leq 21$, and
$x_1, x_2 \geq 0$.

8. A truck can carry a total of 10 tons of a product. Three types of products are available for shipment. Their weights and values are tabulated. Assuming that at least one of each type must be shipped, determine the type of loading that will maximize the total value.

Product type	Value (in ₹)	Weight (in tons)
A	20	1
B	50	2
C	60	3

10. Use the principle of optimality to find the maximum value of

$$\text{Max } Z = b_1x_1 + b_2x_2 + \ldots + b_nx_n$$

subject to the constraint

$$x_1 + x_2 + \ldots + x_n = c;$$

$$x_1, x_2, \ldots, x_n \geq 0.$$

11. A student has to take an examination in three courses x, y, and z. He has 3 days for studying. He feels that it would be better to devote a whole day to study one single course. So he may study a course for 1 day, 2 days, or 3 days or not at all. His estimates of grades he may get according to days of study he puts in are as follows:

Study days	Course		
	x	y	z
0	1	2	1
1	2	2	2
2	2	4	4
3	4	5	4

How many days should he allocate to each course so that he gets the best results.

12. A chairman of a certain political party is planning his election to the Parliament. He has engaged the services of six volunteer workers and wishes to assign them four districts in such a way as to maximize their effectiveness. He feels that it would be inefficient to assign one worker to more than one district but he is also willing to assign no worker to any one of the districts, judging by what the workers can accomplish in other districts.

Further reading

Aguilar, R. (1973). Systems analysis and design in engineering, architecture, *construction*. Englewood Cliffs, NJ: Prentice-Hall.

Ayiagari, R. S. (1994). Uninsured idiosyncratic risk and aggregate saving. *Quarterly*, 659−684.

Gero, J., Sheehan, P., & Becker, J. (1978). Building design using feedforward nonserial. *Engineering Optimization*, 183−192.

Kirsch, U. (1981). Optimum structural design: Concepts*, methods, and applications*. New York, NY: McGraw-Hill.

Rao, S., & Das, G. (1984). Reliability-based optimum design of gear trains. *ASME Journal*, 17−22.

Rao, S. S. (1995). *Optimization theory and applications*. New Delhi: New Age International (P) Limited Publishers.

Sharma, J. K. (2006). *Operations research theory & applications*. New Delhi: Macmillan India Ltd.

Taha, H. A. (1992). *Operations research: An introduction*. New York, NY: Macmillan.

This page intentionally left blank

CHAPTER EIGHT

Integer programming

8.1 Integer linear programming

Integer linear programming (ILP) is a special type of linear programming (LP) problem where each variable, viz., decision variable and slack and/or surplus variable, is allowed to take any discrete or fractional value. However, there is an example of a problem in which the fractional value of the decision variables has no significance. For example, there is no sense to say that 6.5 men will be working on a project, 3.2 machines are working in a workshop. On the other hand, in the problems such as financial problems, capital budgeting, construction management, scheduling, plant location and size, routing and shipping schedule, batch size, capacity expansion, fixed charge, and so on, integer programming are used to quantify the involved variables. But, it needs to classify the types of integer programming to have a clear idea of the problem identification and methodology to be used.

8.1.1 Types of integer programming problems

Linear integer programming problems can be classified into the following three categories:

1. Pure (all) integer programming problems in which all decision variables are restricted to integer values.
2. Mixed-integer programming problems in which some, but not all, of the decision variables, are restricted to integer values.
3. Zero-one integer programming problems in which all decision variables are restricted to integer values of either 0 or 1.

The pure ILP problems in its standard form can be stated as follows:

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \tag{8.1}$$

subject to the constraints

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2$$
$$\vdots$$
$$a_{m1}x_1 + a_m x_2 + \cdots + a_{mn}x_n = b_m \tag{8.2}$$

and

$$x_1,\ x_2,\ \cdots,\ x_n \geq 0 \text{ and are integers.} \tag{8.3}$$

In this section, we discuss mainly the following two types of methods for solving integer programming problems.

1. Gomory's cutting plane method.
2. Branch and bound method.

8.1.2 Enumeration and concept of cutting plane solution

In 1956, R. E. Gomory developed a cutting plane method to solve ILP problems. The idea of this method is based on developing a sequence of linear inequalities, which is known as cuts. Then these cuts are used to reduce a part of the feasible region and give a feasible region of the ILP problem. The hyperplane boundary of a cut is called the cutting plane. The following example problem will be useful to understand this concept in detail.

Example 8.1. Optimize the linear integer programming problem

$$\text{Maximize } Z = 14x_1 + 16x_2$$

subject to the constraints

$$4x_1 + 3x_2 \leq 12$$
$$6x_1 + 8x_2 \leq 24$$

and integers $x_1, x_2 \geq 0$.

Solution: Ignoring the integer requirement, this problem can be solved graphically, and for easy demonstration, the solution is shown in Fig. 8.1. The optimal solution to this LP problem occurs at B, that is, $x_1 = 1.71,\ x_2 = 1.71$, and the maximum value of Z is 51.42. As both x_1 and x_2 are not integers, this solution does not satisfy the integer requirement of variables x_1 and x_2. Hence, this is not the required solution.

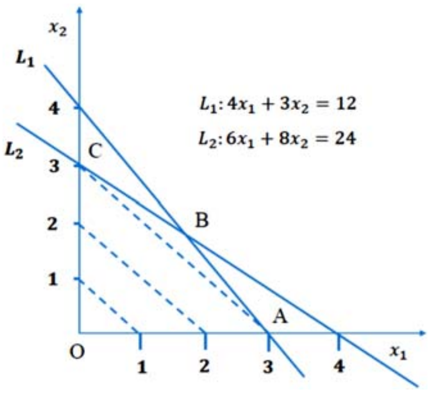


Figure 8.1 Graphical solution of Example 8.1.

The first approach we may do that is rounding off this solution to $x_1 = 2$ and $x_2 = 2$. But it does not satisfy both the constraints, and therefore the solution is infeasible. So, we reject this solution. Next, we can construct lattice points, which represent all of the integer solutions that lie within the feasible solution space of this LP problem. However, it is difficult to evaluate every such point to determine the value of the objective function.

In Fig. 8.1, it may be noted that the optimal lattice point C lies at the corner of the solution space $OABC$, which is obtained by cutting away the small portion above the dashed line. This suggests a solution procedure that successively reduces the feasible solution space until an integer-valued corner is obtained. The optimal integer solution is $x_1 = 0$ and $x_2 = 3$, and the corresponding maximum value of Z is 48. The lattice point C is not even adjacent to the most desirable LP problem solution corner B.

Remark. Reducing the feasible region by adding extra constraints (cut) can never give an improved objective function value. If Z_{IP} represents the maximum value of objective function in the ILP problem and Z_{LP} is the maximum value of the objective function in a LP problem, then $Z_{IP} \leq Z_{LP}$.

8.1.3 Gomory's all integer cutting plane method

In this section, a procedure called Gomory's all integer algorithm will be discussed for generating "cuts" (additional linear constraints) so as to ensure an integer solution to the given LP problem in a finite number of steps. Gomory's algorithm has the following properties.

1. Additional linear constraints never cut off that portion of the original feasible solution space that contains a feasible integer solution to the original problem.
2. Each new additional constraint (or hyperplane) cuts off the current noninteger optimal solution to the LP problem.

8.1.3.1 Method for constructing additional constraint (cut)

Gomory's method begins by solving a LP problem ignoring the integer value requirement of the decision variables. If the solution so obtained is an integer, that is, all variables in the "x_B"-column (also called basis) of the simplex table assume nonnegative integer values, the current solution is the optimal solution to the given ILP problem. However, if some of the basic variables do not have a nonnegative integer value, an additional linear constraint called the Gomory constraint (or cut) is generated. After having generated a linear constraint (or cutting plane), it is added to the bottom of the optimal simplex table. The new problem is then solved by using the dual simplex method. If the optimal solution, so obtained, is again a noninteger, then another cutting plane is generated. The procedure is repeated until all basic variables assume nonnegative integer values.

8.1.3.2 Procedure

In the optimal solution simplex table, select a row called source row for which the basic variable is noninteger. Then to develop a "cut," consider only fractional part of the coefficients in a source row. Such a cut is also referred to as a fractional cut.

Suppose the basic variable x_r has the largest fractional value among all basic variables required to assume integer value. Then the rth constraint equation (row) from the simplex table can be rewritten as follows:

$$x_{Br}(=b_r) = 1.x_r + (a_{r1}x_1 + a_{r2}x_2 + \cdots) = x_r + \sum_{j \neq r} a_{rj}x_j \tag{8.4}$$

where x_j ($j = 1,2,3\ldots$) represents all the nonbasic variables in the rth constraint (row), except the variables x_r, and $b_r(=x_{Br})$ is the noninteger value of the variable x_r.

For example, consider the following optimal solution of a LP problem (Table 8.1).

Table 8.1 The optimal solution of a test linear programming problem.

	c_j	1	1	0	0		
c_B	**Basis**	x_1	x_2	s_1	s_2	b	θ
1	x_1	1	0	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{1}{3}$	
1	x_2	0	1	0	1	2	
	$Z_j = \sum c_B a_{ij}$	1	1	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{7}{3}$	
	$C_j = c_j - Z_j$	0	0	$-\frac{1}{3}$	$\frac{2}{3}$		

In Table 8.1, x_1 is the only basic variable whose value is a nonnegative fractional value, and therefore take the first row (x_1 row) as a source row to generate the following Gomory cut.

$$\frac{1}{3} = 1 \times x_1 + 0 \times x_2 + \left(0 + \frac{1}{3}\right) \times s_1 + \left(-1 + \frac{1}{3}\right) \times s_2. \tag{8.5}$$

Decompose the coefficients of variables x_j and x_r as well as x_{Br} into integer and nonnegative fractional parts in Eq. (8.4) as follows:

$$[x_{Br}] + f_r = (1+0)x_r + \sum_{j \neq i} \left\{ \left[a_{rj}\right] + f_{rj} \right\} x_j, \tag{8.6}$$

where $[x_{Br}]$ and $\left[a_{rj}\right]$ denote the largest integer value obtained by truncating the fractional part from x_{Br} and a_{rj}, respectively.

Hence, Eq. (8.5) becomes

$$\left(0 + \frac{1}{3}\right) = (1+0) \times x_1 + \left(0 + \frac{1}{3}\right) \times s_1 + \left(-1 + \frac{1}{3}\right) \times s_2. \tag{8.7}$$

Rearranging Eq. (8.6) so that all the integer coefficients appear on the left-hand side, we obtain

$$f_r + \left\{ [x_{Br}] - x_r - \sum_{j \neq r} \left[a_{rj}\right] x_j \right\} = \sum_{j \neq r} f_{rj} x_j, \tag{8.8}$$

where f_r is strictly a positive fraction $(0 < f_r < 1)$, while f_{rj} is a nonnegative fraction $\left(0 \leq f_{rj} \leq 1\right)$.

Eq. (8.7) can be written as follows:

$$\frac{1}{3} + (s_2 - x_1) = \frac{1}{3}s_1 + \frac{1}{3}s_2. \tag{8.9}$$

Since all the variables (including slacks) are required to assume integer values, the terms in the bracket on the left-hand side and on the right-hand side must be nonnegative numbers. Since the left-hand side in Eq. (8.8) is f_r plus a nonnegative number, we may write in the form of the following inequalities.

$$f_r \leq \sum_{j \neq r} f_{rj} x_j \tag{8.10}$$

$$\sum_{j \neq r} f_{rj} x_j = f_r + s_g \text{or} - f_r = s_g - \sum_{j \neq r} f_{rj} x_j, \tag{8.11}$$

where s_g is a nonnegative slack variable and is also called Gomory slack variable.

Eq. (8.11) represents Gomory's cutting plane constraint. When this new constraint is added to the bottom of the optimal solution simplex table, it would create an additional row in the table, along with a column for the new variable s_g.

8.1.3.3 Steps of Gomory's all integer programming algorithm

An iterative procedure for the solution of an all integer programming problem by Gomory's cutting plane method can be summarized in the following steps.

Algorithm 8.1.

Step 1: Initialization

Formulate the standard integer LP problem. If there are any noninteger coefficients in the constraint equations, convert them into integer coefficients. Solve the problem by the simplex method, ignoring the integer value requirement of the variables.

Step 2: Test the optimality

1. Examine the optimal solution. If all basic variables (i.e., $x_{Bi} = b_i \geq 0$) have integer values, then the integer optimal solution has been obtained and the procedure is terminated.

2. If one or more basic variables with integer value requirement have non-integer solution values, then go to Step 3.

Step 3: Generate a cutting plane

Choose a row r corresponding to a variable x_r that has the largest fractional value f_r and follow the procedure to develop a "cut" (a Gomory constraint) as explained in Eq. (8.11). If there are more than one variable with the same largest fraction, then choose the one that has the smallest profit/unit coefficient in the objective function of the maximization LP problem or the largest cost/unit coefficient in the objective function of minimization LP problem.

Step 4: Obtain the new solutionAdd the additional constraint (cut) generated in Step 3 to the bottom of the optimal simplex table. Find a new optimal solution by using the dual simplex method, that is, choose a variable that is to be entered into the new solution having the smallest ratio: $\left\{\frac{(c_j - z_j)}{y_{ij}} : y_{ij} < 0\right\}$ and return to Step 2.

The process is repeated until all basic variables with integer value requirement assume nonnegative integer values.

Example 8.2. Find the integer solution for the following ILP problem using the cutting plane method.

$$\text{Maximize } Z = 2x + 20y - 10z \tag{8.12}$$

subject to the constraints

$$2x + 20y + 4z \leq 15$$

$$5x + 20y + 4z = 20$$

$$\text{and integers } x,\ y,\ z \geq 0. \tag{8.13}$$

Solution: Adding a slack variable s_1 in the first constraint and artificial variable in the second constraint, the LP problem is stated in the standard problem as follows:

$$\text{Maximize } Z = 2x + 20y - 10z + 0s_1 - MA_1 \tag{8.14}$$

subject to the constraints

$$2x + 20y + 4z + s_1 = 15$$

$$5x + 20y + 4z + A_1 = 20$$

$$\text{and integers } \ x,\ y,\ z,\ s_1,\ A_1 \geq 0. \tag{8.15}$$

The optimal solution of the LP problem, ignoring the integer value requirement using the simplex method, is presented in Table 8.2.

The noninteger solution presented in Table 8.2 is $x = \frac{5}{4}$, $y = \frac{5}{8}$, and $z = 0$, and the corresponding optimal solution is Max $Z = 15$.

To obtain an optimal solution satisfying integer value requirement, we proceed to construct Gomory's constraint. In this solution, the value of both basic variables x and y are noninteger. Since the fractional part of the value of basic variable $y = \left(0 + \frac{5}{8}\right)$ is more than that of the basic variable

Table 8.2 The optimal solution of the linear programming problem Example 8.2.

	c_j	2	20	-10	0		
c_B	**Basis**	x	y	z	s_1	b	θ
20	y	0	1	$\frac{1}{5}$	$\frac{3}{40}$	$\frac{5}{8}$	
2	x	1	0	0	$-\frac{1}{4}$	$\frac{5}{4}$	
	$Z_j = \sum c_B a_{ij}$	2	20	4	1	15	
	$C_j = c_j - Z_j$	0	0	-14	-1		

$x = \left(1 + \frac{1}{4}\right)$, the y row is selected for constructing Gomory cut as follows:

$$\frac{5}{8} = 0 \times x + y + \frac{1}{5}z + \frac{3}{40}s_1. \tag{8.16}$$

The factoring of the y row gives

$$\left(0 + \frac{5}{8}\right) = (1 + 0)y + \left(0 + \frac{1}{5}\right)z + \left(0 + \frac{3}{40}\right)s_1$$

$$\Rightarrow \frac{5}{8} - y = \frac{1}{5}z + \frac{3}{40}s_1. \tag{8.17}$$

Furthermore, from Eq. (8.17), we obtain

$$\frac{5}{8} \le \frac{1}{5}z + \frac{3}{40}s_1. \tag{8.18}$$

On adding a slack variable s_{G_1}, the Gomory's fractional cut becomes

$$\frac{5}{8} + s_{G_1} = \frac{1}{5}z + \frac{3}{40}s_1 \Rightarrow s_{G_1} - \frac{1}{5}z - \frac{3}{40}s_1 = -\frac{5}{8}. \tag{8.19}$$

Adding this additional constraint at the bottom of the optimal simplex Table 8.2, the new values so obtained are depicted in Table 8.3.

Iteration 1:

Remove the variable s_{G_1} from the basis and enter variable s_1 into the basis by applying the dual simplex method. The new solution is presented in Table 8.4.

