DE GRUYTER

Elias Munapo, Santosh Kumar

LINEAR INTEGER PROGRAMMING

THEORY, APPLICATIONS, RECENT DEVELOPMENTS

DE GRUYTER SERIES ON THE APPLICATIONS OF MATHEMATICS IN ENGINEERING AND INFORMATION SCIENCES

Elias Munapo, Santosh Kumar

Linear Integer Programming

De Gruyter Series on the Applications of Mathematics in Engineering and Information Sciences

Edited by
Mangey Ram

Volume 9

Elias Munapo, Santosh Kumar

Linear Integer Programming

Theory, Applications, Recent Developments

DE GRUYTER

Authors
Prof. Elias Munapo
North West University
Dept. of Business Statistics &
Operations Research
Dr Albert Luthuli Drive
Mmabatho
2745
South Africa
Elias.Munapo@nwu.ac.za

Prof. Santosh Kumar
RMIT University
School of Mathematical and Geospatial Sciences
Melbourne
Australia
Santosh.Kumarau@gmail.com

ISBN 978-3-11-070292-7
e-ISBN (PDF) 978-3-11-070302-3
e-ISBN (EPUB) 978-3-11-070311-5
ISSN 2626-5427

Library of Congress Control Number: 2021946656

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.

Cover image: MF3d/E+/Getty Images
Typesetting: Integra Software Services Pvt. Ltd.
Printing and binding: CPI books GmbH, Leck

www.degruyter.com

To our families, and past, present, and future students
EM and SK

Acknowledgements

The work in this book is based on our recently published papers in various journals. We wish to express our sincere gratitude to many anonymous referees who raised searching questions and made suggestions for improving the paper. We learn from their comments and remarks, and while improving the presentation of the paper, we also improved our own understanding of the subject. We are thankful to these referees, to our students and to our colleagues who redirected us on the right path, when we found ourselves a bit lonely and off the track.

The authors would also like to express sincere thanks to their respective universities, i.e., the North West University, South Africa and the RMIT University, Australia.

We also want to say ‘thank you’ to our families for their patience and understanding.

Elias Munapo
Santosh Kumar OAM

https://doi.org/10.1515/9783110703023-202

Preface

'Hide-and-seek' is a universal game played by children all over the world. Irrespective of their background, nationality, and race; the game enters everyone's life in one or the other form. For example, on Easter, children enjoy playing the 'Hide-and-Seek' with Easter eggs. Many of us associate the 'Hide-and-seek phenomenon' only with children, but a closer examination reveals that our lives are full of 'hides-and-seeks' in the form of decision making. Good decisions hide and one is required to seek them. These good-decisions on one hand can bring prosperity, however, danger associated with a bad-decision also stays with us. These decisions in some commerce, business, and industrial environments can be a matter of life and death. Some of these situations are so complex that we make use of mathematical modelling tools to examine their complexities and seek better outcomes from the situation. Therefore, this 'Hide-and-seek' game never ends; and one is surrounded by the consequences of decisions. The outcome of a bad decision can challenge the very existence of the company. Mathematical programming is an abstract form of the 'Hide-and-Seek' game, where good decisions are hiding and with the help of mathematical rigour, we try to unearth the good decisions.

This book is an attempt to develop some strategies to unearth good decisions for a class of problems that can be represented in the form of a mathematical model of the form:

$$Max\ z = f(x) \tag{P1}$$

Subject to

$$g_i(x) = \sum_{j=1}^{n} a_{ij} x_j \le b_i,\ \ i = 1,\ 2,\ \ldots,\ m \tag{P2}$$

$$x_j \ge 0 \ and\ integer,\ \ j = 1,\ 2,\ \ldots,\ n$$

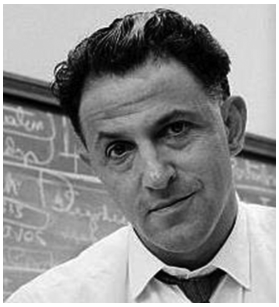
Here, we discuss only those cases where $f(x)$ *and* $g_i(x)$ are linear functions and x_j are restricted to integer values. The authors have developed some new strategies during their journey in the field of integer programming. Many pioneering ideas have existed, and where possible, we added a bit more during our journey in this fascinating field. The journey will never end; it will continue.

This book outlines some of our attempts in this fascinating field of mathematical programming.

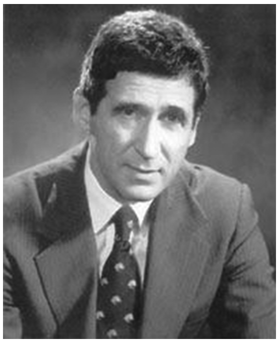
Before we conclude this preface, we briefly recall a few pioneers in the general field of Mathematical Programming. It is difficult to do proper justice to their contributions, however, our tribute is just a symbol of our respect for them.

https://doi.org/10.1515/9783110703023-203

First let us recall the father of linear programming, George B. Dantzig (8-11-1914 to 13-5-2005) an American mathematician, Professor Emeritus of Operations Research and Computer Science who made significant contributions to industrial engineering, operations research, computer science, economics, and statistics. He developed the LP model and devised the 'Simplex Method' for solving the LP model. This mathematical model has found immense applications in various fields and helped decision makers. The integer programming is a variation of a LP, where variables are restricted to integer values. Most approaches dealing with integer programming commence the integer optimal search from the LP optimal solutions.