The optimal solution presented in Table 8.4 is still noninteger. Therefore one more fractional cut needs to be generated. Since x is the only basic variable, whose value is a nonnegative fractional value, consider the x row (because of the largest fractional part) for constructing the cut:

$$\frac{10}{3} = x + \frac{2}{3}z - \frac{10}{3}s_{G_1}. \tag{8.20}$$

Table 8.3 First optimal but infeasible solution using Gomory's fractional cut of Example 8.2.

	c_j	2	20	-10	0	0	
c_B	**Basis**	x	y	z	s_1	s_{G_1}	b
20	y	0	1	$\frac{1}{5}$	$\frac{3}{40}$	0	$\frac{5}{8}$
2	x	1	0	0	$-\frac{1}{4}$	0	$\frac{5}{4}$
0	s_{G_1}	0	0	$-\frac{1}{5}$	$\boldsymbol{-\frac{3}{40}}$	1	$-\frac{5}{8} \rightarrow$
	$Z_j = \sum c_B a_{ij}$	2	20	4	1	0	15
	$C_j = c_j - Z_j$	0	0	-14	-1	0	
	$\text{Min} = \frac{C_j}{r_{3j}}(<0)$	–	–	70	$\frac{40}{3}$	–	
					$\uparrow$		

Table 8.4 First final optimal but infeasible solution using Gomory's fractional cut of Example 8.2.

	c_j	2	20	-10	0	0	
c_B	**Basis**	x	y	z	s_1	s_{G_1}	b
20	y	0	1	0	0	-1	0
2	x	1	0	$\frac{2}{3}$	0	$-\frac{10}{3}$	$\frac{10}{3}$
0	s_1	0	0	$\frac{8}{3}$	1	$-\frac{40}{3}$	$\frac{25}{3}$
	$Z_j = \sum c_B a_{ij}$	2	20	$\frac{4}{3}$	0	0	$\frac{20}{3}$
	$C_j = c_j - Z_j$	0	0	$-\frac{34}{3}$	0	$-\frac{40}{3}$	

The factoring of the x source row gives

$$\left(3+\frac{1}{3}\right) = (1+0)x + \left(0+\frac{2}{3}\right)z + \left(-4+\frac{2}{3}\right)s_{G_1} \tag{8.21}$$

$$\frac{1}{3} + \left(3 - x + 4s_{G_1}\right) = \frac{2}{3}z + \frac{2}{3}s_{G_1} \tag{8.22}$$

or

$$\frac{1}{3} \le \frac{2}{3}z + \frac{2}{3}s_{G_1}. \tag{8.23}$$

On adding another Gomory slack variable s_{G_2}, the second Gomory's fractional cut becomes

$$\frac{1}{3} + s_{G_2} = \frac{2}{3}z + \frac{2}{3}s_{G_1} \tag{8.24}$$

or

$$s_{G_2} - \frac{2}{3}z - \frac{2}{3}s_{G_1} = -\frac{1}{3}. \tag{8.25}$$

Adding this cut to the optimal simplex Table 8.4, the new table so obtained is presented in Table 8.5.

Iteration 2:

Enter nonbasic variable z into the basis to replace the basic variable s_{G_2} by applying the dual simplex method. The new solution is presented in Table 8.6.

The optimal solution presented in Table 8.6 is still noninteger because variable z does not assume an integer value. Thus a third fractional cut needs to be constructed with the help of the z row:

$$\frac{1}{2} = z + s_{G_1} - \frac{3}{2}s_{G_2} \tag{8.26}$$

$$\left(0 + \frac{1}{2}\right) = (1+0)z + (1+0)s_{G_1} + \left(-2 + \frac{1}{2}\right)s_{G_2} \tag{8.27}$$

or

$$\frac{1}{2} + \left(2s_{G_2} - z - s_{G_1}\right) = \frac{1}{2}s_{G_2} \tag{8.28}$$

or

$$\frac{1}{2} \leq \frac{1}{2}s_{G_2} \tag{8.29}$$

Table 8.5 Second optimal but infeasible solution using Gomory's fractional cut of Example 8.2.

	c_j	2	20	−10	0	0	0	
c_B	Basis	x	y	z	s_1	s_{G_1}	s_{G_2}	b
20	y	0	1	0	0	-1	0	0
2	x	1	0	$\frac{2}{3}$	0	$-\frac{10}{3}$	0	$\frac{10}{3}$
0	s_1	0	0	$\frac{8}{3}$	1	$-\frac{40}{3}$	0	$\frac{25}{3}$
0	s_{G_2}	0	0	$\mathbf{-\frac{2}{3}}$	0	$-\frac{2}{3}$	1	$-\frac{1}{3} \rightarrow$
	$Z_j = \sum c_B a_{ij}$	2	20	$\frac{4}{3}$	0	0	0	$\frac{20}{3}$
	$C_j = c_j - Z_j$	0	0	$-\frac{34}{3}$	0	$-\frac{40}{3}$	0	
	$\text{Min} = \frac{C_j}{r_{4j}}(<0)$	–	–	17	–	20	–	
				$\uparrow$				

Table 8.6 Second final optimal but infeasible solution using Gomory's fractional cut of Example 8.2.

	c_j	2	20	− 10	0	0	0	
c_B	**Basis**	x	y	z	s_1	s_{G_1}	s_{G_2}	b
20	y	0	1	0	0	− 1	0	0
2	x	1	0	0	0	− 4	0	3
0	s_1	0	0	0	1	− 16	4	7
−10	z	0	0	1	0	1	$-\frac{3}{2}$	$\frac{1}{2}$
	$Z_j = \sum c_B a_{ij}$	2	20	− 10	0	− 38	15	1
	$C_j = c_j - Z_j$	0	0	0	0	− 2	− 17	

Table 8.7 Third optimal but infeasible solution using Gomory's fractional cut of Example 8.2.

	c_j	2	20	− 10	0	0	0	0	
c_B	**Basis**	x	y	z	s_1	s_{G_1}	s_{G_2}	s_{G_3}	b
20	y	0	1	0	0	− 1	0	0	0
2	x	1	0	0	0	− 4	0	0	3
0	s_1	0	0	0	1	− 16	4	0	7
−10	z	0	0	1	0	1	$-\frac{3}{2}$	0	$\frac{1}{2}$
0	s_{G_3}	0	0	0	0	0	$\boldsymbol{-\frac{1}{2}}$	1	$-\frac{1}{2}$ →
	$Z_j = \sum c_B a_{ij}$	2	20	− 10	0	− 38	15	0	1
	$C_j = c_j - Z_j$	0	0	0	0	− 2	−17	0	
	Min $= \frac{C_j}{r_{5j}}(<0)$	–	–	–	–	–	34	–	
							↑		

The required Gomory's fractional cut obtained by adding slack variable s_{G_2} is represented as follows:

$$\frac{1}{2} + s_{G_3} = \frac{1}{2} s_{G_2} \Rightarrow s_{G_3} - \frac{1}{2} s_{G_2} = -\frac{1}{2} \tag{8.30}$$

Adding this cut to the bottom of the optimal simplex Table 8.6, the new table so obtained is presented in Table 8.7.

Iteration 3:

Remove the variable s_{G_3} from the basis and enter variable s_{G_2} into the basis by applying the dual simplex method. The new solution is presented in Table 8.8.

In Table 8.8, since the value of all basic variables is an integer value and all $c_j - Z_j \leq 0$, the current solution is an integer optimal solution, that is, $x = 2$, $y = 0$, and $z = 2$, and the corresponding optimal solution Max $Z = -16$.

Table 8.8 Third final optimal but infeasible solution using Gomory's fractional cut of Example 8.2.

	c_j	2	20	− 10	0	0	0	0	
c_B	**Basis**	x	y	z	s_1	s_{G_1}	s_{G_2}	s_{G_3}	b
20	y	0	1	0	0	− 1	0	0	0
2	x	1	0	0	0	− 4	0	2	2
0	s_1	0	0	0	1	− 16	0	8	3
−10	z	0	0	1	0	1	0	− 3	2
0	s_{G_2}	0	0	0	0	0	1	− 2	1
	$Z_j = \sum c_B a_{ij}$	2	20	− 10	0	− 38	0	0	−16
	$C_j = c_j - Z_j$	0	0	0	0	− 2	0	− 34	

8.1.4 Gomory's mixed-integer cutting plane method

In the previous section, the Gomory cutting method was discussed for LP problems where all decision variables, slack variables, and surplus variables were assumed to have integer values. This implies that this method is applicable only when all variables in the LP problem assume integer values. For example, consider the following constraint:

$$\frac{1}{2}x_1 + x - 2 \le \frac{11}{3}$$

or

$$\frac{1}{2}x_1 + x_2 + s_1 = \frac{11}{3}.$$

In this equation, decision variables x_1 and x_2 can assume integer values only if s_1 is noninteger. This situation can be avoided in two ways.

1. The noninteger coefficients in the constraints in the constraint equation can be removed by multiplying both sides of the constraint with a proper constant. For example, the above constraint is multiplied by 6 to obtain $3x_1 + 6x_2 \le 22$. However, this type of conversion is possible only when the magnitude of the integer coefficients is small.
2. Use a special cut called Gomory's mixed-integer cut or simply mixed cut, where only a subset of variables may assume integer values and the remaining variables (including slack and surplus variables) remain continuous. The details of developing this cut are presented in the following section.

8.1.4.1 Method for constructing additional constraint (cut)

Consider the following mixed integer programming problem:

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \tag{8.31}$$

subject to the constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \tag{8.32}$$

and x_j are integers; $j = 1, 2, \cdots, k$ $(k < n)$.

Suppose that the basic variables, x_r is restricted to be an integer and has the largest fractional value among all those basic variables that are restricted to take integer values. Then rewrite the rth constraint (row) from the optimal simplex table as follows:

$$x_{Br} = x_r + \sum_{j \neq r} a_{rj}x_j, \tag{8.33}$$

where x_j represents all the nonbasic variables in the rth row except variable, and x_r and x_{Br} is the noninteger value of the variable x_r.

Decompose coefficients of x_j and x_r variables and x_{Br} into integer and nonnegative fractional parts as follows.

$$x_{Br} = [x_{Br}] + f_r, \tag{8.34}$$

and $R_+ = \{j : a_{rj} \geq 0\}$, columns in simplex table for which $a_{rj} \geq 0$, $R_- = j : \text{a}_{rj} < 0$, columns in simplex table for which $a_{rj} < 0$.

Then Eq. (8.33) can be rewritten as follows:

$$[x_{Br}] + f_r = (1 + 0)x_r + \sum_{j \in R_+} a_{rj}x_j + \sum_{j \in R_-} a_{rj}x_j \tag{8.35}$$

Rearrange the terms in Eq. (8.35), so that all of the integer coefficients appear on the right-hand side. This gives:

$$\sum_{j \in R_+} a_{rj}x_j + \sum_{j \in R_-} a_{rj}x_j = f_r + \{[x_{Br}] - x_r\} = f_r + I, \tag{8.36}$$

where f_r is a strictly positive fraction number (i.e., $0 < f_r < 1$), and I is the integer value.

Since the terms in the bracket on the right-hand side of Eq. (8.36) are integers, left-hand side in Eq. (8.36) is either positive or negative according to the fact $f_r + I$ is positive or negative.

Case 1:. Let $f_r + I$ be positive. Then it must be f_r, $1+f_r$, $2+f_r$, $\cdots$ and we shall then have

$$\sum_{j\in R_+} a_{rj}x_j + \sum_{j\in R_-} a_{rj}x_j \geq f_r \tag{8.37}$$

Since $a_{rj} \in R_-$ are nonpositive and $x_j \geq 0$.

$$\sum_{j\in R_+} a_{rj}x_j \geq \sum_{j\in R_+} a_{rj}x_j + \sum_{j\in R_-} a_{rj}x_j \tag{8.38}$$

and hence $\sum_{j\in R_+} a_{rj}x_j \geq f_r$.

Step 2: Let $f_r + I$ be negative. Then it must be f_r, $-1+f_r$, $-2+f_r$, $\cdots$ and we shall have

$$\sum_{j\in R_-} a_{rj}x_j \leq \sum_{j\in R_+} a_{rj}x_j + \sum_{j\in R_-} a_{rj}x_j \leq -1+f_r \tag{8.39}$$

Multiplying both sides of Eq. (8.39) by the negative number $\left(\frac{f_r}{f_r-1}\right)$, we get

$$\left(\frac{f_r}{f_r-1}\right)\sum_{j\in R_-} a_{rj}x_j \geq f_r \tag{8.40}$$

Either of inequalities Eqs. (8.37) and (8.40) holds, since in both the cases, the left-hand side is nonnegative and one of these is greater than or equal to f_r. Thus any feasible solution to the mixed-integer programming must satisfy the following inequality:

$$\sum_{j\in R_+} a_{rj}x_j + \left(\frac{f_r}{f_r-1}\right)\sum_{j\in R_-} a_{rj}x_j \geq f_r \tag{8.41}$$

Inequality Eq. (8.41) is not satisfied by the optimal solution of the LP problem without integer requirement. This is because by putting $x_j = 0$ for all j, the left-hand side becomes zero, and the right-hand side becomes positive. Thus inequality Eq. (8.41) defines a cut.

Adding a nonnegative slack variable, we can rewrite Eq. (8.41) as follows:

$$s_g = -f_r + \sum_{j\in R_+} a_{rj}x_j + \left(\frac{f_r}{f_r-1}\right)\sum_{j\in R_-} a_{rj}x_j \tag{8.42}$$

Eq. (8.42) represents the required Gomory's cut.

For generating the cut Eq. (8.42), it was assumed that the basic variables x_r should take the integer value. But if one or more x_j, $j \in R$ are restricted to be integers to improve a cut shown in Eq. (8.41) to a better cut, that is, the coefficients a_{rj}, $j \in R_+$, and $a_{rj}\left(\frac{f_r}{f_r - 1}\right)$, $j \in R_-$ are desired to be as small as possible, we can proceed as follows. The value of the coefficients of x_r can be increased or decreased by an integral amount in Eq. (8.35) to get a term with the smallest coefficients in Eq. (8.41). To reduce the feasible region as much as possible through cutting planes, the coefficients of an integer variable x_r must be as small as possible. The smallest positive coefficient for x_r must be as small as possible. The smallest positive coefficient for x_r in Eq. (8.35) is given as follows:

$$\left\{ f_{rj} : \frac{f_r}{1 - f_r}(1 - f_r) \right\}. \tag{8.43}$$

The smaller of the two coefficients would be considered to cut Eq. (8.42), which penetrates deeper into the original feasible region. A cut is said to be deep if the intercepts of the hyperplane represented by a cut with the x-axis are larger. Hence,

$$f_r \le \frac{f_r}{1 - f_r}(1 - f_r); f_{rj} \le f_r \text{ and } f_r > \frac{f_r}{1 - f_r}(1 - f_r); f_{rj} > f_r. \tag{8.44}$$

Thus the new cut can be expressed as follows:

$$s_g = -f_r + \sum_{j \in R} f_{rj}^* x_j, \tag{8.45}$$

where

$$f_{rj}^* = \begin{cases} a_{rj}, & a_{rj} \ge 0 \text{ and } x_j \text{ noninteger} \\ \left(\dfrac{f_r}{1 - f_r}\right) a_{rj}, & a_{rj} < 0 \text{ and } x_j \text{ noninteger} \\ f_{rj}, & f_{rj} \le f_r \text{ and } x_j \text{ integer} \\ \left(\dfrac{f_r}{1 - f_r}\right) 1 - f_{rj}, & f_{rj} > f_r \text{ and } x_j \text{ integer} \end{cases} \tag{8.46}$$

Gomory's mixed-integer cutting plane method can be summarized in the following steps.

Algorithm 8.2.

Step 1: Initialization

Formulate the standard integer LP problem. Solve it by the simplex method, ignoring integer requirement of variables.

Step 2: Test of optimality

1. Examine the optimal solution. If all integers restricted basic integer values, then terminate the procedure. The current optimal solution is the optimal basic feasible solution to the integer LP problem.

2. If all integer-restricted basic variables are not integers, then go to Step 3.

Step 3: Generate a cutting plane

Choose a row r corresponding to a basic variable x_r that has the highest fractional value f_r and generate a cutting plane as explained earlier in the form (Eq. 8.42).

Step 4: Obtain the new solution

Add the cutting plane generated in Step 3 to the bottom of the optimal simplex table. Find a new optimal solution by using the dual simplex method and return to Step 2. The process is repeated until all restricted basic variables are integers.

Example 8.3. Solve the following mixed integer programming problem.

Maximize $Z = -3x + y + 3z$

Subject to the constraints

$$-x + 2y + z \leq 4$$

$$2y - \frac{3}{2}z \leq 1$$

$$x - 3y + 2z \leq 3$$

and $x, y \geq 0$, z nonnegative integer.

Solution: Expressing the LP problem in its standard form as follows:

Maximize $Z = -3x + y + 3z + 0s_1 + 0s_2 + 0s_3$

subject to the constraints

$$-x + 2y + z + s_1 = 4$$

$$2y - \frac{3}{2}z + s_2 = 1$$

$$x - 3y + 2z + s_3 = 3$$

and $x, y, s_1, s_2, s_3 \geq 0$, z nonnegative integer.

Step 1:

Ignoring the integer requirement, the optimal solution of the LP problem using the simplex method is presented in Table 8.9.

The noninteger optimal solution in Table 8.9 is $x = 0$, $y = \frac{5}{7}$, $z = \frac{13}{7}$, and Max $Z = \frac{44}{7}$.