While linear programming was being developed, Richard Bellman (26-8-1920 to19-3-1984) was developing dynamic programming. He was at Rand Corporation, where the concept of dynamic programming was developed. Later he moved to University of Southern California, Los Angeles. The dynamic programming has applications in many real-life industrial engineering and control engineering processes. The dynamic programming creates a new approach to problem solving, fundamentally it is a process of solving a 'n' dimensional problem as 'n' problems of one dimension each. It is a simple but very powerful concept, and it is an art of problem solving.



A transition from linear programming to integer programming occurred by the pioneering work of Ralph E. Gomory (born 7 -5 -1929). He is an American applied mathematician and served as Chairman of IBM Mathematical Sciences Department, Director of Research for the IBM and a research professor at New York University.

The above scientists along with many other researchers have developed the fascinating filed of mathematical programming, that has unending applications in industry and commerce. The authors have made a few advances in this fascinating field. More results will continue to make the field richer and helpful to its users in various decision-making situations.

Elias Munapo and Santosh Kumar OAM

Contents

About the authors

Elias Munapo is a Professor of Operations Research at the North West University, Mafikeng Campus in South Africa. He is a guest editor of the Applied Sciences journal, has published two books, edited several books, a reviewer for several journals. He has published a significant number of journal articles and book chapters. In addition, he has presented at both local and international conferences and has supervised doctoral students to completion. His research interests are in the broad area of operations research.

Prof Kumar OAM is author and co-author of 197 papers and 3 books in the field of Operations Research. His contributions to the field of OR have been recognised in the form of 'Ren Pot Award' from the Australian Society for Operations Research (ASOR) in 2009 and a recognition award from the South African OR society as a non-member of the society in 2011. He has served as the President of the Asia Pacific Operations Research societies (1995–97), where ASOR was a member along with China, India, Japan, Korea, Malaysia, New Zealand, and Singapore. He is currently an Adjunct Professor at the RMIT University, Melbourne. He is a Fellow of the Institute of Mathematics and its Applications, UK. Recently, on Queen's birthday, 14 June 2021 he was awarded a Medal of the Order of Australia.

https://doi.org/10.1515/9783110703023-205

Chapter 1
Segment search approach for the general linear integer model

Abstract: This chapter presents a segment-search approach for the general linear integer problem. In this approach the feasible region is divided into segments and searched for the optimal solution. The segments are searched in their ascending or descending order depending on the nature of the problem and a solution feasible in the kth segment is also optimal to the whole problem. This approach has the strength that large numbers of sub-problems that usually result in large linear problems can now be managed easily in this way.

Keywords: Linear integer problem, Sub-problems, Segment-search, Feasible region, Variable limits

1.0 Introduction

The general linear integer programming model (LIP) is one of the most difficult class of problems in combinatorial optimization and is in the *NP*-complete class of hard problems. This is a linear programming problem in which variables are restricted to integer values. There are several techniques that are available that can be used to solve this problem. These include heuristics, use of cuts, branch and bound (BB) [see Gomory (1958), Taha (2017), Land and Doig (1960), Dakin (1965)], pricing, hybrids such as branch and cut [Brunetta, Conforti and Rinaldi (1977), Mitchell (2001), Padberg and Rinaldi(1991)], branch and price [Barnerd et al. (1998), Salvelsbergh (1997)], branch price and cut [Barnhart et al. (2000), Fukasawa et al. (2006), Ladanyi et al. (2001)]. A linear relation to identify the required integer optimal solution was developed by the authors, [Kumar, Munapo and Jones (2007), Kumar and Munapo (2012)]. However, up to now (December 2020) there is no efficient and consistent general known approach that has been developed to solve this difficult model. The branch and bound is one the earliest approaches in the history of integer programming and is still receiving attention from researchers [Munapo and Kumar (2016), Al-Rabeeah et al. (2019)], and Munapo (2020)].

In this chapter, the feasible region is divided into segments and searched for the optimal solution. The segments are searched in their ascending or descending order depending on the nature of the problem and a feasible solution in the kth segment is also optimal to the whole problem. This approach has the strength that large numbers of sub-problems that usually result in large linear integer model can now be managed easily in this way.

https://doi.org/10.1515/9783110703023-001

1.1 The Linear Integer Programming (LIP) Model and some preliminaries

A standard linear integer programming model is given by:
Maximize

$$Z = \sum_j c_j x_j$$

Such that

$$\sum_i \sum_j a_{ij} x_j \le b_i \tag{1.1}$$

Where a_{ij}, b_i, c_j are constants, $x_j \ge 0$ and integer.

$$i = 1, 2, \ldots, m \;\; and \;\; j = 1, 2, \ldots, n$$

Earlier approaches solved the relaxed linear integer model as a LP and modified the LP optimal solution to an IP optimal solution in different ways, see Kumar et al. (2010). Instead of solving the relaxed integer linear model, an alternative possibility is to determine some very useful additional constraints, which can be in the form of:
(i) original variable sum constraint,
(ii) slack and excess variable sum constraint, and
(iii) slack and excess variable limits.

These approaches are discussed below.

The original variable sum constraint can be in the form of equation (1.2).

$$x_1 + x_2 + \ldots + x_n - \phi_c = 0. \tag{1.2}$$

Where ϕ_c is not necessarily integer.

The excess and or slack variable sum constraint can be in the form of equation (1.3)

$$s_1 + s_2 + \ldots + s_m - \lambda = 0 \tag{1.3}$$

Where λ is an integer.

The slack and excess variable limits are generated from the given constraints in the given model (1.1). This is easily done using the following procedure.

From each column of the coefficient matrix select the largest coefficient c_j^L. Then from the selected values from all the columns, select the minimum value as slack and or excess variable limit ℓ. as given by equation (1.4).

$$0 \le s_j \le (\ell - 1), \forall \text{ slack or excess variables.} \tag{1.4}$$

The integer restrictions can be relaxed and the given model (1.1) can be solved as an ordinary LP. The continuous optimal solution obtained will be in the form as given in Table 1.1.

Table 1.1: The general continuous optimal tableau.

	x_1	x_2	...	x_m	s_1	s_2	s_n	...	r.h.s.
Z	0	0	...	0	ω_1	ω_2	ω_n	...	Z_0
x_1	1	0	...	0	α_{11}	α_{12}	α_{1n}	...	β_1
x_2	0	1	...	0	α_{21}	α_{22}	α_{2n}	...	β_2
...									
x_m	0	0	...	1	α_{m1}	α_{m2}	α_{nn}	...	β_m

Where $\omega_1,\ \omega_2,\ \ldots,\ \omega_n, \beta_1, \beta_2,\ \ldots,\ \beta_m$ are non-negative constants, α_{ij} and Z_0 are constants. Z_0 is a continuous optimal value. The variables $x_1\ x_2 \ldots x_m$ are basic and $s_1\ s_2 \ldots s_n$ are non-basic for convenience. Some other arrangements of these variables are also acceptable.

1.2 The concept of segments

From Figure 1.1, the segments are in descending order i.e.

$$Z_0 > Z_1 > Z_2 > Z_3 > \ldots > Z_k > Z_{k+1}. \tag{1.5}$$

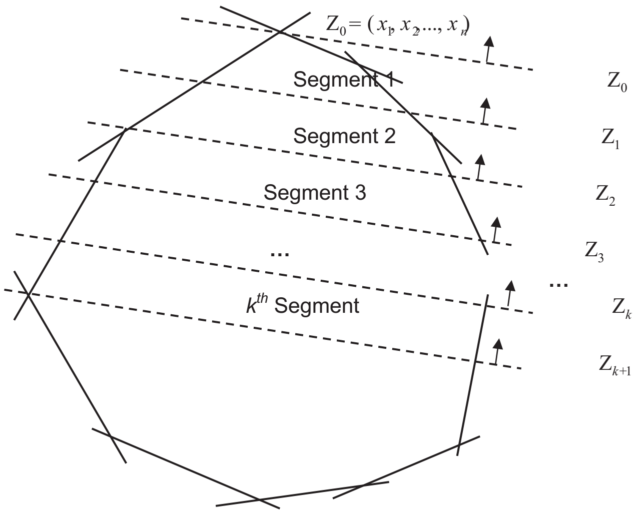


Figure 1.1: Segments of the given LP feasible region.

The idea is to search segment 1, then segment 2 in that order until the optimal solution is found in the k^{th} segment. It is easier to search a segment at a time than the whole feasible region.

1.2.1 Selection of the segment interval

There is need to select segment intervals so that the sizes of these segments are approximately equal. This may pose a challenge, and in this chapter, we present a technique to select the segment intervals. Suitable shapes that can be used to approximate the intervals are triangle, cone, pyramid etc. In this chapter a triangle shape is selected for its simplicity.

From Figure 1.1, the segments are in descending order i.e.

$$Z_0 > Z_1 > Z_2 > Z_3 > \ldots > Z_{k-1} > Z_k > Z_{k+1} \tag{1.6}$$

The idea is to search segment 1, then segment 2 in that order until the optimal solution is found. Suppose the optimal solution is obtained in the k^{th} segment, which becomes the required optimal solution. It is easier to search a segment at a time than the whole feasible region.