Step 2:

Since basic variable z is required to be an integer, then consider z row to develop Gomory's mixed-integer cut as follows:

$$\frac{13}{7} = \frac{5}{14}x + z + \frac{3}{7}s_1 + \frac{2}{7}s_3. \tag{8.47}$$

Since, the coefficients of nonbasic variables x, s_1, and s_3 are positive, therefore factorizing the coefficients using rule [Eq. (8.46)], we obtain

$$\left(1 + \frac{6}{7}\right) = \left(0 + \frac{5}{14}\right)x + (1 + 0)z + \left(0 + \frac{3}{7}\right)s_1 + \left(0 + \frac{2}{7}\right)s_3 \tag{8.48}$$

or

$$\frac{6}{7} + (1 - z) = \frac{5}{14}x + \frac{3}{7}s_1 + \frac{2}{7}s_3 \tag{8.49}$$

or

$$\frac{6}{7} \leq \frac{5}{14}x + \frac{3}{7}s_1 + \frac{2}{7}s_3. \tag{8.50}$$

On adding slack variable s_{G_1}, we obtain Gomory's mixed-integer cut as follows:

$$\frac{6}{7} + s_{G_1} = \frac{5}{14}x + \frac{3}{7}s_1 + \frac{2}{7}s_3 \tag{8.51}$$

Table 8.9 Optimal noninteger solution for Example 8.3.

	c_j	-3	1	3	0	0	0	
c_B	**Basis**	x	y	z	s_1	s_2	s_3	b
1	y	$-\frac{3}{7}$	1	0	$\frac{2}{7}$	0	$-\frac{1}{7}$	$\frac{5}{7}$
0	s_2	$\frac{9}{7}$	0	0	$\frac{1}{7}$	1	$\frac{10}{7}$	$\frac{48}{7}$
3	z	$\frac{5}{14}$	0	1	$\frac{3}{7}$	0	$\frac{2}{7}$	$\frac{13}{7}$
	$Z_j = \sum c_B a_{ij}$	$\frac{9}{14}$	1	3	$\frac{11}{7}$	0	$\frac{5}{7}$	$\frac{44}{7}$
	$C_j = c_j - Z_j$	$-\frac{51}{14}$	0	0	$-\frac{11}{7}$	0	$-\frac{5}{7}$	

or

$$-\frac{5}{14}x-\frac{3}{7}s_1-\frac{2}{7}s_3+s_{G_1}=-\frac{6}{7} \tag{8.52}$$

Add this constraint to the bottom of Table 8.9. The new values so obtained are presented in Table 8.10.

Iteration 1:

Enter the nonbasic variable s_3 into the basis to replace with basic variable s_{G_1} using the dual simplex method. The new solution so obtained is presented in Table 8.11.

Since variable z has assumed integer value and all $c_j - Z_j \le 0$, the optimal mixed integer solution is $x=0$, $y=\frac{8}{7}$, $z=1$, and Max $Z=\frac{29}{7}$.

Table 8.10 Optimal but infeasible solution for Example 8.3.

	c_j	**−3**	**1**	**3**	**0**	**0**	**0**	**0**	
c_B	**Basis**	x	y	z	s_1	s_2	s_3	s_{G_1}	b
1	y	$-\frac{3}{7}$	1	0	$\frac{2}{7}$	0	$-\frac{1}{7}$	0	$\frac{5}{7}$
0	s_2	$\frac{9}{7}$	0	0	$\frac{1}{7}$	1	$\frac{10}{7}$	0	$\frac{48}{7}$
3	z	$\frac{5}{14}$	0	1	$\frac{3}{7}$	0	$\frac{2}{7}$	0	$\frac{13}{7}$
0	s_{G_1}	$-\frac{5}{14}$	0	0	$-\frac{3}{7}$	0	$\boldsymbol{-\frac{2}{7}}$	1	$-\frac{6}{7}\rightarrow$
	$Z_j=\sum c_B a_{ij}$	$\frac{9}{14}$	1	3	$\frac{11}{7}$	0	$\frac{5}{7}$	0	$\frac{44}{7}$
	$C_j=c_j-Z_j$	$-\frac{51}{14}$	0	0	$-\frac{11}{7}$	0	$-\frac{5}{7}$	0	
	Min $=\frac{C_j}{r_{4j}}(<0)$	$\frac{51}{5}$	—	—	$\frac{11}{3}$	—	$\frac{5}{2}$	—	
							$\uparrow$		

Table 8.11 The optimal solution for Example 8.3.

	c_j	**−3**	**1**	**3**	**0**	**0**	**0**	**0**	
c_B	**Basis**	x	y	z	s_1	s_2	s_3	s_{G_1}	b
1	y	$-\frac{1}{4}$	1	0	$\frac{1}{2}$	0	0	$-\frac{1}{2}$	$\frac{8}{7}$
0	s_2	$-\frac{1}{2}$	0	0	-2	1	0	5	$\frac{18}{7}$
3	z	0	0	1	0	0	0	1	1
0	s_3	5/4	0	0	$\frac{3}{2}$	0	1	$-\frac{7}{2}$	3
	$Z_j=\sum c_B a_{ij}$	$-\frac{1}{4}$	1	3	$\frac{1}{2}$	0	1	$\frac{5}{2}$	$\frac{29}{7}$
	$C_j=c_j-Z_j$	$-\frac{11}{4}$	0	0	$-\frac{1}{2}$	0	0	$-\frac{5}{2}$	

8.1.5 Branch and bound method

The branch and bound method was developed first by A. H. Land and A. G. Doig, which is used to solve all integer, mixed integer, and zero-one LP problems. The concept behind this method is to divide the feasible solution space of a LP problem into smaller parts called subproblems and then evaluate corner (extreme) points of each subproblem for an optimal solution.

The branch and bound method start by imposing bounds on the value of the objective function that helps to determine the subproblem to be eliminated from consideration when the optimal solution has been found. If the solution to a subproblem does not yield an optimal integer solution, a new subproblem is selected for branching. At a point where no more subproblem can be created, an optimal solution is arrived at.

The branch and bound method for the profit maximization ILP problem can be summarized in the following steps.

8.1.5.1 Procedure

Step 1: Initialization

Consider the following all integer programming problem

$$\text{Maximize } Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \tag{8.53}$$

subject to the constraints

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \\ \text{and } x_j \geq 0 \text{ and nonnegative integers.} \end{gathered} \tag{8.54}$$

Obtain the optimal solution of the given LP problem ignoring integer restriction on the variables.

1. If the solution of this LP problem (8.54) is infeasible or unbounded, the solution to the given all integer programming problem is also infeasible or unbounded, as the case may be.
2. If the solution satisfies the integer restrictions, the optimal integer solution has been obtained. If one or more basic variables do not satisfy integer requirement, then go to Step 2. Let the optimal value of the objective function of Eq. (8.54) be Z_1. This value provides an initial upper bound on objective function value and is denoted by Z_U.

3. Find a feasible solution by rounding off each variable value. The value of the objective function so obtained is used as the lower bound and is denoted by Z_L.

Step 2: Branching step

1. Let x_k be one basic variable that does not have an integer value and also has the largest fraction value.
2. Branch (or partition) the linear program Eq. (8.54) into two new LP subproblems (also called nodes) based on integer values of x_k that are immediately above and below its noninteger value. That is, it is partitioned by adding two mutually exclusive constraints.

$$x_k \le [x_k] \text{ and } x_k \ge [x_k] + 1$$

to the original LP problem. Here, $[x_k]$ is the integer portion of the current noninteger value of the variable x_k. This is done to exclude the noninteger value of the variable x_k. The two new LP subproblems are as follows.

LP subproblem B	LP subproblem C
Max $Z = \sum_{j=1}^{n} c_j x_j$	Max $Z = \sum_{j=1}^{n} c_j x_j$
Subject to $\sum_{j=1}^{n} a_{ij} x_j = b_i$	Subject to $\sum_{j=1}^{n} a_{ij} x_j = b_i$
$x_k \le [x_k]$	$x_k \ge [x_k] + 1$
and $x_j \ge 0$.	and $x_j \ge 0$.

Step 3: Bound step

Obtain the optimal solution of subproblems B and C. Let the optimal value of the objective function of LP B be Z_2 and that of LP C be Z_3. The best integer solution value becomes the lower bound on the integer LP problem objective function value (initially this is the rounded off value). Let the lower bound be denoted by Z_L.

Step 4: Fathoming step

Examine the solution of both LP B and LP C.

1. If a subproblem yields an infeasible solution, then terminate the branch.
2. If a subproblem yields a feasible solution but not an integer solution, then return to Step 2.
3. If a subproblem yields a feasible integer solution, examine the value of the objective function. If this value is equal to the upper bound, an optimal solution has been reached. But if it is not equal to the upper bound but exceeds the lower bound, this value is considered as a new

upper bound and return to Step 2. Finally, if it is less than the lower bound, terminate this branch.

Step 5: Termination

The procedure of branching and bounding continues until no further subproblem remains to be examined. At this stage, the integer solution correspond to the current lower bound in the optimal all integer programming problem solution.

Example 8.4. Solve the following all integer programming problem using the branch and bound method.

$$\text{Minimize } Z = 3x + 2.5y$$

subject to the constraints

$$x + 2y \geq 20$$

$$3x + 2y \geq 50$$

and

$$x, y \geq 0 \text{ and integers.}$$

Solution: Relaxing the integer requirement, the optimal noninteger solution of the given ILP problem, obtained by the graphical method, is $x = 15$, $y = 2.5$, and $Z_1 = 51.25$. This value of Z_1 represents the initial lower bound, $Z_L = 51.25$ on the value of the objective function, that is the value of the objective function in the subsequent steps cannot be less than 51.25.

The variable $y = 2.5$ is the only noninteger solution value and is, therefore, selected for dividing the given problem into two subproblems: B and C. To eliminate the fractional part of $y = 2.5$, two new constraints $y \leq 2$ and $y \geq 3$ are created by adding the given set of constraints as follows:

Linear subproblem B	Linear subproblem C
Maximize $Z = 3x + 2.5y$	Maximize $Z = 3x + 2.5y$
subject to	subject to
$x + 2y \geq 20$	$x + 2y \geq 20$
$3x + 2y \geq 50$	$3x + 2y \geq 50$
$y \leq 2$	$y \geq 3$
and	and
$x, y \geq 0$ and integers.	$x, y \geq 0$ and integers.

Subproblems B and C are solved graphically. The feasible solutions are as follows:

Subproblem B: $x = 16$, $y = 2$ and Minimum $Z_2 = 53$.

Subproblem C: $x = 14.66$, $y = 3$ and Minimum $Z_3 = 51.5$.

Since the solution of subproblem B is all integer, no further decomposition (branching) of this subproblem is required. The value of $Z_2 = 53$ becomes the new lower bound. A noninteger solution of subproblem C and also $Z_3 < Z_2$ indicates that further decomposition of this problem needs to be done to search for the desired integer solution. However, if $Z_3 \geq Z_2$, then no further branching was needed from subproblem C. The second lower bound takes on the value $Z_L = 51.25$ at node A.

Dividing subproblem C into two new subproblems D and E by adding constraints $x \leq 14$ and $x \geq 15$ as follows:

Linear subproblem D	Linear subproblem E
Maximize $Z = 3x + 2.5y$	Maximize $Z = 3x + 2.5y$
subject to	subject to
$x + 2y \geq 20$	$x + 2y \geq 20$
$3x + 2y \geq 50$	$3x + 2y \geq 50$
$x \leq 14$	$x \geq 15$
and	and
$x, y \geq 0$ and integers.	$x, y \geq 0$ and integers.

Subproblems D and E are solved graphically. The feasible solutions are given as follows:

Subproblem D: $x = 14$, $y = 4$, and Minimum $Z_4 = 52$.

Subproblem E: $x = 15$, $y = 3$, and Minimum $Z_5 = 52.5$.

The feasible solutions of both subproblems D and E are all integer, and therefore the branch and bound procedure is terminated. The feasible solution of subproblem D is considered as an optimal basic feasible solution because this solution is all integer and the value of the objective function is the lowest among all such values. The branch and bound procedure for the given problem is shown in Fig. 8.2.

Remarks. The aforementioned procedure can be represented by an enumeration tree. Each node in the tree represents a subproblem to be evaluated. Each branch of the tree creates a new constraint that is added to the original problem.

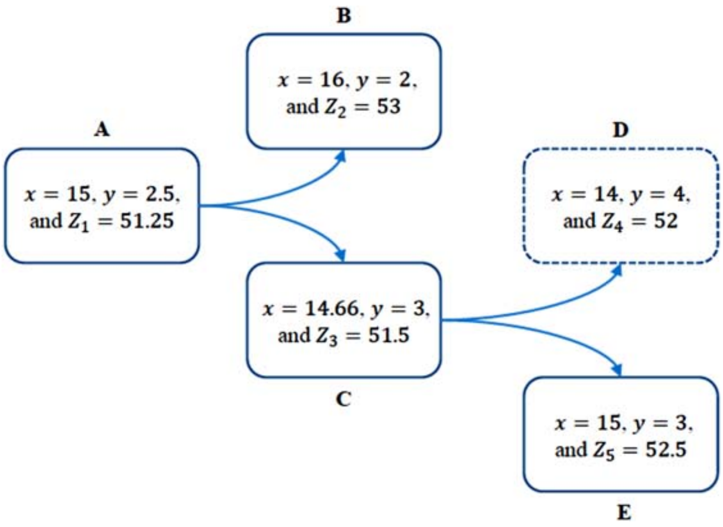

B
x = 16, y = 2, and Z₂ = 53
A
x = 15, y = 2.5, and Z₁ = 51.25
D
x = 14, y = 4, and Z₄ = 52
x = 14.66, y = 3, and Z₃ = 51.5
C
x = 15, y = 3, and Z₅ = 52.5
E

Figure 8.2 Branch and bound solution of Example 8.4.

8.1.6 Applications of zero-one integer programming

A large number of real-world problems such as capital budgeting problems, fixed cost problems, sequencing problems, scheduling problems, location problems, traveling salesman problem, etc., require all or some of the decision variables to assume the value of either zero or one. The zero-one integer programming problem is stated as follows:

$$\text{Minimize } Z = \sum_{j=1}^{n} c_j x_j \tag{8.55}$$

subject to the constraints

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i;\ i = 1,\ 2,\ \cdots,\ m$$

$$\text{and } x_j = 0 \text{ or } 1. \tag{8.56}$$

8.2 Integer nonlinear programming

The idea is that we can convert an integer polynomial programming problems to zero-one LP problems and then investigate the same. Consider the optimization problem

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

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m$$

$$x_i = \text{integer},\ i = 1,\ 2,\ \cdots,\ n \tag{8.57}$$

where f and g_j, $j = 1, 2, \cdots, m$ are polynomials in the variables $x_1,\ x_2,\ \cdots,\ x_n$. A typical term in the polynomials can be represented as follows:

$$c_k \prod_{l=1}^{n_k} (x_l)^{a_{kj}}, \tag{8.58}$$

where c_k is a constant, a_{kl} a nonnegative constant exponent, and n_k the number of variables appearing in the kth term. We shall convert the integer polynomial programming problem stated in Eq. (8.57) into an equivalent zero-one LP problem in two stages. In the first stage, we see how an integer variable x_i can be represented by an equivalent system of zero-one (binary) variables. We consider the conversion of a zero-one polynomial programming problem into a zero-one LP problem in the second stage.

8.2.1 Representation of an integer variable by an equivalent system of binary variables

Let x_i be any integer variable whose upper bound is given by u_i so that

$$x_i \leq u_i < \infty. \tag{8.59}$$

We assume that the value of the upper bound u_i can be determined from the constraints of the given problem.

We know that in the decimal number system, an integer p is represented as follows:

$$p = p_0 + 10^1 p_1 + 10^2 p_2 + \cdots,\ 0 \leq p_i \leq (10 - 1 = 9) \text{ for } i = 0,\ 1,\ 2,\ \cdots$$

and written as $p = p_2 p_1 p_0$ by neglecting the zeros to the left. For example, we write the number $p = 008076$ as 8076 to represent $p = 6 + \left(10^1\right)7 + \left(10^2\right)0 + \left(10^3\right)8 + \left(10^4\right)0 + \left(10^5\right)0$. Similarly, the integer p can also be represented in the binary number system as follows:

$$p = q_0 + 2^1 q_1 + 2^2 q_2 + 2^3 q_3 + \cdots$$

where $0 \leq q_i \leq (2 - 1 = 1)$ for $i = 0, 1, 2, \cdots$

In general, if $y_i^{(0)},\ y_i^{(1)},\ y_i^{(2)},\ \cdots$ denote binary numbers (which can take a value of 0 or 1), the variable x_i can be expressed as follows:

$$x_i = \sum_{k=0}^{N_i} 2^k y_i^{(k)}, \tag{8.60}$$

where N_i is the smallest integer, such that

$$\frac{u_i + 1}{2} \leq 2^N. \tag{8.61}$$

Thus the value N_i can be selected for any integer variable x_i once its upper bound u_i is known. For example, for the number 97, we can take $u_i = 97$ and hence the relation

$$\frac{u_i + 1}{2} = \frac{98}{2} = 49 \leq 2^N,$$

is satisfied for $N_i \geq 6$. Hence by taking $N_i = 6$, we can represent u_i as follows:

$$97 = q_0 + 2^1 q_1 + 2^2 q_2 + 2^3 q_3 + 2^4 q_4 + 2^5 q_5 + 2^6 q_6$$

where $q_0 = 1,\ q_1 = q_2 = q_3 = q_4 = 0$, and $q_5 = q_6 = 1$. A systematic method of finding the values of $q_0,\ q_1,\ q_2,\ \cdots$ is given in the following section.