Using the triangle to approximate intervals

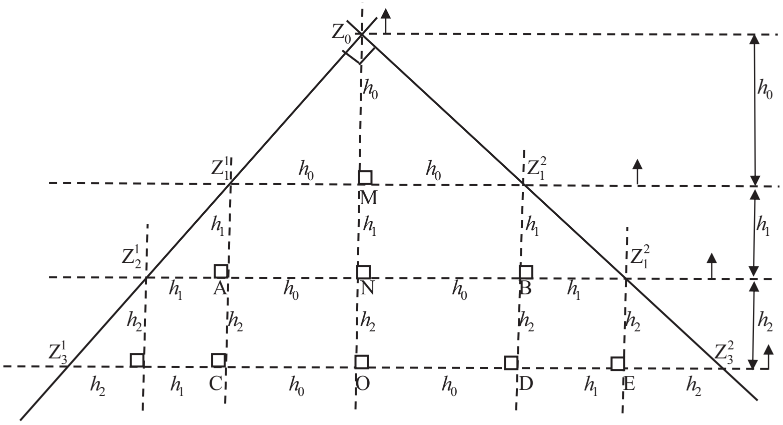


Figure 1.2: Determining the distance between intervals.

Assumption 1 - Assume we are viewing the feasible region in two dimensions.
Assumption 2 - We also assume the angle at the optimal vertex is a right angle, i.e.

$$Z_1^1\hat{Z}_0Z_1^2 = 90^o \tag{1.7}$$

Assumption 3 - Assume the surface areas of the segments are equal.
Assumption 4 - Assume decrease in the objective value from the first segment to the second segment is known and is h_0.

First Segment

The first segment is an isosceles triangle. Area is given by (1.8).

Area of First Segment (S_{A1}):

$$S_{A1} = \frac{1}{2}(h_0 + h_0)h_0 = h_0^2 \tag{1.8}$$

Second Segment

The second segment is a trapezium.

The first parallel side is $(h_0 + h_0) = 2h_0$, and the second parallel side is $(2h_0 + 2h_1)$ and the height is h_0.

Area of second segment (S_{A2}):

$$S_{A2} = \frac{1}{2}\left[2h_0 + (2h_0 + 2h_1)\right] h_1 = \frac{1}{2}(4h_0 + 2h_1)\, h_1 = (2h_0 + h_1)\, h_1. \tag{1.9}$$

Since we are assuming the areas of segments to be equal then we have from (1.8) and (1.9):

$$S_{A1} = S_{A2}.$$

$$h_0^2 = (2h_0 + h_1)h_1,$$

i.e. $$0 = -h_0^2 + 2h_0h_1 + h_1^2.$$

Solving as quadratic equation we have:

$$h_1 = -h_0 + h_0\sqrt{2}$$

$$h_1 = 0.4142h_0 \tag{1.10}$$

The third segment is also a trapezium. The first parallel side is $(2h_0 + 2h_1)$, and the second parallel side is $(2h_0 + 2h_1 + 2h_2)$ and the height is h_2.

Area of the third segment (S_{A3}):

$$S_{A3} = \frac{1}{2}[(2h_0 + 2h_1) + (2h_0 + 2h_1 + 2h_2)]h_2,$$

$$S_{A3} = 2h_0h_2 + 2h_1h_2 + h_2^2. \tag{1.11}$$

Since $h_1 = 0.4142h_0,$ one can rewrite (1.11) as

$$S_{A3} = 2h_0h_2 + 2(0.4142)h_0h_2 + h_2^2\ S_{A3} = 2.8284h_0h_2 + h_2^2.$$

Since, by assumption, areas of different segments are equal, then $S_{A1} = S_{A3}$. giving

$$h_0^2 = 2.8284h_0h_2 + h_2^2,$$

i.e.

$$0 = -h_0^2 + 2.8284h_0h_2 + h_2^2.$$

Solving as quadratic equation we have:

$$h_2 = \frac{-2.8284h_0 + h_0\sqrt{11.9998}}{2} = 0.317h_0 \tag{1.12}$$

In other words, using the proposed approach we do not decrease the objective value by the same value in all the segment. This implies when using the proposed approach there is need to accurately estimate (h_0). All the other intervals depend on h_0 as given in (1.13) and (1.14).

$$h_0, h_1, h_2, h_3, \ldots, h_k, h_{k+1} \tag{1.13}$$

i.e.

$$h_0, 0.4h_0, 0.3h_0, 0.2h_3, \ldots \tag{1.14}$$

In other words, the decrease (h_0) by a factor of approximately 0.75 from one segment to the next.

1.2.2 Decreasing the objective value

First segment

To search over the first segment, add the inequality (1.15) to the continuous optimal tableau given in Table 1.1.

$$\omega_1s_1 + \omega_2s_2 + \ldots + \omega_ns_n \le h_0. \tag{1.15}$$

Second segment

The second segment is searched by adding the following two constraints to the continuous optimal tableau.

$$\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \geq h_0. \tag{1.16}$$

$$\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \leq h_0 + h_1. \tag{1.17}$$

Third segment

Similarly, the third segment is searched by adding to the continuous optimal tableau.

$$\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \geq h_0 + h_1.$$

$$\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \leq h_0 + h_1 + h_2.$$

$$\ldots$$

Kth segment

The Kth segment is searched by adding (1.18) and (1.19) to the continuous optimal tableau.

$$\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \geq h_0 + h_1 + h_2 + \ldots + h_k \tag{1.18}$$

$$\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \leq h_0 + h_1 + h_2 + \ldots + h_k + h_{k+1} \tag{1.19}$$

1.2.3 Original variable sum limit

The original variable sum (sum of the given variables) is given in (1.20).

$$x_1 + x_2 + \ldots + x_n = \phi + f. \tag{1.20}$$

Where $\phi_c = \phi + f$ and ϕ, is the integer part and f is the fractional part.