Practice set

1. Solve the following all integer programming problems, using Gomory's cutting plane algorithm:

a. Max $Z = x_1 + 2x_2$
subject to
$2x_2 \leq 7,$
$x_1 + x_2 \leq 7,$
$2x_1 \leq 11$
and $x_2 \geq 0$ and integers.

b. Max $Z = 2x_1 + 1.7x_2$
subject to
$4x_1 + 3x_2 \leq 7,$
$x_1 + x_2 \leq 4,$
and $x_2 \geq 0$ and integers.

c. Max $Z = 3x_1 + 2x_2 + 5x_3$
subject to
$5x_1 + 3x_2 + 7x_3 \leq 28,$
$4x_1 + 5x_2 + 5x_3 \leq 30$
and $x_1,\ x_2,\ x_3 \geq\ 0$ and integers.

d. Max $Z = 3x_1 + 4x_2$
subject to
$3x_1 + 2x_2 \leq 8,$
$x_1 + 4x_2 \geq 10$
and $x_1,\ x_2 \geq 0$ and integers.

e. Max $Z = 4x_1 + 3x_2$
subject to
$x_1 + 2x_2 \leq 4,$
$2x_1 + x_2 \leq 6$
and $x_1,\ x_2 \geq 0$ and integers.

f. Max $Z = 7x_1 + 9x_2$
subject to
$-x_1 + 3x_2 \leq 6,$
$7x_1 + x_2 \leq 35$
and $x_1,\ x_2 \geq 0$ and integers.

2. Solve the following integer programming problems using both Gomory's cutting plane algorithm and by the branch or bound method

a. Max $Z = 1.5x_1 + 3x_2 + 4x_3$
subject to
$2,5x_1 + 2x_2 + 4x_3 \leq 12,$
$2x_1 + 4x_2 - x_3 \leq 7,$
and
integers x_1, x_2, $x_3 \geq 0$.

b. Max $Z = 7x_1 + 6x_2$
subject to
$2x_1 + 3x_2 \leq 12,$
$6x_1 + 5x_2 \leq 30,$
and
$x_{1,}\ x_2 \geq 0$ and integers.

c. Max $Z = 5x_1 + 4x_2$
subject to
$x_1 + x_2 \geq 2,$
$5x_1 + 3x_2 \leq 15,$
$3x_1 + 5x_2 \leq 15,$

and
$x_1, x_2 \geq 0$ and integers.

d. Max $Z = -3x_1 + x_2 + 3x_3$
subject to
$-x_1 + 2x_2 + x_3 \leq 4,$
$2x_1 + 1.5x_2 \leq 1,$
$x_1 - 3x_2 + 2x_3 \leq 3,$
and
$x_1, x_2 \geq 0$; x_3 nonnegative integers.

e. Max $Z = x_1 + x_2$
subject to
$2x_1 + 5x_2 \geq 16,$
$6x_1 + 5x_2 \leq 30,$
and
$x_2 \geq 0$; x_1 nonnegative integers.

f. Max $Z = 4x_1 + 3x_2 + 5x_3$
subject to
$2x_1 - 2x_2 + 4x_3 \geq 7,$
$2x_1 + 6x_2 - 2x_3 \geq 5,$
and
$x_2 \geq 0$ and x_1, x_3 nonnegative integers.

g. Max $Z = 110x_1 + 100x_2$
subject to
$6x_1 + 5x_2 \leq 29,$
$4x_1 + 14x_2 \leq 48,$
and
$x_1, x_2 \geq 0$ and integers.

h. Max $Z = 2x_1 + 3x_2$
subject to
$x_1 + 3x_2 \leq 9,$
$3x_1 + x_2 \leq 7,$
$x_1 - x_2 \leq 1,$
and
$x_1, x_2 \geq 0$ and integers.

3. A manufacturer of toys makes two types of toys, A and B. Processing of these two toys is done on two machines X and Y. The toy A requires 2 hours on machine X and 6 hours on machine Y. Toy B requires 4 hours on machine X and 5 hours on machine Y. There are 16 hours of time per day available on machine X and 30 hours on machine Y. The profit obtained on both the toys is the same, that is, ₹5 per toy. Formulate and solve this problem as an integer LP problem to determine the daily production of each of the two toys.

4. A stereo equipment manufacturer can produce two models A and B of 40 and 80 watts of total power each. Each model passes through three manufacturing divisions, namely, 1, 2 and 3, where model A takes 4, 2.5, and 4.5 hours each and model B takes 2, 1, and 1.5 hours each. The three divisions have a maximum of 1600, 1200, and 1600 hours every month, respectively. Model A gives a profit contribution of ₹400 each and B of ₹100 each. Assuming abundant product demand, formulate and solve this problem as an integer LP problem to determine the optimal product mix and the maximum contribution.

5. A manufacturing company produces two types of screws—metal and wooden. Each screw has to pass through the slotting and threading machines. The maximum time that each machine can be run is 150 hours per month. A batch of 50 wooden screws requires 2 minutes on the threading machine and 3 minutes on the slotting machine. Metal screws of the same batch size require 8 minutes on the threading machine and 2 minutes on the slotting machine. The profit contribution for each batch of wooden and metal screws is ₹1 and ₹2.50, respectively. Formulate and solve this problem as an integer LP problem to determine the optimal product mix for maximum profit contribution.

6. A dietician for a hospital is considering a new breakfast menu that includes oranges and cereal. This breakfast must meet the minimum requirements for the vitamins A and B. The number of milligrams of each of these vitamins contained in a purchasing unit, for each of these foods, is as follows:

Vitamin	**Milligrams per purchasing unit of food**		**Minimum requirement (mg)**
	Oranges (doz)	**Cereal (box)**	
A	1	2	20
B	3	2	50

The cost of the food ingredients is ₹15 per dozen for oranges and ₹12.50 per box of cereal. For dietary reasons, at least one unit of each food type must be used in the menu plan. Formulate and solve this problem as an integer programming problem.

7. The owner of a ready-made garments store makes two types of shirts: Arrow and Wings. He makes a profit of ₹10 and ₹15 per shirt on Arrow and Wings, respectively. To stitch these shirts, he has also two tailors A and B at his disposal. Tailors A and B can devote, at the most, 12 hours each day. Both these shirts are to be stitched by both the tailors. Tailor A and Tailor B spend 3 hours and 4 hours, respectively, in stitching an Arrow shirt and 4 hours and 3 hours in stitching a Wings shirt. How many shirts of both types should be stitched to maximize daily profits? (A noninteger solution for this problem will not be accepted.)
8. Suppose five items are to be loaded in a vessel. The weight (W), volume (V), and price (P) are tabulated. The maximum cargo weight and cargo volume are $W = 112$ and $V = 109$, respectively. Determine the most valuable cargo load in the discrete unit of each item.

Item	**W**	**V**	**Price (in ₹)**
1	5	1	4
2	8	8	7
3	3	6	6
4	2	5	5
5	7	4	4

Formulate and solve this problem as an ILP model.

9. The cutting division of Photo Films Corporation requires from the stock control department, plastic films of 85 feet of fixed unit length that can be cut according to two different patterns. The first pattern would cut each film length into two 35 feet pieces with the remaining 15 feet to scrap. The second pattern will cut each film length into a 35 feet piece and two 25 feet pieces with nothing to scrap. The present order from a customer is for eight pieces of 35 feet long and six pieces of 25 feet length. What number of plastic films of 85 feet should be cut according to the patterns (assuming both patterns have to be used) to minimize the scrap? (A noninteger solution for this problem will not be accepted).
10. A production manager faces the problem of job allocation between his two production crews. The production rate of crew one is five units per hour and that of crew two is six units. The normal working hours for each crew is 8 hours per day. If the firm requires overtime

operations, each crew can work up to 11 hours (union contract). The firm has a special contract with a customer to provide a minimum of 120 units of the product the next day. The union contract calls for working arrangements, where each crew should work at least 8 hours per day, and any overtime not exceeding 3 hours should be in terms of increments of an hour. The operating costs per hour are ₹200 for crew one and ₹220 for crew two. Formulate and solve this problem as an integer programming model to determine the optimum job allocation.

11. A company makes two products, each of which requires time on four different machines. Only integer amounts of each can be made. The company wants to find the output mix that maximizes total profits, given the specifications shown in the following table:

Total machine hours available	**Machine**	**Machine hours (per unit)**	
		Product I	**Product II**
3400	A	200	500
1450	B	100	200
3000	C	400	300
2400	D	400	100
Profit/unit (₹1000)		6	6

Further reading

Balas, E. (1965). An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 517–546.

Bhavikatti, S. S. (2010). *Fundamentals of optimum design in engineering*. New Delhi: New Age International (P) Limited Publishers.

Gisvold, K., & Moe, J. (1972). A method for nonlinear mixed-integer programming and its application to design problems. *Journal of Engineering for Industry*, 353–364.

Gomory, R. (1963). An all-integer programming algorithm. In J. Muth, & G. Thompson (Eds.), *Industrial*. Englewood Cliffs, NJ: Prentice-Hall. (Chapter 13).

Land, A., & Doig, A. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 497–520.

Rao, S. S. (1995). *Optimization theory and applications*. New Delhi: New Age International (P) Limited Publishers.

Sharma, J. K. (2006). *Operations research theory & applications*. New Delhi: Macmillan India Ltd.

Taha, H. A. (1992). *Operations research: An introduction*. New York: Macmillan.

Watters, L. (1967). Reduction of integer polynomial programming problems to zero–one linear. *Operations Research*, 1171–1174.

Winston, W. L. (1991). *Operations research: Applications and algorithms*. Boston, MA: PWS-Kent.

CHAPTER NINE

Multiobjective optimization

Decision-making is the process of selecting a possible course of action from all the available alternatives. In almost all such problems, the multiplicity of criteria for judging the alternative is pervasive. That is, for many such problems, the decision-maker (DM) wants to attain more than one objective or goal in selecting the course of action while satisfying the constraints dictated by the environment, process, and resources. Another characteristic of these problems is that the objectives are non-commensurable. Mathematically, these problems can be represented as follows:

$$\text{Max}\left[f_1(X), f_2(X), \cdots, f_k(X)\right]$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m, \tag{9.1}$$

where X is an n-dimensional decision variable vector. The problem consists of n decision variables, m constraints, and k objectives. Any or all of the functions may be nonlinear. In the literature, this problem is often referred to as a vector maximum problem (VMP).

Traditionally, there are two approaches for solving the VMP. One of them is to optimize one of the objectives while appending the other objectives to a constraint set so that the optimal solution would satisfy these objectives at least up to a predetermined level. The problem is given as follows:

$$\text{Max}\, f_i(X)$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m$$

$$f_l(X) \geq a_l,\ l = 1,\ 2,\ \cdots,\ k, \text{ and } l \neq i, \tag{9.2}$$

where a_l is any acceptable predetermined level for objective l. The other approach is to optimize a superobjective function created by multiplying

© 2020 Elsevier Inc.
All rights reserved.

each objective function with suitable weight and then by adding them together. This approach leads to the solution of the following problem:

$$\text{Max} \sum_{i=1}^{k} w_i f_i(X)$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m. \tag{9.3}$$

The weights are usually normalized, so that $\sum_{i=1}^{k} w_i = 1$.

Temporary scenarios of the aforementioned approaches are best. Often they lead to a solution that may not be the best or most satisfactory. Because of the incommensurability and the conflicting nature of the multiple criteria, the problem becomes complex, and it becomes difficult to choose the acceptable levels, a_l, in Eq. (9.2), which will result in a nonempty constraint set in the first attempt for the solution. In the first approach, the implied value trade-off between f_l and f_i is represented as follows:

$$\text{Value trade} - \text{off} = \begin{cases} 0, & f_l \geq a_l \\ \infty, & f_l < a_l \end{cases}. \tag{9.4}$$

This may not be the actual value structure of the DM, and this value structure is sensitive to the level a_l. For the second approach, the major problem is in determining the proper weight w_i. The w_is are sensitive to the level of the particular objective and the levels of all other objectives.

Multiple objective decision-making (MODM) methods are the result of the desire to eliminate the aforementioned difficulties as well as to treat the objectives independently. The rapid progress in such a short time necessitates a thorough review of the existing literature and systematic classification of methods for the guidance of future users.

A multiobjective optimization problem with inequality constraints can be stated as (equality constraints, if they exist, can also be included in the formulation of the problem) follows:

Find

$$X = \begin{Bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}, \tag{9.5}$$

which minimizes

$$f_1(X), f_2(X), \cdots, f_k(X) \tag{9.6}$$

subject to

$$g_j(X) \leq 0, \ j = 1, \ 2, \ \cdots, \ m, \tag{9.7}$$

where k denotes the number of objective functions to be minimized. Any or all of the functions $f_i(X)$ and $g_j(X)$ may be nonlinear. The multiobjective optimization problem is also known as a vector minimizing problem.

In general, no solution vector X exists that minimizes all the k objective functions simultaneously. Hence, a new concept, known as the Pareto optimum solution, is used in multiobjective optimization problems. A feasible solution X is called Pareto optimal if there exist no other feasible solution Y, such that $f_i(Y) \leq f_i(X)$ for $i = 1, 2, \cdots, k$ with $f_j(Y) < f_i(X)$ for at least one j. In other words, a feasible vector X is called Pareto optimal if there is no other feasible solution Y that would reduce some objective function without causing a simultaneous increase in at least one other objective function. For example, if the objective functions are given by $f_1 = (x-3)^4$ and $f_2 = (x-6)^2$. For this problem, all the values of x between 3 and 6 (points on the line segment $P\ Q$) denote Pareto optimal solutions.

The four words most used in multiple criteria decision-making (MCDM) literature are attributes, objectives, goals, and criteria. Some authors make distinctions in their usage, while many use them interchangeably. We shall make some distinctions among these words in terms of their usage. They are defined as follows:

1. Attributes

Attributes are the characteristics, qualities, or performance parameters of alternatives. Multiple attribute decision problems involve the selection of the "best" alternative from a pool of preselected alternatives described in terms of their attributes. As an example, consider the problem facing a city commission where the issue is to select the direction of the future expansion of the city. Due to the geography of the area, the planners have determined that the city can grow only in one of the three directions—east, south-east, and west. Here, the DMs are presented with three alternatives, and they must choose the one they think best for the city. The alternatives will be compared based on the attributes settled on by the commission. The attributes may be the cost of service expansion, the effect on downtown vitality, the possibility of flooding, preservation of balance in city neighborhoods, proximity to existing recreational facilities, etc.

2. Objectives

Objectives are the directions to do better as perceived by the DM. Thus objectives are reflections of the desires of the DM, and they indicate the direction in which the DM wants the organization to work. MODM problems, as a result, involve the design of alternatives that optimize or "best satisfy" the objectives of the DM. Consider the problem of making development plans for the government of a developing country.

3. Goals

Goals are things desired by the DM expressed in terms of a specific state in space and time. Thus while objectives give the desired direction, goals give a desired (or target) level to achieve.

4. Criteria

Criteria are standards of judgment or rules to test acceptability. This is the dictionary definition of the term. However, as used in the MCDM literature, it indicates attributes and/or objectives. In this sense, an MCDM problem includes either multiple attributes or multiple objectives or both. This is used to indicate the general class of problems with which we are concerned here. The notations that will be used throughout this review are defined next. This will save us from confusion and provide a notational uniformity.

The multiple objective decision problems will be defined mathematically as follows:

$$\text{Maximize}\left[f_1(X),\, f_2(X),\, \cdots,\, f_k(X)\right]$$

subject to

$$g_j(X) \le 0,\; j = 1,\; \ldots,\; m,$$

where X is an n-dimensional vector. The problem indicates that there are k number of objectives that are to be maximized, m number of constraints, and n number of decision variables. Any or all of the functions $f_i(X)$ and $g_j(X)$ can be nonlinear. As mentioned earlier, this problem is also referred to as a VMP. The symbol $\leq$ will be defined as follows for use in vectors.

1. For any two vectors x and y, $x < y$ iff $x_i \leq y_i$ $\forall i$ for which the two vectors are defined. Similarly, $y \geq x$ iff $y_i \geq x_i$ $\forall i$. For the VMP, the constraints $g_j(X) \leq 0$ define a feasible set X, which is defined as follows.
2. The feasible set X is the set of decision variable x, which satisfy the constraint set $g(x) \leq 0$, that is, $X = \{x | g(x) \leq 0\}$.

   With each point X, there is an associated $f(x)$ vector. So, it is possible to map X into a set S in the objective function space.
3. S is a set of functional values $f(x)$ for each vector $x \in X$, that is, $S = \{f(x) | x \in X\}$.

   As a result, a VMP can be represented in any of the following three equivalent notations:

$$\text{Max}\, f(X)$$

subject to

$$g(X) \leq 0$$

or,

$$\text{Max}\, f(X)$$

subject to

$$x \in X$$

or,

$$\text{Max}\, f(X)$$

subject to

$$f(x) \in S.$$

We shall now introduce the idea of an optimal solution, nondominated solution, and preferred solution in the context of a VMP. A clear notion of these concepts is crucial to the understanding of the solution methodologies for multiple criteria decision problems. We will clarify these concepts with the help of numerical Examples 9.1 and 9.2.