From the sum of original variables, we develop two sub-problems A and B.

Sub-problem A:

$$x_1 + x_2 + \ldots + x_n \leq \phi. \tag{1.21}$$

Sub-problem B:

$$x_1 + x_2 + \ldots + x_n \geq \phi + 1. \tag{1.22}$$

1.2.4 Determination of h_0 value

When the relaxed model in (1.1) together with the useful additional constraints is solved, we obtain the optimal solution (Z_0). From the two sub-problems A and B we obtain (Z_A) and (Z_B). From the sub-problem A, h_0 is obtained as given in (1.23).

$$h_0 = Z_0 - Z_A. \tag{1.23}$$

Similarly, from the Sub-Problem B, h_0 is obtained as given in (1.24).

$$h_0 = Z_0 - Z_B. \tag{1.24}$$

1.3 Segment-search approach

1.3.1 General integer model

The segment-search approach algorithm for the general integer model is summarized below.

Step 1: Solve the relaxed model (together with the useful additional constraints and slack/excess variable limits) and find an optimal solution.

Step 2: Is the continuous optimal solution integer?
- If yes, then go to Step 3.
- If no, then go to Step 4.

Step 3: Stop an optimal integer solution is available.

Step 4: From each of the two sub-problems determine h_0 and then go to Step 5.

Step 5: Search segments $1, 2, 3, \ldots, k$ until an integer optimal solution is obtained.

Segment 1: $\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0$.

Segment 2: $$\begin{aligned} &\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \ge h_0 \\ &\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0 + h_1. \end{aligned}$$

Segment 3: $$\begin{aligned} &\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \ge h_0 + h_1 \\ &\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0 + h_1 + h_2. \end{aligned}$$

$$\ldots$$

Segment k: $$\begin{aligned} &\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \ge h_0 + h_1 + h_2 + \ldots + h_k \\ &\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0 + h_1 + h_2 + \ldots + h_k + h_{k+1}. \end{aligned}$$

The algorithm is also presented in Figure 1.3.