Example 9.1.

$$\text{Max}\, f_1(X) = x + y$$

$$\text{Max}\, f_2(X) = y - x$$

subject to

$$x \leq 4$$
$$y \leq 4$$
$$x,\ y \geq 0.$$

Example 9.2.

$$\text{Max}\, f_1(X) = y$$
$$\text{Max}\, f_2(X) = y - x$$

subject to

$$x \leq 3$$
$$y \leq 3$$
$$x,\ y \geq 0.$$

Fig. 9.1 shows the decision variable space representation of Example 9.1. Fig. 9.3 shows the objective function space representation of

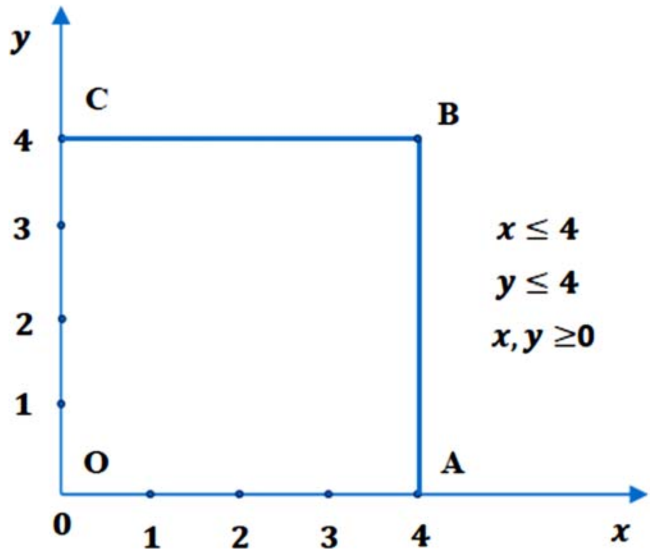


Figure 9.1 Decision variable space representation of the feasible area of Example 9.1.

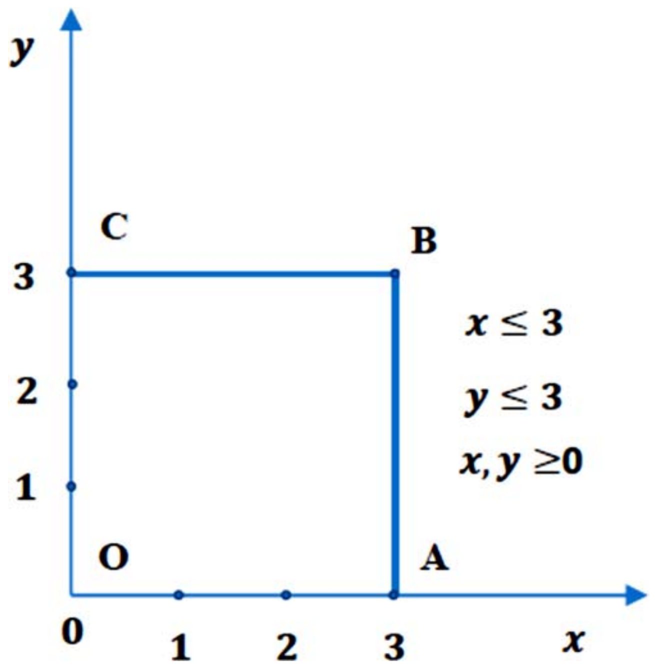


Figure 9.2 Decision variable space representation of the feasible area of Example 9.2.

Example 9.1. Similarly, Fig. 9.2 shows the decision variable space representation of Example 9.2. Fig. 9.4 shows the objective function space representation of Example 9.2.

An optimal solution to a VMP is one that results in the maximum value of each of the objective functions simultaneously. That is, x^* is an optimal solution to the VMP iff $x^* \in X$ and $f(x^*) \geq f(x) \forall x \in X$.

Since it is the nature of MODM problems to have conflicting objectives, usually there is no optimal solution to a VMP. Looking at our examples, we notice that Example 9.1 has an optimal solution. This optimal solution is located at $(x, y) = (0, 4)$, where $(f_1, f_2) = (4, 4)$; this is shown by point B' in Fig. 9.3. For Example 9.1, there is no optimal solution. The maximum value of $f_1 = 8$, which occurs at $(x, y) = (4, 4)$; the maximum value of $f_2 = 4$, which occurs at $(x, y) = (0, 4)$. The point shown in Fig. 9.3 gives $f_1 = 8$ and $f_2 = 4$; however, the point is outside the feasible region S_1. It may be noted that the optimal solution is also known as the superior solution or the maximum solution.

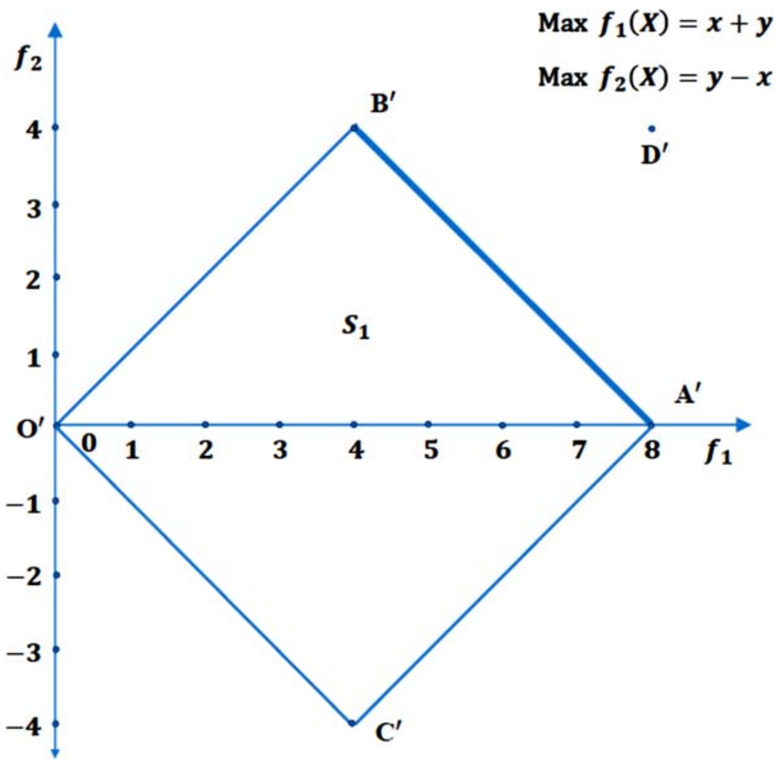


Figure 9.3 Objective function space representation of the feasible area of Example 9.1.

In the absence of an optimal solution to the VMP, the idea of a preferred solution has been introduced. However, before defining a preferred solution, a nondominated solution is introduced in the context of VPM.

4. A nondominated solution is one in which no one objective function can be improved without a simultaneous detriment to at least one of the other objectives of the VMP. That is x^* is a nondominated solution to the VMP iff there does not exist any $x \in X$, such that $f_i(x^*) \le f_i(x)$ $\forall i$ and $f_j(x^*) < f(x)$ for at least one j.

The nondominated solution is also known as Pareto optimal solution, noninferior solution, or efficient solution. In Example 9.1, all solutions for $y = 4$ and $0 \le x \le 4$ are nondominated. In Fig. 9.3, these nondominated solutions are represented by the straight line B′A′. All the nondominated solutions must lie on the boundary B′A′ of S_1 because any point interior of S_1 is dominated by at least one point on the boundary B′A′. This is a very important characteristic of the set of nondominated solutions of VMP.

For most of the VMPs, the number of solutions in the set of nondominated solutions is quite large. So, the DM must make a final selection of the most satisfactory solution by using some other criteria. Thus this final solution chosen is known as the preferred solution.

5. A preferred solution is a nondominated solution that is chosen by the DM, through some additional criteria, as the final decision. As such, it lies in the region of acceptance of all the criteria values for the problem. A preferred solution is also known as the best solution given in (Fig. 9.4).

Several methods have been developed for solving a multiobjective optimization problem. Some of these methods are briefly described in the following paragraphs. Most of these methods generate a set of Pareto optimal solutions and use some additional criterion or rule to select one particular Pareto optimal solution as the solution of the multiobjective optimization problem.

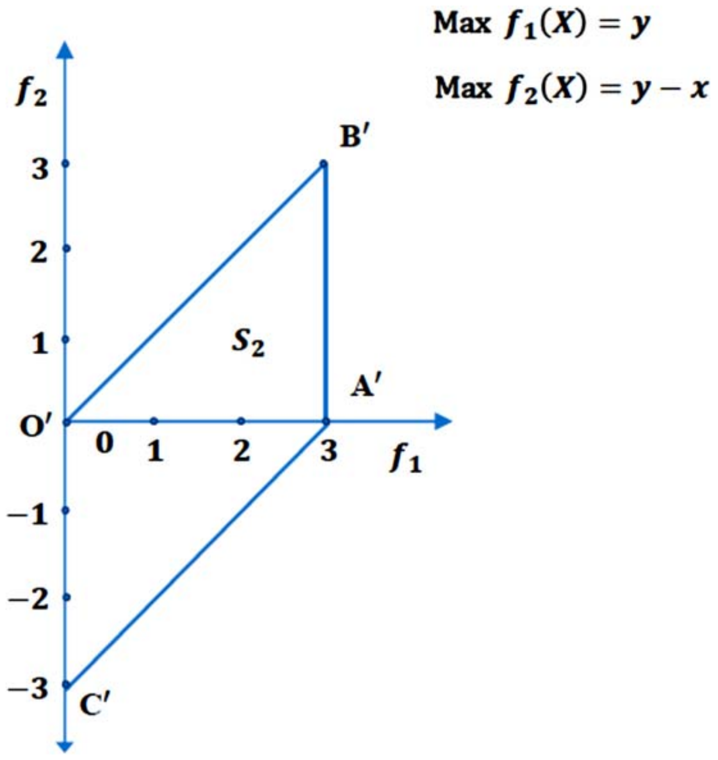


Figure 9.4 Objective function space representation of the feasible area of Example 9.2.

9.1 Global criterion method

In the global criterion method, the optimum solution X^* is found by minimizing a preselected global criterion, $F(X)$, such as the sum of the squares of the relative deviations of the individual objective functions from the feasible ideal solutions. Thus X^* is found by minimizing

$$F(X) = \sum_{i=1}^{k} \left\{ \frac{f_i(X_i^*) - f_i(X)}{f_i(X_i^*)} \right\}^p \tag{9.8}$$

subject to

$$g_j(X) \le 0, \ j = 1, \ 2, \ 3, \ \cdots, \ m,$$

where p is a constant (a usual value of p is 2) and X_i^* is the ideal solution for the ith objective function. The solution X_i^* is obtained by minimizing $f_i(X)$ subject to the constraints $g_j(X) \le 0, j = 1, 2, \cdots, m$.

9.1.1 Methods for a priori articulation information given

A priori means the preference information is given to the analyst (who is responsible for the solution of the VMP) before it solves the problem. The DM provides the information during or after the actual mathematical formulation of the problem. The information may be either (1) cardinal or (2) mixed (ordinal and cardinal) information. In the case of cardinal information, the DM must give some judgment about specific objective preference levels or specific trade-offs. But if the information is mixed, the DM must rank the objectives in the order of their importance (Example 9.3).

Example 9.3.

Solve the following VMP:

$$\text{Max} \, f_1(x, y) = 0.4x + 0.3y$$

$$\text{Max} \, f_2(x, y) = x$$

subject to

$$g_1(x,y) = x + y \leq 400$$

$$g_2(x,y) = 2x + y \leq 500$$

$$x,\ y \geq 0.$$

Solution: The problem can be solved by using the following three steps.

Step 1: Obtain the ideal solutions.

Considering the first objective function, we may write

$$\text{Max}\, f_1(x,y) = 0.4x + 0.3y$$

subject to

$$g_1(x,y) = x + y \leq 400$$

$$g_2(x,y) = 2x + y \leq 500$$

$$x,\ y \geq 0.$$

This linear programming problem can be solved by any standard method, and the solution obtained is $x = 100$ and $y = 300$, and the optimal function value is $f_1 = 130$.

Considering the second objective function, we may write

$$\text{Max}\, f_1(x,y) = x$$

subject to

$$g_1(x,y) = x + y \leq 400$$

$$g_2(x,y) = 2x + y \leq 500$$

$$x,\ y \geq 0.$$

This linear programming problem can be solved by any standard method, and the solution is $x = 250$ and $y = 0$, and the optimal function value is $f_2 = 250$.

Step 2: Construct a pay-off table.

The obtained results of Step 1 are presented in Table 9.1.

In Table 9.1, the first row represents the solution vector $(100, 300)$, which maximizes the objective function $f_1 = 130$ and the objective

Table 9.1 Obtained results of Step 1.

	f_1	f_2	x_1	x_2
f_1	130	100	100	300
f_2	100	250	250	0

function $f_2 = 100$. Similarly, the second row represents the solution vector $(250, 0)$, which maximizes the objective function $f_2 = 250$ and the objective function $f_1 = 100$.

Step 3: Obtain the desired solution.

1. The desired solution for $p = 1$ is to find the vector $z = (x, y)$ to minimize

$$F = \frac{130 - (0.4x + 0.3y)}{130} + \frac{250 - x}{250}$$

subject to

$$g_1(x, y) = x + y \leq 400$$
$$g_2(x, y) = 2x + y \leq 500$$
$$x,\ y \geq 0.$$

This linear programming problem can be solved by any standard method, and the solution is $x = 250$ and $y = 0$; optimal function values are $f_1 = 100$ and $f_2 = 250$.

2. The desired solution for $p = 2$ is to find the vector $z = (x, y)$ to minimize

$$F = \left(\frac{(130-(0.4x+0.3y))}{130}\right)^2 + \left(\frac{250-x}{250}\right)^2$$

subject to

$$g_1(x, y) = x + y \leq 400$$
$$g_2(x, y) = 2x + y \leq 500$$
$$x,\ y \geq 0.$$

This is a nonlinear programming problem, which can be can be solved by the sequential unconstrained optimization technique, and the solution is $x = 230.7$ and $y = 38.6$; the optimal function values are $f_1 = 103.9$ and $f_2 = 230.7$. Hence, this is an improved and desired solution.

9.2 Utility function method

In the utility function method, a utility function $U_i(f_i)$ is defined for each objective depending on the importance of f_i compared with the other objective functions. Then, a total or overall utility function U is defined, for example, as follows:

$$U = \sum_{i=1}^{k} U_i(f_i). \tag{9.9}$$

The solution vector X^* is then found by maximizing the total utility U subjected to the constraints $g_j(X) \le 0,\ j = 1,\ 2,\ \cdots,\ m$. A simple form of Eq. (9.9) is given as follows:

$$U = \sum_{i=1}^{k} U_i = -\sum_{i=1}^{k} w_i f_i(X) \tag{9.10}$$

where w_i is a scalar weighting factor associated with the ith objective function. This method (Eq. (9.10)) is also known as the weighting function method.

The major advantage of this method is that U has been correctly assessed and used, and it will ensure the most satisfactory solution to the DM. The solution will be a point at which the nondominated set of solutions and the indifference curves of the DM are tangent to each other. The indifference curves can be considered as contours of equal utility. Thus the solution will have the highest utility for the DM, and it will also be nondominated. The major difficulty with the method is that the DM is required to articulate preference judgment in an information vacuum. Moreover, like the single-objective utility functions, the multiobjective utility functions suffer from their intransitivity in terms of time and state.

9.3 Inverted utility method

In the inverted utility function method, we invert each utility and try to minimize or reduce the total undesirability. Thus if $U_i(f_i)$ denotes the utility function corresponding to the ith objective function, the total undesirability is obtained as follows:

$$U^{-1} = \sum_{i=1}^{k} U_i^{-1} = \sum_{i=1}^{k} \frac{1}{U_i}. \tag{9.11}$$

The solution of the problem is found by minimizing U^{-1} subject to the constraints $g_j(X) \le 0, j = 1, 2, \cdots, m$.

9.4 Bounded objective function method

In the bounded objective function method, the minimum and the maximum acceptable achievement levels for each objective function f_i are specified as $L^{(i)}$ and $U^{(i)}$, respectively, for $i = 1, 2, \cdots, k$. Then, the optimum solution X^* is found by minimizing the most important objective function, say, the rth one, as follows:

$$\text{Minimize } f_r(X)$$

subject to

$$g_j(X) \le 0,\ j = 1,\ 2,\ \cdots,\ m$$

$$L^{(i)} \le f_i \le U^{(i)},\ i = 1,\ 2,\ \cdots,\ k,\ i \ne r. \tag{9.12}$$

The difficulty with these methods is to find $L^{(i)}$ and $U^{(i)}$ values from the DM before any preliminary solution. Since the DM has to give these values in an information void, they may result in turning Eq. (9.12) into problems with inconsistent constraints. Even if the DM is well informed, and the problems are not turned into inconsistent ones, the solution of Eq. (9.12) may not necessarily provide a satisfactory solution for DM. Another question in this approach is which objective should be used for $f_r(x)$.

Because of the aforementioned difficulties, these methods are now rarely used alone; rather they are used in conjunction with other solution methods.

9.4.1 Methods for mixed ordinal and cardinal information given

The most important information needed for the lexicographic method and the goal programming is the ordinal interobjective preference information although some of these methods also require some cardinal information.

9.5 Lexicographic model

In the lexicographic model, the objectives are ranked in the order of importance by the designer. The optimum solution X^* is then found by minimizing the objective functions starting with the most important and proceeding according to the order of importance of the objectives. Let the subscripts of the objectives indicate not only the objective function number but also the priorities of the objectives. Thus $f_i(X)$ and $f_k(X)$ denote the most and least important objective functions, respectively. The first problem is formulated as follows:

$$\text{Minimize}\, f_i(X)$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m, \tag{9.13}$$

and its solution X_1^* and $f_1^* = f_1(X_1^*)$ is obtained. Then, the second problem is formulated as follows:**

$$\text{Minimize}\, f_2(X)$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m$$

$$f_1(X) = f_1^*. \tag{9.14}$$

The solution to this problem is obtained as X_2^* and $f_2^* = f_2(X_2^*)$. This procedure is repeated until all the k objectives have been considered. The ith problem is given by

$$\text{Minimize}\, f_i(X)$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m$$

$$f_l(X) = f_l^*,\ l = 1,\ 2,\ \cdots,\ i-1, \tag{9.15}$$

and its solution is found as X_i^* and $f_i^* = f_i(X_i^*)$. Finally, the solution obtained at the end (i.e., X_k^*) is taken as the desired solution X^* of the original multiobjective optimization problem.

9.6 Goal programming method

In the simplest version of goal programming, the designer sets goals for each objective that the designer wishes to attain. The optimum solution X^* is then defined as the one that minimizes the deviations from the set goals. Thus the goal programming formulation of the multiobjective optimization problem leads to

$$\text{Minimize}\left[\sum_{j=1}^{k}\left(d_j^+ + d_j^-\right)^p\right]^{\frac{1}{p}}, p \geq 1$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m$$

$$f_j(X) + d_j^+ - d_j^- = b_j,\ j = 1,\ 2,\ \cdots,\ k$$

$$d_j^+ \geq 0,\ j = 1,\ 2,\ \cdots,\ k$$

$$d_j^- \geq 0,\ j = 1,\ 2,\ \cdots,\ k$$

$$d_j^+ d_j^- = 0,\ j = 1,\ 2,\ \cdots,\ k, \tag{9.16}$$

where b_j is the goal set by the designer for the jth objective and d_j^+ and d_j^- are, respectively, the underachievement and overachievement of the jth goal. The value of p is based on the utility function chosen by the designer. Often the goal for the jth objective, b_j, is found by first solving the problem:

$$\text{Minimize}\, f_j(X)$$

subject to

$$g_j(X) \leq 0,\ j = 1,\ 2,\ \cdots,\ m. \tag{9.17}$$

If the solution to the problem stated in Eq. (9.17) is denoted by X_j^*, then b_j is taken as $b_j = f_j(X_j^*)$.

9.6.1 Practice set

1. Consider the following two objective functions:

$$\text{Minimize}\, f(x, y) = x^4 + y^4$$

$$\text{Minimize}\; g(x, y) = e^{x^2} + e^{y^2}$$

If both objectives are to be optimized, does the resulting problem give rise to multiple Pareto optimal solutions?

2. Consider the following multiobjective optimization problem.

$$f_1(x, y) = y$$

$$f_2(x, y) = x$$

subject to

$$g(x, y) = x^2 + y^2 \leq 1$$

$$h(x, y) = (x-1.5)^2 + (x-1.5)^2 \leq 1,$$

and identify the Pareto optimal set when

a. $f_1(x, y)$ is minimized and $f_2(x, y)$ is maximized.

b. $f_1(x, y)$ is maximized and $f_2(x, y)$ is minimized.

c. Both $f_1(x, y)$ and $f_2(x, y)$ are minimized.

d. Both $f_1(x, y)$ and $f_2(x, y)$ are maximized.

3. Find the Pareto optimal solution of the following multiobjective optimization problem:

$$\text{Minimize}\, f_1(x, y) = x^2 + y^2$$

$$\text{Minimize}\, f_2(x, y) = y^2 - x + 5$$

subject to

$$-5 \leq x, y \leq 5.$$

Use the utility function $U = 50 - f_1(x, y) - f_2(x, y)$.

4. Find the Pareto optimal solution of the following multiobjective optimization problem.

$$\text{Minimize}\, f_1(x, y) = x^4 + y^4$$

$$\text{Minimize}\, f_2(x, y) = y^2 - x^2 + 25$$

subject to

$$-5 \le x, y \le 5.$$

Use the utility function $U = 50 - 2f_1(x, y) + f_2(x, y)$.

5. Solve the following goal programming problem.

$$\text{goal}\cdot f_1(x_1, x_2) = x_1^2 + x_2^2 \le 2$$

$$\text{goal}\cdot f_2(x_1, x_2) = x_1^2 - x_2^2 \le -2$$

in terms of the weight factors $(\theta, 1-\theta)$. What are the solutions for $\theta = 1, 0.5$, and 0?

Further reading

Deb, K. (2003). *Multi-objective optimization using evolutionary algorithms*. Singapore: Wiley.

Eschenauer, H., Koski, J., & Osyczka, A. (1990). *Multicriteria design optimization: Procedures and applications*. New York, NY: Springer-Verlag.

Fonseca, C., & Fleming, P. (1995). An overview of evolutionary algorithms in multi-objective optimization. *Evolutionary Computation Journal*, 1−16.

Hwang, C.-L., & Md. Masud, A. (1979). Multiple objective decision making—Methods and applications. In M. Beckmann, & H. Kunzin (Eds.), *Lecture notes in economics and mathematical systems* (p. 164). Heidelberg: Springer-Verlag Berlin.

Ignizio, J. (1982). *Linear programming in single- and multiple-objective systems*. Englewood Cliffs, NJ: Prentice-Hall.

Rao, S. S. (1995). *Optimization theory and applications*. New Delhi: New Age International (P) Limited Publishers.

CHAPTER TEN

Nature-inspired optimization

10.1 Genetic algorithm

In traditional search optimization techniques, viz., feasible direction method, sequential linear programming, and sequential unconstrained minimum technique, the search starts from a designed point by selecting the appropriate direction along with the associated length. Besides, the main focus is on the inviolability of constraints, and the process is terminated when the improvement in the objective function and designed variables is negligible. These techniques lead the objective function to a local optimum value. But to get global optimum, different starting points are considered. However, the genetic algorithm (GA) is a nontraditional search technique. Nontraditional techniques can avoid the disadvantage of traditional techniques such as the assumption of continuous variables. For example, if we consider a structural engineering problem, the area of steel sections is not a continuous variable. Even in concrete structures from practical considerations, cross-sections of members are to be kept in multiples of 5–10 mm. In traditional search, variables are treated as continuous, and after getting optimum, next higher sections are chosen.

The GA is ideally suited to handle discrete variable problems. In most cases, it finds a globally optimum solution. This algorithm starts with a set of design points with available variables and proceeds on the mechanics of natural genetic and natural selection. From the initial set of design points, a new set of design points are produced and weaker points are eliminated. Thus, the search is based on Darwin's theory of survival of the fittest. The GA gets its name since it uses basic elements of natural genetics that is a reproduction, crossover, and mutation.

As a first approach, let us restrict the view that GAs are optimization methods. In general, optimization problems are given in the following form.

Find an $x^* \in X$, such that f is maximal in x^*, where $f : X \rightarrow R$ is an arbitrary real-valued function, that is,

© 2020 Elsevier Inc.
All rights reserved.

$$f(x^*) = \max_{x \in X} f(x). \tag{10.1}$$

In practice, it is sometimes almost impossible to obtain global solutions in the strict sense of Eq. (10.1). Depending on the actual problem, it can be sufficient to have a local maximum or to be at least close to a local or global maximum. So, let us assume in the following that we are interested in values x, where the objective function f is "as high as possible."

The search space X can be seen in direct analogy to the set of competing individuals in the real world, where f is the function that assigns a value of "fitness" to each individual (i.e., of course, a serious simplification). In the real world, reproduction and adaptation are carried out on the level of genetic information. Consequently, GAs do not operate on the values in the search space X, but on some coded versions of them (strings for simplicity).

Assume S to be a set of strings (in nontrivial cases with some underlying grammar). Let X be the search space of an optimization problem as earlier, then a function

$$c: X \longrightarrow S x \longmapsto c(x) \tag{10.2}$$

is called the coding function. Conversely, a function

$$\tilde{c}: S \longrightarrow X s \longmapsto \tilde{c}(s) \tag{10.3}$$

is called the decoding function.

In general coding and decoding, functions may not be bijective. It has to be specified depending on the needs of the actual problem. However, it is in most cases useful to work with injective decoding functions. Moreover, the following equality is often supposed to be satisfied

$$(c \circ \tilde{c}) \equiv id_s. \tag{10.4}$$

Finally, we can write down the general formulation of the encoded maximization problem. Find an $s^* \in S$ such that $\tilde{f} = f \circ \tilde{c}$ is as large as possible.

Table 10.1 provides a list of different expressions, which are common in genetics, along with their equivalent in the framework of GAs.

After this preparatory work, we can write the basic structure of a GA in Algorithm 10.1.

Table 10.1 List of different expressions, which are common in genetics, along with their equivalent in the framework of GAs.

Natural evolution	Genetic algorithm
(i) Genotype	(i) Coding string
(ii) Phenotype	(ii) Decoded point
(iii) Chromosome	(iii) String
(iv) Gene	(iv) String position
(v) Allele	(v) Value at a certain position
(vi) Fitness	(vi) Objective function value

Algorithm 10.1.

```
Step 1: Consider $t = 0$; compute initial population $B_0$.
Step 2: Assign the stopping condition.
Step 3: Select individuals for reproduction.
Step 4: Create off-springs by crossing individuals.
Step 5: Eventually mutate some individuals.
Step 6: Compute the new generation and go to Step 2.
If the stopping condition is fulfilled then Stop.
```

From the above algorithm, it is seen that the transition from one generation to the next consists of the following four fundamental components.

Selection: It is a mechanism for selecting individuals (strings) for reproduction according to their fitness (objective function value).

Crossover: It is a method of merging the genetic information of two individuals; if the coding is chosen properly, two good parents produce good children.

Mutation: In real evolution, the genetic material can be changed randomly by erroneous reproduction or other deformations of genes, for example, by gamma radiation. In GAs, a mutation can be realized as a random deformation of the strings with a certain probability. The positive effect is the preservation of genetic diversity and, as an effect, that local maxima can be avoided.

Sampling: It is a process that computes a new generation from the previous one and its off-springs.

Comparing with traditional continuous optimization methods, viz., Newton or gradient descent methods, we can observe the following important differences.

1. GAs manipulate coded versions of the problem parameters instead of the parameters themselves, that is, the search space is S instead of X itself.
2. While almost all conventional methods search from a single point, GAs always operate on a whole population of points (strings). This contributes much to the robustness of GAs. It improves the chance of reaching the global optimum and vice versa reduces the risk of becoming trapped in a local stationary point.
3. Normal GAs do not use any auxiliary information about the objective function value such as derivatives. Therefore, they can be applied to any kind of continuous or discrete optimization problem. The only thing to be done is to specify a meaningful decoding function.
4. GAs use probabilistic transition operators, while conventional methods for continuous optimization apply deterministic transition operators. More specifically, the way a new generation is computed from the actual one has some random components.

Example 10.1. Let us assume that the strings we consider are all from the set.

$$S = \{0,1\}^n, \tag{10.5}$$

where n is the length of the strings.

The population size is denoted by m.

The generation at time t is a list of m strings, which we will denote with

$$\mathrm{B}_t = (b_{1,t}, b_{2,t}, \cdots, b_{m,t}) \tag{10.6}$$

In this chapter, all GAs will obey the following basic structure.

Algorithm 10.2.

```
Step 1: Consider $t = 0$; compute initial population
  $B_0 = (b_{1,0}, \cdots, b_{m,0})$.
Step 2: Assign the stopping condition.
Step 3: Start
for $i = 1$ to $m$
Select an individual $b_{i,t+1}$ from $\mathrm{B}_t$;
```

```
for $i = 1$ to $m - 1$
if random $[0, 1] \leq p_c$ then
cross $b_{i,t+1}$ with $b_{i+1,t+1}$;
for $i = 1$ to $m$
Eventually, mutate $b_{i,t+1}$;
$tUNDEFINEDratio; = t + 1$
end
```

Here, selection, crossover (done only with a probability of p_c), and mutation are still degrees of freedom, while the sampling operation is already specified. As it is easy to observe, every selected individual is replaced by one of its children after crossover and mutation. Unselected individuals die immediately. This is a rather common sampling operation, although other variants are known and reasonable.

In the following, we will discuss the three remaining operations selection, crossover, and mutation.

10.1.1 Genetic operations on binary strings

10.1.1.1 Selection

Selection is the component that guides the algorithm to the solution by preferring individuals with high fitness over low-fitted ones. It can be a deterministic operation, but in most implementations, it has random components. One variant, which is very popular nowadays is the following scheme, where the probability to choose a certain individual is proportional to its fitness. It can be regarded as a random experiment with

$$\mathrm{P}\left[b_{j,t} \text{ is selected}\right] = \frac{f\left(b_{j,t}\right)}{\sum\limits_{k=1}^{m} f\left(b_{k,t}\right)}. \tag{10.7}$$

Eq. (10.7) only makes sense if all the fitness values are positive. If this is not the case, a nondecreasing transformation $\varphi: R \to R^+$ must be applied (a shift in the simplest case). Then, the probabilities can be expressed as follows:

$$\mathrm{P}\left[b_{j,t} \text{ is selected}\right] = \frac{\varphi\left(b_{j,t}\right)}{\sum\limits_{k=1}^{m} \varphi\left(b_{k,t}\right)}. \tag{10.8}$$

We can force the property (Eq. (10.7)) to be satisfied by applying a random experiment, which is, in some sense, a generalized roulette game.

In this roulette game, the slots are not equally wide, that is, the different outcomes can occur with different probabilities. The algorithmic formulation of the selection scheme (Algorithm 10.2) can be written as follows, analogously for the case of Algorithm 10.3.

Algorithm 10.3.

```
$x = \text{random}[0, 1]$;
$i = 1$
while $i < m$ & $x < \frac{\sum_{j=1}^{i} f(b_{j,t})}{\sum_{j=1}^{m} f(b_{j,t})}$
$i = i + 1$;
select $b_{i,t}$.
```

This method is often called the proportional selection.

10.1.1.2 Crossover

In natural reproduction, as it appears in the real world, the genetic material of the two parents is mixed when the gametes of the parents merge. Usually, chromosomes are randomly split and merged, with the consequence that some genes of a child come from one parent, while others come from the other parents. This mechanism is called crossover. It is a very powerful tool for introducing new genetic material and maintaining genetic diversity, but with the outstanding property, that good parents also produce well-performing children or even better ones.

Crossover is the exchange of genes between the chromosomes of the two parents. In the simplest case, we can realize this process by cutting two strings at a randomly chosen position and swapping the two tails.

Algorithm 10.4.

```
$pos = \text{random}1, \cdots, n-1$;
for $i = 1$ to $pos$ do
begin
child$_1[i]$ = parent$_1[i]$;
child$_2[i]$ = parent$_2[i]$;
end
for $i = pos + 1$ to $n$ do
```

```
begin
child$_1$[$i$] = parent$_1$[$i$];
child$_2$[$i$] = parent$_2$[$i$];
end
```

One-point crossover is a simple and often-used method for GAs, which operate on binary strings. For other problems or different codings, other crossover methods can be useful or even necessary.

1. *N*-point crossover: Instead of only one, *N* breaking points are chosen randomly. Every second section is swapped. Among this class, the two-point crossover is particularly important.
2. Segmented crossover: Similar to *N*-point crossover with the difference that the number of breaking points can vary.
3. Uniform crossover: For each position, it is decided randomly if the positions are swapped.
4. Shuffle crossover: First, a randomly chosen permutation is applied to the two parents, then *N*-point crossover is applied to the shuffled parents; finally, the shuffled children are transformed back with the inverse permutation.

10.1.1.3 Mutation

The last component of a simple GA is mutation. Here, the mutation is the random deformation of the genetic information of an individual employing radioactive radiation or other environmental influences. In real reproduction, the probability that a certain gene is mutated in almost equal for all genes. So, it is near at hand to use the following mutation technique for a given binary string *s*, where *pM* is the probability that a single gene is modified:

Algorithm 10.5.

```
for $iUNDEFINEDratio; = 1$ to $n$
if random $[0, 1] < pM$ then
invert $s[i]$;
```

Of course, *pM* should be rather low to avoid that the GA behaves chaotically like a random search. Again, similar to the case of crossover, the choice of the appropriate mutation technique depends on the coding and the problem itself. Here, we include a few alternatives.

1. Inversion of single bits: With probability pM, one randomly chosen bit is negated.
2. Bitwise inversion: The whole string is inverted bit by bit with pM.
3. Random selection: With probability pM, the string is replaced by a randomly chosen one.

Example 10.2. Consider the problem of finding the global maximum of the following function:

$$f_1 : \{0, \ldots, 31\} \to R$$
$$x \mapsto x^2$$

The solution is obvious, but the simplicity of this problem allows us to compute some steps by hand to gain some insight into the principles behind GAs. The first step on the checklist of things, which have to be done to make a GA work, is to specify a proper string space along with an appropriate coding and decoding scheme. In this problem, it is near at hand to consider $S = \{0, 1\}^5$, where a value from $\{0, \cdots, 31\}$ is coded by its binary representation. Correspondingly, a string is decoded as follows:

$$\tilde{c}(s) = \sum_{i=0}^{4} s[4 - i] \cdot 2^i$$

Assume that we use Algorithm 10.5 as it is, with a population size of $m = 4$, a crossover probability $p_c = 1$, and a mutation probability of $pM = 0.001$. If we compute the initial generation randomly with a uniform distribution over $\{0, 1\}^5$, we obtain the following in the first step:

Individual No.	String (genotype)	x value (phenotype)	$f(x) = x^2$	$\frac{f_i}{\sum f_i}$
1	0 1 1 0 1	13	169	0.14
2	1 1 0 0 0	24	576	0.49
3	0 1 0 0 0	8	64	0.06
4	1 0 0 1 1	19	361	0.31

One can compute easily that the sum of fitness values is 1170, where the average is 293 and the maximum is 576. We see from the last column in which way proportional selection favors high-fitted individuals (such as no. 2) over low-fitted ones (such as no. 3). A random experiment could, for instance, give the result that individuals' no. 1 and no. 4 are selected for the new generation, while no. 3 dies and no. 2 is selected twice, and we obtain the second generation as follows.