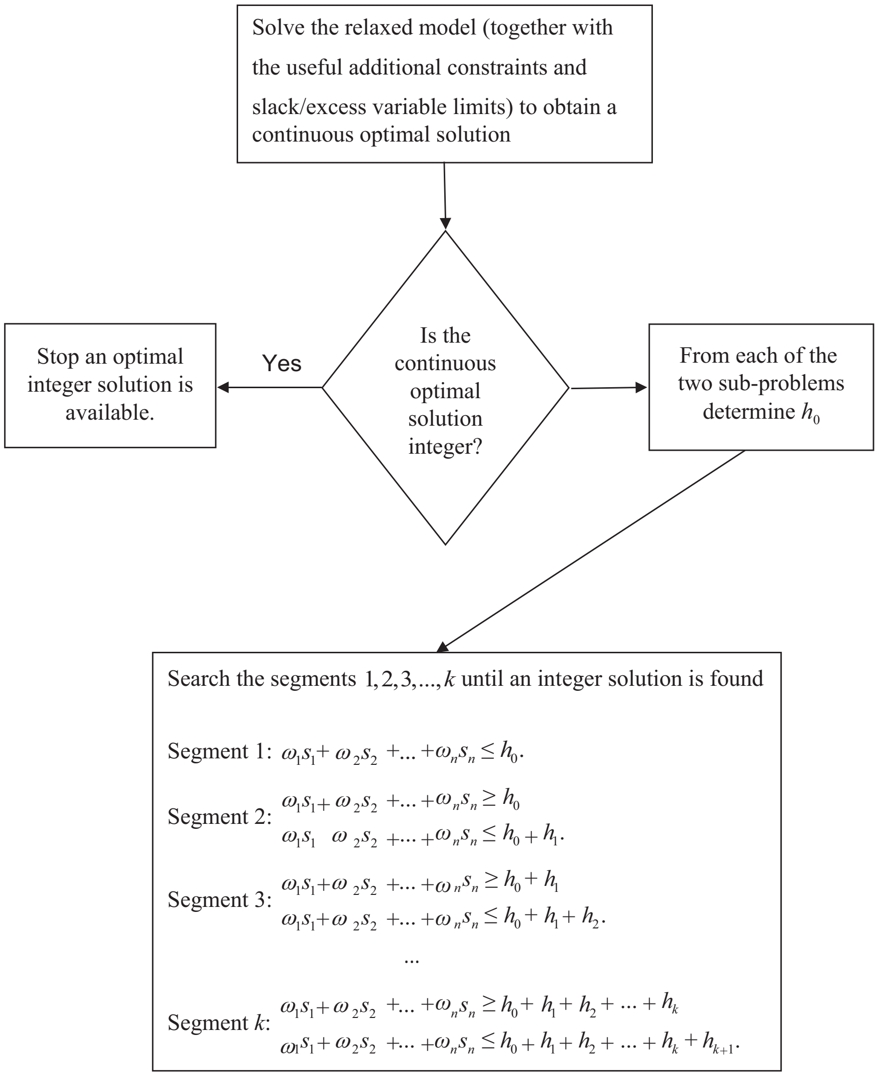

Solve the relaxed model (together with the useful additional constraints and slack/excess variable limits) to obtain a continuous optimal solution
Is the continuous optimal solution integer?
Yes
Stop an optimal integer solution is available.
From each of the two sub-problems determine h_0
Search the segments $1, 2, 3, ..., k$ until an integer solution is found
Segment 1: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \leq h_0$.
Segment 2: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \geq h_0$
$\omega_1 s_1 \quad \omega_2 s_2 + ... + \omega_n s_n \leq h_0 + h_1$.
Segment 3: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \geq h_0 + h_1$
$\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \leq h_0 + h_1 + h_2$.
...
Segment k: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \geq h_0 + h_1 + h_2 + ... + h_k$
$\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \leq h_0 + h_1 + h_2 + ... + h_k + h_{k+1}$.

Figure 1.3: Segment searching algorithm for the general linear integer model.

1.4 Mixed integer model

With the mixed integer model some of the variables are non-integer. As a result, we cannot use the same technique that we used for the general case. Only the integer restricted variables $(x_1, x_2, \ldots, x_r)$ are used to generate the variable sum constraint as is given in (1.25).

$$x_1 + x_2 + \ldots + x_r = \phi + f. \tag{1.25}$$

The two sub-problems A and B.

Sub-problem A:

$$x_1 + x_2 + \ldots + x_r \le \phi, \tag{1.26}$$

Sub-problem B:

$$x_1 + x_2 + \ldots + x_r \ge \phi + 1. \tag{1.27}$$

The segments are searched in such a way that all the non-integer variables are ignored during the search. The segment-search approach for the mixed integer model is summarized below. The decrease in objective value (h_0) is not necessarily integer.

Step 1: Solve the relaxed model (together with the useful additional constraints) to get an optimal solution.

Step 2: Is the continuous optimal solution satisfies integer requirement on integer restricted variables?
- If yes, then go to Step 3.
- If no, then go to Step 4.

Step 3: Stop as optimal mixed integer solution has been obtained.

Step 4: From each of the two sub-problems determine h_0 and then go to Step 5.

Step 5: Search segments $1, 2, 3, \ldots, k$ until the desired mixed integer solution is found,

Segment 1: $\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0.$

Segment 2: $\begin{array}{l}\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \ge h_0 \\ \omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0 + h_1.\end{array}$

Segment 3: $\begin{array}{l}\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \ge h_0 + h_1 \\ \omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0 + h_1 + h_2.\end{array}$

Segment k: $\begin{array}{l}\omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \ge h_0 + h_1 + h_2 + \ldots + h_k \\ \omega_1 s_1 + \omega_2 s_2 + \ldots + \omega_n s_n \le h_0 + h_1 + h_2 + \ldots + h_k + h_{k+1}.\end{array}$

The algorithm is presented in Figure 1.4.

Optimality

As given in Figure 1.1, the segments are in descending order. Since $Z_0 > Z_1 > Z_2 > Z_3 > \ldots > Z_{k-1} > Z_k > Z_{k+1}$, it implies that Z_k is feasible and optimal provided Z_0, Z_1, Z_2, Z_3, . . ., Z_{k-1} are infeasible.

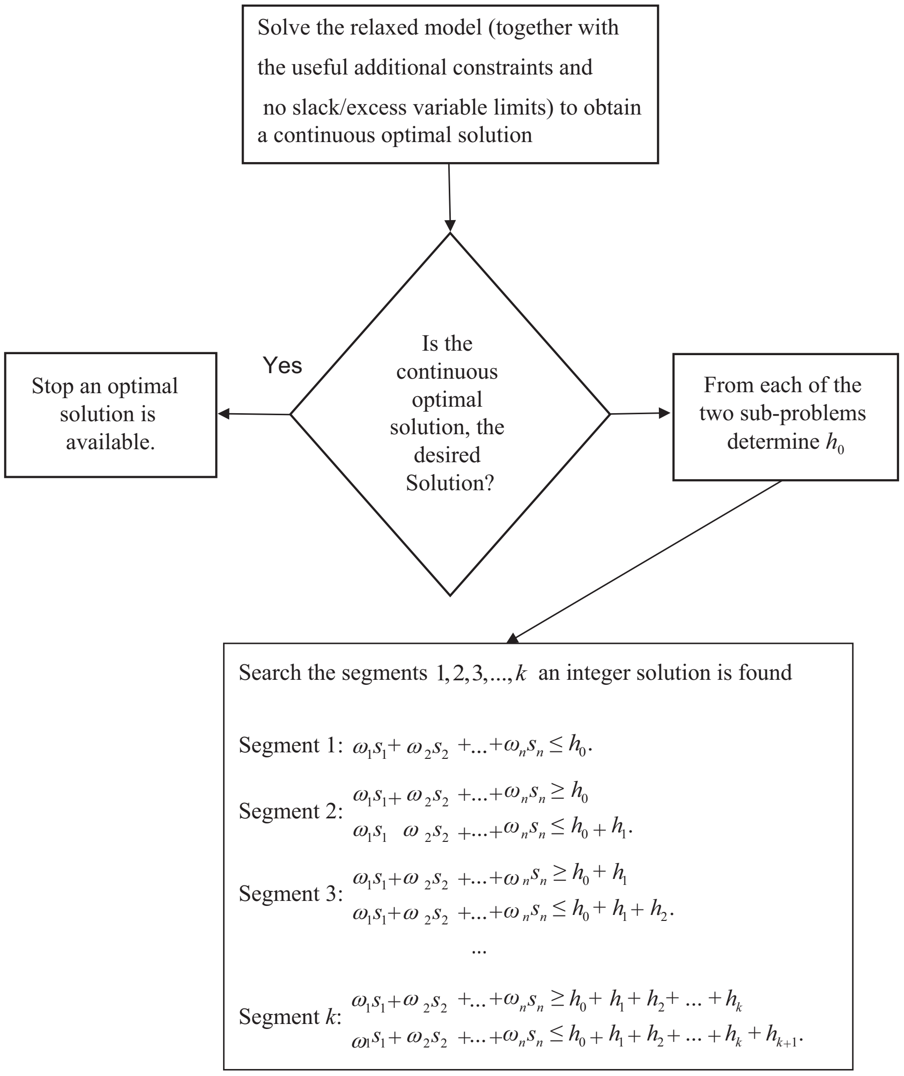

Solve the relaxed model (together with the useful additional constraints and no slack/excess variable limits) to obtain a continuous optimal solution
Is the continuous optimal solution, the desired Solution?
Yes
Stop an optimal solution is available.
From each of the two sub-problems determine h_0
Search the segments $1,2,3,...,k$ an integer solution is found
Segment 1: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \le h_0$.
Segment 2: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \ge h_0$
$\omega_1 s_1 \quad \omega_2 s_2 + ... + \omega_n s_n \le h_0 + h_1$.
Segment 3: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \ge h_0 + h_1$
$\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \le h_0 + h_1 + h_2$.
...
Segment k: $\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \ge h_0 + h_1 + h_2 + ... + h_k$
$\omega_1 s_1 + \omega_2 s_2 + ... + \omega_n s_n \le h_0 + h_1 + h_2 + ... + h_k + h_{k+1}$.

Figure 1.4: Segment searching algorithm for the mixed general linear integer model.

1.5 Numerical illustration

1.5.1 General linear integer model: Example 1.1

Maximize

$$Z = 23x_1 + 25x_2 + 24x_3$$

Such that:

$$\begin{aligned}6x_1 + 23x_2 + 24x_3 &\le 113 \\ 29x_1 + 15x_2 + 13x_3 &\le 211 \\ 18x_1 + 7x_2 + 16x_3 &\le 132\end{aligned} \tag{1.28}$$

Where x_1, x_2 and x_3 are non-negative integers.

Solving the relaxed problem, we obtain the continuous optimal solution as given in Table 1.2.

Table 1.2: Tableau giving continuous optimal solution to numerical illustration (1.28).

	x_1	x_2	x_3	s_1	s_2	s_3	r.h.s.
Z	0	0	0.3449	0.6586	0.66568	0.0000	213.0139
x_2	0	1	1.0.711	0.0503	−0.0104	0.0000	3.4853
x_1	1	0	0.1057	0.0260	0.0399	0.0000	5.4731
s_3	0	0	0.4055	−0.1161	0.6447	1.0000	9.0867
$Z_0 = 213$							

Direct branch and bound

Solving directly by branch and bound method, the worst case takes <u>**25**</u> sub-problems to verify the optimal solution given in (1.29).

$$Z_{opt} = 190, x_1 = 5, x_2 = 3, x_3 = 0 \tag{1.29}$$

1.5.2 Solution by the proposed segment-search algorithm

Useful additional constraints

From the coefficient matrix:

$$\begin{array}{cccccc} & 6x_1 + & 23x_2 + & 24x_3 \le 113 & & \\ & 29x_1 + & 15x_2 + & 13x_3 \le 211 & & \\ & 18x_1 + & 7x_2 + & 16x_3 \le 132 & & \\ & \uparrow & \uparrow & \uparrow & & \\ \text{Max} & 29 & 23 & 24 & \rightarrow & \text{Min} = 23 \end{array}$$

i.e. $c_1^L = 29, c_2^L = 23,\ c_3^L = 24, \ell = 23$ and $\ell - 1 = 23 - 1 = 22$.

Let our three slack variables be s_1, s_2, s_3. $\therefore s_1 \le 22, s_2 \le 22, s_3 \le 22$.

Original variable sum constraint: $x_1 + x_2 + x_3 - \phi_c = 0$.
Slack variable sum constraint: $s_1 + s_2 + s_3 - \lambda = 0$.