Set of selected individuals	**Crossover site (random)**	**New population**	***x* value**	$f(x) = x^2$
0 1 1 0 |1	4	0 1 1 0 0	12	144
1 1 0 0 |0	4	1 1 0 0 1	25	625
1 1 |0 0 0	2	1 1 0 1 1	27	729
1 0 |0 1 1	2	1 0 0 0 0	16	256

So, we obtain a new generation with a sum of fitness values of 1754, an average of 439, and a maximum of 729. We can see from this very basic example in which way selection favors high-fitted individuals and how the crossover of two parents can produce offspring, which is even better than both of its parents. In this way, the reader can continue the process and complete this example.

10.1.2 Analysis of GA

For conventional deterministic optimization methods, such as gradient methods, Newton or Quasi-Newton methods, etc., it is rather usual to have results that guarantee that the sequence of iterations converges to a local optimum with a certain speed or an order. For any probabilistic optimization method, theorems of this kind cannot be formulated, because the behavior of the algorithm is not determinable in general. Statements about the convergence of probabilistic optimization methods can only give information about the expected or average behavior. In the case of GAs, there are a few circumstances that make it even more difficult to investigate their convergence behavior.

1. Since a single transition from one generation to the next is a combination of usually three probabilistic operators (selection, crossover, and mutation), the inner structure of a GA is rather complicated.
2. For each of the involved probabilistic operators, many different variants have been proposed; thus, it is not possible to give general convergence results since the choice of the operators influences the convergence fundamentally.

For simplicity, we will restrict to algorithms of type 10.1, that is, GAs with a fixed number m of binary strings of fixed length n. Unless stated otherwise, no specific assumptions about selection, crossover, or mutation will be made. Briefly reconsider the example in 10.2. We saw that the transition from the first to the second generation is given as follows:

Gen. #1	$f(x)$		Gen. #2	$f(x)$
0 1 1 0 1	169		0 1 1 0 0	144
1 1 0 0 0	576	$\Longrightarrow$	1 1 0 0 1	625
0 1 0 0 0	64		1 1 0 1 1	729
1 0 0 1 1	361		1 0 0 0 0	256

It is easy to see the advantage to have a 1 in the first position. The number of strings having this property increased from 2 in the first to 3 in the second generation.

10.2 Neural network-based optimization

Artificial neural network (ANN) is one of the well-known and trendy areas of artificial intelligence (AI), which deals with the idea and principles of an organizational structure of the human brain. ANN is an abstract computational model based on natural neural networks. Dr. Robert Hecht-Nielsen defines a neural network as a computing system made up of several simple, but highly interconnected processing elements that process information by their dynamic state response to external inputs. ANN algorithms are processing devices that are modeled after the neuronal structure of the mammalian cerebral cortex but on much smaller scales. The first conceptual ANN model was prepared by researchers in 1943. Their research articles tell that the concept of a neuron, a single cell living in a network of cells that receives inputs, processes those inputs and generates an output. ANN is a data modeling tool that depends on various parameters and learning methods. Neural networks are typically organized in layers. Layers are made up of several interconnected nodes, which contain activation functions. ANN process information parallels through nodes to handle certain problems. ANN acquires knowledge through learning, and this knowledge is stored within interneuron connections' strength, which is expressed by numerical values called weights. These weights are used to compute output signal values for a new testing input signal value.

Patterns are presented to the network via the input layer that communicates to one or more hidden layers, where the actual processing is done via a system of weighted connections. Then, the hidden layers link to an output layer, where the solution of the problem, that is, the output, is obtained. A basic structure of ANN is shown in Fig. 10.1. In Fig. 10.1, x_j represents the input nodes, w_{ij} are defined as the weights from the input layer to the hidden layer, and v_i and y denote the weights from the hidden layer to the output layer and the output node, respectively. Due to the special quality of learning capacity, ANN is recognized as a powerful technique to investigate a wide range of real-life problems. We may see the application of the same in the fields of pattern recognition, clustering, function approximation, prediction, identification problem, solving ordinary, and partial differential equations. Furthermore, the insight study of ANN involves architectures of ANN, various paradigms of learning, different learning processes, and activation functions, which are briefly discussed in the following sections.

10.2.1 Architecture of ANN

This is one of the most important components of the ANN mechanism. The architecture of ANN is used to build an intelligent program using models that simulate the working of neurons in the human brain. The prime element of the network is the structure of the information processing system. The network is composed of a large number of highly interconnected processing elements (nodes) working in parallel to investigate a certain problem. This computing system is made up of a large number of artificial neurons (nodes) and a huge number of interconnections among

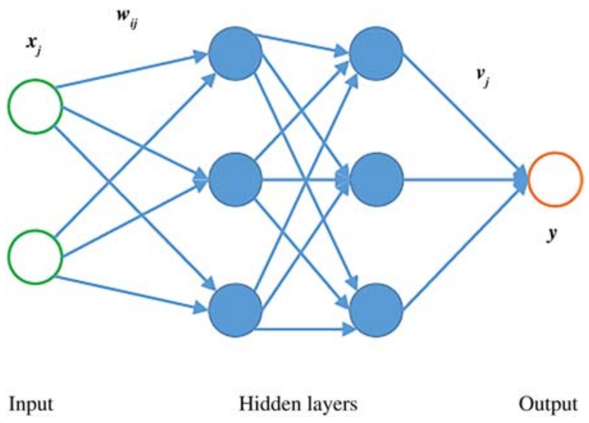


Figure 10.1 Model diagram of artificial neural network.

them. Based on the interconnections, the following classes of neural network architectures can be identified.

1. *Feed-forward neural network*

In a feed-forward neural network, neurons are organized in the form of layers. Neurons in a layer receive input from the previous layer and feed their output to the next layer. Network connections to the same or previous layers are not allowed. The data processed from the input node to the output node in a strictly feed-forward manner. There is no feedback, that is, the output of any layer does not affect the same layer.

Consider, a vector $X = (x_1, x_2, \cdots, x_n)$ is the input vector, f is the activation function, and $O = (o_1, o_2, \cdots, o_m)$ is the output vector, and $W = w_{ij}$ is the weight matrix. Then, matrix WX is the input value, which is a scalar product of input vectors and weight vectors w_{ij}

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} \tag{10.9}$$

2. *Feedback neural network*

In a feedback neural network, the output of one layer routes back to the previous layer. This network can have signals traveling in both directions by the introduction of loops in the network. This network is very powerful and, at times, gets extremely complicated. All possible connections between neurons are allowed. Feedback neural networks are used in optimization problems, where the network looks for the best arrangement of interconnected factors. They are dynamic, and their state changes continuously until they reach an equilibrium point.

10.2.2 Paradigms of learning

One of the most powerful features of ANN is the ability to learn and generalize the data from a set of training data. Mainly, the learning situations in ANN are classified into the following two types, that is, supervised and unsupervised.

1. *Supervised learning*

In supervised training, both inputs and outputs are provided. Then, the network processes the inputs and compares its obtained outputs against the desired outputs. A comparison is made between the network's computed output and the corrected expected output to determine the error. The error can then be used to change network parameters, which results in an improvement in performance.

2. *Unsupervised or self-organization learning*

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. Then, the system itself decides what features it will use to group the input data. This is often referred to as self-organization or adaptation. Unsupervised learning seems much harder than supervised learning, and this type of training generally fits into the decision problem framework because the goal is not to produce a classification but to make decisions that maximize rewards. An unsupervised learning task is used to find the hidden structure in unlabeled data.

10.2.3 Learning processes

Learning is one of the most important features of the ANN. Every neural network possesses knowledge that is contained in the values of the connection weights. Most of the ANN contains some form of learning rule that modifies the weights of the connections according to the input patterns that it is presented with. Although there are various kinds of learning rules used by neural networks, the delta learning rule is often utilized by the most common class of ANNs called back-propagation neural networks. Following are the common error back-propagation learning algorithm or delta learning rule for ANN.

1. Hebbian learning rule
2. Perceptron learning rule
3. Widrow—Hoff learning rule
4. Winner-Take-All learning rule

For more depth knowledge and other rules, readers may refer to the resources given in the reference section.

Error back-propagation learning algorithm or delta learning rule is one of the commonly used learning rules. This learning algorithm is valid for continuous activation function and is used in the supervised/unsupervised training method. A simple perceptron can handle linearly separable or linearly independent problems. Taking the partial derivative of error of the network with respect to each of its weights, one may know the flow of error direction in the network. If the negative derivative is taken and proceeds to add it to the weights, then the error will decrease until it approaches a local minimum. One has to add a negative value to the weight or the reverse if the derivative is negative. Then, these partial derivatives are applied to each of the weights, starting from the output layer weights to the hidden layer weights and then from the hidden layer weights to the input layer weights. Generally, training of the network involves feeding samples as input vectors, calculating the error of the output layer and then adjusting the weights of the network to minimize the error. The average of all the squared errors E for the outputs is computed to make the derivative simpler. After the error is computed, the weights can be updated one after another. The descent depends on the gradient ∇E for the training of the network.

Consider a multilayer neural architecture containing one input node x, three nodes in the hidden layer y_j, $j=1, 2, 3$ and one output O. By applying feed-forward recall with error back-propagation learning to the model, one has the following algorithm.

Algorithm 10.6.

Step 1: Initialization

Initialize the weights W from the input layer to the hidden layer and weights V from the hidden layer to the output layer.

Choose the learning parameter $\eta \in [0, 1]$ and error E_{max}.

Take the initial value of error $E=0$.

Step 2: Training

Outputs of the hidden layer and the output layer are computed as follows:

$$y_j \leftarrow f\left(w_j x\right),\ j=1,\ 2,\ 3 \quad o_k \leftarrow f(v_k y),\ k=1 \tag{10.10}$$

where

w_j is the jth row of W for $j=1,\ 2,\ 3$

v_k is the kth row of V for $k=1$

f is the activation function.

Step 3: Error computation

The error is computed by the following manner:

$$E=\frac{1}{2}(d_k-o_k)^2+E \tag{10.11}$$

where

d_k is the desired output

o_k is the output of ANN

Step 4: Error for output and hidden layers

The error terms of the output and hidden layers are computed as follows:

$$\delta_{ok} = \left[(d_k o_k) f^{'}(v_k y)\right] \text{(error of the output layer)}$$

$$\delta_{yj} = \left[\left(1 - y_j\right) f\left(w_j x\right)\right] \delta_{ok} v_{kj} \text{ (error of the hidden layer)} \quad (10.12)$$

where $o_k = f(v_k y), j = 1, 2, 3$ and $k = 1$.

Step 5: Computation of error gradients

The error gradient vectors are computed as follows:

$$\frac{\partial E}{\partial w_{ji}} = \delta_{yj} x_i \text{for} j = 1, 2, 3 \text{ and } i = 1, \quad (10.13)$$

$$\frac{\partial E}{\partial v_{kj}} = \delta_{ok} y_j \text{for} j = 1, 2, 3 \text{ and } k = 1. \quad (10.14)$$

Step 6: Modification of weights

The weights are modified by using the gradient descent method from the input layer to the hidden layer and from the hidden layer to the output layer as follows:

$$w_{ji}^{n+1} = w_{ji}^{n} + \Delta w_{ji}^{n} = w_{ji}^{n} + \left(-\eta \frac{\partial E}{\partial w_{ji}^{n}}\right) \quad (10.15)$$

$$v_{kj}^{n+1} = v_{kj}^{n} + \Delta v_{kj}^{n} = v_{kj}^{n} + \left(-\eta \frac{\partial E}{\partial v_{kj}^{n}}\right) \quad (10.16)$$

where

η is the learning parameter

n is the iteration step

E is the error function.

Step 7: Termination

If the error $E = E_{max}$, then stop the training. Else, go to Step 2 with $E \leftarrow 0$ and initiate new training.

The generalized delta learning rule propagates the error back by after another layer, allowing the same process to be repeated for every layer.

10.2.4 Activation functions

Activation is defined as a function that acts upon the net (input) to get the output of the network. It translates input signals (information) to output signals (information). It acts as a squashing function such that the output of the neural network lies between certain values (usually between 0 and 1 or -1 and 1). Five types of activation functions are commonly used.

1. Unit step (threshold) function
2. Piecewise linear function
3. Gaussian function
4. Sigmoid function
   a. Unipolar sigmoid function
   b. Bipolar sigmoid function
5. Tangent hyperbolic function.

The first three functions are very common. Following are the remaining two functions.

1. Sigmoid Function

The sigmoid function is defined as a strictly increasing and continuously differentiable function. It exhibits a graceful balance between linear and nonlinear behavior.

a. Unipolar sigmoid function

The unipolar sigmoid function is defined by the following formula:

$$f(x) = \frac{1}{1 + e^{-\lambda x}} \tag{10.17}$$

where $\lambda > 0$ is the slope of the function.

The output of the unipolar sigmoid function lies in $[0, 1]$.

b. Bipolar sigmoid function

The bipolar sigmoid function is formulated as follows:

$$f(x) = \frac{2}{1 + e^{\lambda x}} - 1 = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}} \tag{10.18}$$

The output of the bipolar sigmoid function lies between $[-1, 1]$.

2. Tangent hyperbolic

The tangent hyperbolic function is defined as follows:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{10.19}$$

The output of the tangent hyperbolic function lies in $[-1, 1]$ and the graph of a tangent hyperbolic function is the same as bipolar sigmoid function.

10.2.5 Applications of ANN in optimization

The application of ANNs to optimization problems has been an active area of research since the early eighties. The research work has shown that ANNs are nonlinear dynamic systems from the system theory point of view. A neural network with the following properties in the state space of interest can perform the task of system optimization.

1. Every network trajectory always converges to a stable equilibrium point.
2. Every state equilibrium point corresponds to an optimal solution to the problem.
3. The first property guarantees that given an initial point to the network, the ensuing network trajectory leads to a stable steady state. The second property ensures that every steady state of the network is a solution of the underlying optimization problem. A sufficient condition for a neural network to possess the first property is the existence of the energy function associated with the network. The second property can be relaxed such that the state of every stable equilibrium point is close to an optimal solution point of the problem. Parallel analog computations in a network of parallel interconnected neurons provide high computational power and speed. The ability of processing feedback in a collective parallel analog mode enables a neural network to simulate the dynamics that represent the optimization of an objective function subjected to its constraints for a given optimization model. Research articles have been published with the development of neural networks applicable to general nonlinear, constrained optimization problems including linear programming and nonlinear programming, and combinatorial optimization problems (COPs). Different neural networks have been analyzed from the viewpoint of both the optimization theory and dynamical system theory.

10.3 Ant colony optimization

In this section, an overview of a metaheuristic ant colony optimization (ACO) method is presented. This method is inspired by the behavior of real ants. Initially, for solving complicated Combinatorial Optimization Problems (COPs), researchers proposed ACO. ACO is one of the branches of swarm intelligence, and ACO algorithms are inspired by the behavior of swarms. Swarm optimization algorithms are constituted up of simple individuals that cooperate through self-organization. The idea came to the researchers when the social behavior of insects was observed. In the 1940s and 1950s, it is seen that insects are capable to react to what he called "significant stimuli," signals that activate a genetically encoded reaction. Then, further studies showed that the effects of these reactions can act as new significant stimuli for both the insect that produced them and for the other insects in the colony. Hence, it was used to describe this particular type of indirect communication in which the workers are stimulated by the performance they have achieved.

The two main characteristics of indirect coordination through the environment between agents or actions that differentiate it from other means of communication are as follows:

1. The physical, nonsymbolic nature of the information released by the communicating insects, which corresponds to a modification of physical environmental states visited by the insects.
2. The local nature of the released information, which can be accessed only by those insects that visit the place where it was released.

Examples of indirect coordination through the environment between agents or actions can be seen in colonies of ants. In many ant species, ants walking to and from a food source deposit on the ground a substance called pheromone. Other ants can smell this pheromone, and its presence influences the choice of their path, that is, they tend to follow strong pheromone concentrations. The pheromone deposited on the ground forms a pheromone trail, which allows the ants to find good sources of food that have been previously identified by other ants.

Some researchers investigated experimentally this pheromone laying and the following behavior to better understand it and to be able to quantify it. An experiment called a "binary bridge experiment" was also performed for the same. For that experiment, *Linepithema humile* ants were taken. The ants' nest was connected to a food source by two bridges of equal length. The ants could freely choose which bridge to use when searching for food and bringing it back to the nest. Their behavior was then observed over a period of time. In this experiment, initially, there is no pheromone on the two bridges. The ants start exploring the surroundings of the nest and eventually cross one of the bridges and reach the food source. When walking to the food source and back, the ants deposit pheromone on the bridge they use. Initially, each ant randomly chooses one of the bridges. However, because of random fluctuations, after some time there will be more pheromone deposited on one of the bridges than on the other. Because ants tend to prefer in probability to follow a stronger pheromone trail, the bridge that has more pheromone will attract more ants. This, in turn, makes the pheromone trail grow stronger until the colony of ants converges toward the use of the same bridge. This colony-level behavior, based on autocatalysis, that is, on the exploitation of positive feedback, can be exploited by ants to find the shortest path between a food source and their nest. This was demonstrated in another experiment where the two bridges were not of the same length, and one was significantly longer than the other. In this case, the stochastic fluctuations in the initial choice of a bridge were much reduced as a second mechanism played an important role. Those ants choosing by chance the shorter bridge were also the first to reach the nest, and when returning to the nest, they chose the shorter bridge with higher probability as it had a stronger pheromone trail. Therefore, it is noted that the pheromone following and depositing mechanism quickly converged to the use of the shorter bridge.

This basic model, which explains the behavior of real ants, may be used as an inspiration to design artificial ants that solve optimization problems defined similarly. In the aforementioned ant foraging behavior, for example, stigmergic (indirect coordination through the environment between agents or actions) communication happens via the pheromone that ants deposit on the ground. Analogously, artificial ants may simulate pheromone laying by modifying appropriate pheromone variables associated with problem states they visit while building solutions to the optimization problem. Also, according to the stigmergic communication model, artificial ants would have only local access to these pheromone variables. Therefore, the main characteristics of stigmergy can be extended to artificial agents by

1. Associating state variables with different problem states.
2. Giving the agents only local access to these variables.

Another important aspect of real ants' foraging behavior that may be exploited by artificial ants is the coupling between the autocatalytic mechanism and the implicit evaluation of solutions. By implicit solution evaluation, we mean the fact that shorter paths are completed earlier than longer ones, and therefore, they receive pheromone reinforcement quicker. Implicit solution evaluation coupled with autocatalysis can be very effective. The shorter the path, the sooner the pheromone is deposited, and the more ants use the shorter path. If appropriately used, it can be a powerful mechanism in population-based optimization algorithms (e.g., in evolutionary algorithms, autocatalysis is implemented by the selection or reproduction mechanism). Stigmergy, together with implicit solution evaluation and autocatalytic behavior, gave rise to ACO. The basic idea of ACO follows very closely the biological inspiration. Therefore, there are many similarities between real and artificial ants. Both real and artificial ant colonies are composed of a population of individuals that work together to achieve a certain goal. A colony is a population of simple, independent, asynchronous agents that cooperate to find a good solution to the problem at hand. In the case of real ants, the problem is to find the food, while in the case of artificial ants, it is to find a good solution to a given optimization problem. A single ant (either a real or an artificial one) can find a solution to its problem, but only cooperation among many individuals through stigmergy enables them to find good solutions. In the case of real ants, they deposit and react to a chemical substance called pheromone. Real ants simply deposit it on the ground while walking. Artificial ants live in a virtual world, and hence, they only modify numeric values (called for analogy artificial pheromones) associated with different problem states. A sequence of pheromone values associated with problem states is called an artificial pheromone trail. In ACO, the artificial pheromone trails are the sole means of communication among the ants. A mechanism analogous to the evaporation of the physical pheromone in real ant colonies allows the artificial ants to forget the history and focus on new promising search directions. Just like real ants, artificial ants create their solutions sequentially by moving from one problem state to another. Real ants simply walk, choosing a direction based on local pheromone concentrations and a stochastic decision policy. Artificial ants also create solutions step by step, moving through available problem states and making stochastic decisions at each step. The following are some of the important differences between real and artificial ants.

1. Artificial ants live in a discrete world—they move sequentially through a finite set of problem states.
2. The pheromone update (i.e., pheromone depositing and evaporation) is not accomplished in the same way by artificial ants as by real ones. Sometimes the pheromone update is done only by some of the artificial ants and often only after a solution has been constructed.

3. Some implementations of artificial ants use additional mechanisms that do not exist in the case of real ants such as look-ahead, local search, backtracking, etc.

Algorithm 10.7.

```
Step 1: Set the parameters and initialize pheromone trails.
Step 2: Assign the termination conditions.
Step 3: Construct an ant solution.
Step 4: Apply local search (if any).
Step 5: Update pheromones. Go to Step 2.
If the termination conditions not satisfied, then go to Step 3. Else,
   terminate.
```

The ACO metaheuristic is shown in Algorithm 10.7. It consists of an initialization step and a loop over three algorithmic components. A single iteration of the loop consists of constructing solutions by all ants, their (optional) improvement with the use of a local search algorithm, and an update of the pheromones. In the following, we explain these three algorithmic components in detail.

1. Construct ant solutions

A set of m artificial ant construct solutions from elements of a finite set of available solution components $S = \{c_{ij}\}$, $i = 1, 2, \cdots, n; j = 1, 2, \cdots, |D_i|$. A solution construction starts with an empty partial solution ϕ. Then, at each construction step, the current partial solution is extended by adding a feasible solution component from the set of feasible neighbors $N(\phi) \subseteq S$. The process of constructing solutions can be regarded as a path on the construction graph $G_C = (V, E)$. The allowed path in G_C are implicitly defined by the solution construction mechanism that defines the set $N(\phi)$ with respect to a partial solution ϕ.

The choice of a solution component from $N(\phi)$ is done probabilistically at each construction step. The exact rules for the probabilistic choice of solution components vary across different ACO variants. The best-known rule is the one of ant system:

$$p(c_{ij}|\phi) = \frac{\tau_{ij}^{\alpha} \cdot \eta(c_{ij})^{\beta}}{\sum_{c_{ij} \in N(\phi)} \tau_{ij}^{\alpha} \cdot \eta(c_{ij})^{\beta}}, \forall c_{ij} \in N(\phi) \tag{10.20}$$

where τ_{ij} is the pheromone values associated with the component c_{ij}, and $\eta(\cdot)$ is a function that assigns at each construction step a heuristic value to each feasible solution component $c_{ij} \in N(\phi)$. The values that are returned by this function are commonly called heuristic information. Furthermore, α and β are positive parameters, whose values determine the relative importance of pheromone versus heuristic information.

2. Apply local search

Once solutions have been constructed, and before updating pheromones, often some optional actions may be required. These are often called daemon actions and can be used to implement problem specific and/or centralized actions, which cannot be performed by single ants. The most used daemon action consists in the application of local search to the constructed solutions. The locally optimized solutions are then used to decide which pheromones to update.

3. Update pheromones

The pheromone update aims to increase the pheromone values associated with good or promising solutions and to decrease those that are associated with bad ones. Usually, this is achieved (1) by decreasing all the pheromone values through pheromone evaporation and (2) by increasing the pheromone levels associated with a chosen set of good solutions.

10.4 Particle swarm optimization

This section starts with an example of the behavior of birds. Suppose there is a group of birds and all the birds are hungry, they are searching for food. These hungry birds can be correlated with the tasks in a computation system, which are hungry for resources. Now, in the locality of these birds, there is only one food particle. This food particle can be correlated with a resource. As we know, tasks are many, and resources are limited. So this has become a similar condition as in a certain computation environment. Now, the birds don't know where the food particle is hidden or located. In such a scenario, how the algorithm to find the food particle should be designed. If every bird will try to find the food on its own, it may cause devastation and may consume a large amount of time. Thus, on careful observation of these birds, it was realized that though the birds don't know where the food particle is located, they do know their distance from it. Thus, the best approach to finding that food particle is to follow the birds that are nearest to the food particle. This behavior of birds is simulated in the computation environment, and the algorithm so designed is termed as particle swarm optimization (PSO) algorithm.

PSO algorithm is a population-based search algorithm that relies on the simulation of the social behavior of birds within a flock. The initial intent of the particle swarm concept was to graphically simulate the graceful and unpredictable choreography of a bird flock, to discover patterns that govern the ability of birds to fly synchronously, and to suddenly change direction with regrouping in an optimal formation. From this initial objective, the concept evolved into a simple and efficient optimization algorithm. In PSO, individuals, referred to as particles, are flown through high-dimensional search space. Changes to the position of particles within the search space are based on the social-psychological tendency of individuals to emulate the success of other individuals. The changes to a particle within the swarm are therefore influenced by the experience, or knowledge, of its neighbors. The search behavior of a particle is thus affected by that of other particles within the swarm (PSO is, therefore, a kind of symbiotic cooperative algorithm). The consequence of modeling this social behavior is that the search process is such that particles stochastically return toward previously successful regions in the search space.

Individuals in a particle swarm follow a very simple behavior to emulate the success of neighboring individuals and their successes. The collective behavior that emerges from this simple behavior is that of discovering optimal regions of a high-dimensional search space. The PSO algorithm maintains a swarm of particles, where each particle represents a potential solution. In analogy with evolutionary computation paradigms, a swarm is similar to a population, while a particle is similar to an individual. In simple terms, the particles are flown through a multidimensional search space, where the position of each particle is adjusted according to its own experience and that of its neighbors.

Let $x_i(t)$ denote the position of particle i in the search space at time step t.

Unless otherwise stated, t denotes discrete time steps. The position of the particle is changed by adding a velocity $v_i(t)$ to the current position, that is,

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{10.21}$$

with $x_i(0) \sim U(x_{min}, x_{max})$.

It is the velocity vector that drives the optimization process and reflects both the experiential knowledge of the particle and socially exchanged information from the particle's neighborhood. The experiential knowledge of a particle is generally referred to as the cognitive component, which is proportional to the distance of the particle from its own best position (referred to as the particle's personal best position) found since the first time step. The socially exchanged information is referred to as the social component of the velocity equation.

The basic algorithm to form PSO is mentioned in Algorithm 10.8.

Assume,

x_k^i is particle position,

v_k^i is particle velocity,

p_k^i is best-remembered individual particle position,

p_k^g is best-remembered swarm position,

c_1, c_2 are cognitive and social parameters, and

r_1, r_2 are random numbers between 0 and 1.

Position of individual particles is updated as follows:

$$x_{k+1}^i = x_k^i + v_{k+1}^i, \tag{10.22}$$

with the velocity calculated as follows:

$$v_{k+1}^i = v_k^i + c_1 r_1 \left(p_k^i - x_k^i\right) + c_2 r_2 \left(p_k^g - x_k^i\right). \tag{10.23}$$

Algorithm 10.8.

```
Step 1: Initialization
Set constraints $k_{max}, c_1, c_2$.
Randomly initialize particle positions $x_0^i \in D$ in $R^n$ for particles
    $i = 1, 2, \cdots, p$.
Randomly initialize particle velocities $0 \leq v_0^i \leq v_0^{max}$ for particles
    $i = 1, 2, \cdots, p$.
Set $k = 1$.
Step 2: Optimization
Evaluate function value $f_k^i$ using design space coordinates $x_k^i$.
If $f_k^i \leq f_{\text{best}}^i$, then $f_{\text{best}}^i = f_k^i, p_k^i = x_k^i$.
If $f_k^i \leq f_{\text{best}}^g$, then $f_{\text{best}}^g = f_k^i, p_k^g = x_k^i$.
If stopping condition is satisfied, then go to Step 3.
Update all particles velocities $v_k^i$ for $i = 1, 2, \cdots, p$.
Update all particles positions $x_k^i$ for $i = 1, 2, \cdots, p$.
```

Increment *k*.
Go to Step 2
Step 3: Terminate

Following are the advantages of PSO:

1. Insensitive to scaling of design variables.
2. Simple implementation.
3. Easily parallelized for concurrent processing.
4. Derivative free approach.
5. Very few algorithm parameters.
6. Very efficient global search algorithm.

The main disadvantage is the slow convergence in a refined search stage that is weak local searchability.

Following are the applications of PSO:

1. Training of neural networks.
2. Identification of Parkinson's disease.
3. Extraction of rules from fuzzy networks.
4. Image recognition.
5. Optimization of electric power distribution networks.
6. Structural optimization.
7. Optimal shape and sizing design.
8. Topology optimization.
9. Process biochemistry.
10. System identification in biomechanics.

PSO is also characterized in the domain of AI. The term "artificial intelligence" refers to the theory of simulating human behavior through computation. It involves designing such computer systems that can execute tasks that require human intelligence. For example, earlier only humans had the power to recognize the speech of a person. But now, speech recognition is a common feature of any digital device. This has become possible through AI. Other examples of human intelligence may include decision-making, language translation, and visual perception. Various techniques make it possible. These techniques to implement AI into computers are popularly known as methods of AI.

Further reading

Anderson, J. (1995). *An introduction to neural networks*. Cambridge, London: MIT Press.
Bullnheimer, B., Hartl, R., & Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. *Annals of Operatins Research*, 312.
Chakraverty, S., & Mall, S. (2017). *Artificial neural networks for engineers and scientists: Solving ordinary differential equations*. Boca Raton, FL: CRC Press.

Chakraverty, S., & Nayak, S. (2013). Non-probabilistic solution of uncertain neutron diffusion equation for imprecisely defined homogeneous bare reactor. *Annals of Nuclear Energy*, 251−259.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *26*(1), 29.

Eberhart, R., Simpson, P., & Dobbins, R. (1996). *Computational intelligence PC tools*. Boston, MA: Academic Press Professional.

Fogel, D. (1995). *Evolutionary computation*. New York, NY: IEEE Press.

Freeman, J., & Skapura, D. (1991). *Neural networks: Algorithms, applications, and programming techniques*. Boston, MA: Addison-Wesley Publishing Company.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Graupe, D. (2007). *Principles of artificial neural networks*. Toh Tuck Link, Singapore: World Scientific Publishing Co. Ltd.

Haykin, S. (1999). *Neural networks: A comprehensive foundation*. Upper Saddle River, NJ: Prentice Hall Inc.

Herrera, F., Lozano, M., & Verdegay, J. (1998). Tackling realcoded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review*, 265−319.

Hinton, G., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 1527−1554.

Holland, J. (1992). *Adaptation in natural and artificial systems*. Cambridge, MA: The MIT Press.

Kennedy, J., Eberhart, R., & Shi, Y. (2001). *Swarm intelligence*. San Francisco, CA: Morgan Kaufmann Publishers.

Minsky, M., & Papert, S. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Naka, S., Grenji, T., Yura, T., & Fukuyama, Y. (2001). *Practical distribution state estimation using hybrid particle swarm optimization. Proceedings of the IEEE PES Winter Meeting*. OH: Columbus.

Nayak, S., & Chakraverty, S. (2015). Numerical solution of uncertain neutron diffusion equation for imprecisely defined homogeneous triangular bare reactor. *Sadhana*, 2095−2109.

Nayak, S., & Chakraverty, S. (2016). Numerical solution of stochastic point kinetic neutron diffusion equation with fuzzy parameters. *Nuclear Technology*, 444−456.

Otten, R., & Van Ginneken, L. (1989). *The annealing algorithm*. Boston, MA: Kluwer Academic Publishers.

Schalkoff, R. (1997). *Artificial neural networks*. New York, NY: McGraw-Hill.

Schwefel, H. (1995). *Evolution and optimum seeking. Sixth-generation computer technologie series*. New York, NY: JohnWiley & Sons.

Wright, A. (1991). Genetic algorithms for real parameter optimization. In G. Rawlin, & E. Kaufmann (Eds.), *Foundations of genetic algorithms* (1, pp. 205−218). .

Zurada, J. (1992). *Introduction to artificial neural systems*. St. Paul, MN: West Publishing Co..

Index

Note: Page numbers followed by "*f*" and "*t*" refer to figures and tables, respectively.

Fundamentals of Optimization Techniques with Algorithms

Sukanta Nayak

Optimization is a key concept in mathematics, computer science, and operations research, and is essential to the modeling of any system, playing an integral role in computer-aided design. *Fundamentals of Optimization Techniques With Algorithms* presents a complete package of various traditional and advanced optimization techniques along with a variety of example problems, algorithms, and MATLAB© codes, for linear and nonlinear single variable and multivariable models, as well as multiobjective and advanced optimization techniques. It presents both theoretical and numerical perspectives in a clear and approachable way. To help the reader for applying optimization techniques in practice, the book includes detailed program codes and computer-aided designs concerning real-world problems. Ten chapters cover an introduction to optimization, linear programming, single variable nonlinear optimization, multivariable unconstrained nonlinear optimization, multivariable constrained nonlinear optimization, geometric programming, dynamic programming, integer programming, multiobjective optimization, and nature-inspired optimization. This book provides accessible coverage of optimization techniques and helps the reader to apply them in practice.

- Presents optimization techniques, including worked-out examples, from traditional to advanced
- Maps out the relations between optimization and other mathematical topics and disciplines
- Provides systematic coverage of algorithms to facilitate computer coding
- Gives MATLAB© codes concerning optimization techniques and their use in computer-aided design
- Presents nature-inspired optimization techniques including genetic algorithms and artificial neural networks

About the editor

Dr. Sukanta Nayak is an Assistant Professor in the Department of Mathematics, at the Amrita School of Engineering, Amrita Vishwa Vidyapeetham in Coimbatore, India. He previously held a postdoctoral research fellowship at the University of Johannesburg, South Africa, and received his PhD in mathematics from the National Institute of Technology Rourkela, in India. His research interests include numerical analysis, linear algebra, fuzzy finite element method, fuzzy heat, neutron diffusion equations, fuzzy stochastic differential equations, and wavelet analysis. He has published widely in the field, including as co-author of a book entitled *Interval Finite Element Method with MATLAB,* for Elsevier's Academic Press (2018).

Technology and Engineering / General with Algorithms

ISBN 978-0-12-821126-7

9 780128 211267

ELSEVIER

ACADEMIC PRESS
An imprint of Elsevier
elsevier.com/books-and-journals