With these useful additional constraints, the given problem becomes as given in (1.30).
Maximize

$$Z = 23x_1 + 25x_2 + 24x_3$$

Such that:

$$\begin{aligned} &6x_1 + 23x_2 + 24x_3 + s_1 = 113 \\ &29x_1 + 15x_2 + 13x_3 + s_2 = 211 \\ &18x_1 + 7x_2 + 16x_3 + s_3 = 132 \\ &x_1 + x_2 + x_3 - \phi_c = 0 \\ &s_1 + s_2 + s_3 - \lambda = 0 \\ &s_1 \le 22, s_2 \le 22, s_3 \le 22 \end{aligned} \tag{1.30}$$

Where $s_1, s_2, s_3, \phi_c, \lambda$ are non-negative integers.

This solution process for the general case is presented in Figure 1.5.

1.5.3 Mixed integer model: Example 1.2

Maximize

$$Z = 23x_1 + 25x_2 + 24x_3$$

Such that:

$$\begin{aligned} 6x_1 + 23x_2 + 24x_3 &\le 113 \\ 29x_1 + 15x_2 + 13x_3 &\le 211 \\ 18x_1 + 7x_2 + 16x_3 &\le 132 \end{aligned} \tag{1.31}$$

Where x_2 and x_3 are non-negative integers.

Direct branch and bound
Solving directly by branch and bound method, the worst case takes $\underline{7}$ sub-problems to verify the optimal solution given by:

$$Z_{opt} = 206.66,\ \ x_1 = 5.72,\ \ x_2 = 3,\ \ x_3 = 0$$

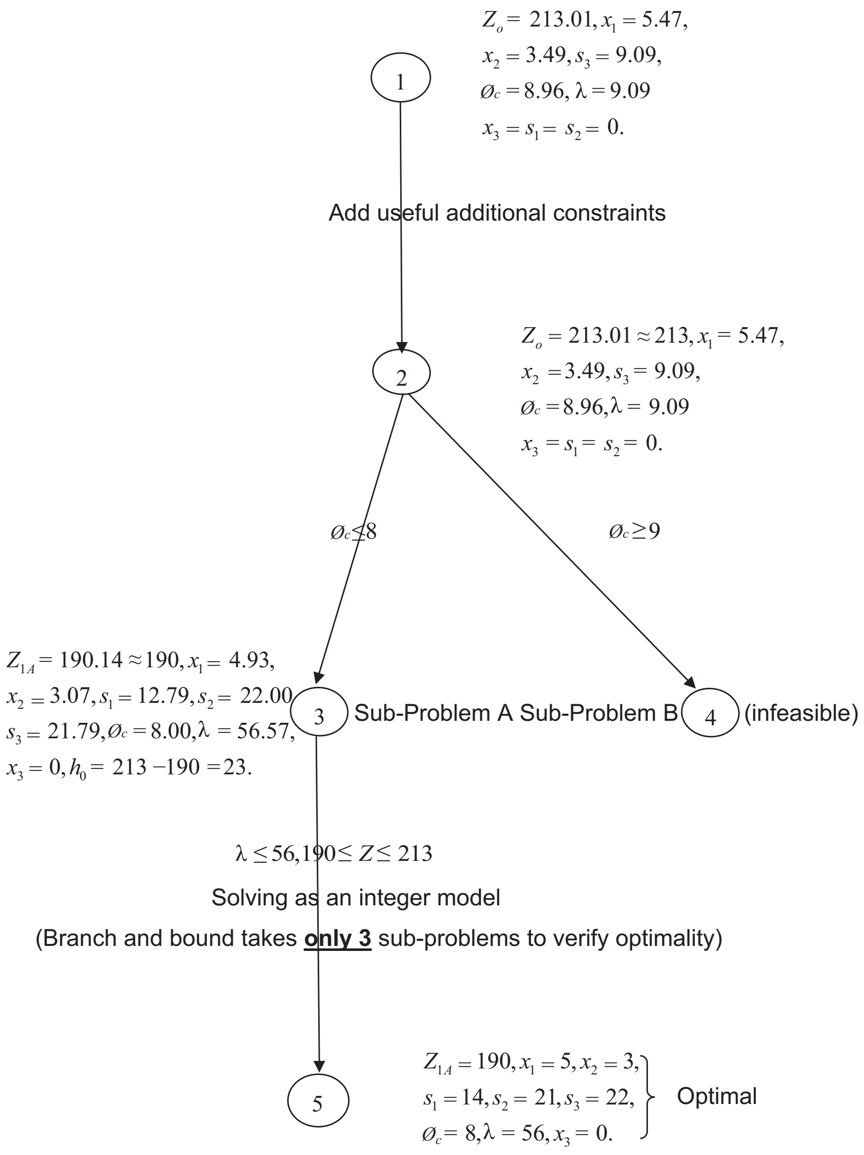

1
$Z_o = 213.01, x_1 = 5.47,$
$x_2 = 3.49, s_3 = 9.09,$
$Ø_c = 8.96, \lambda = 9.09$
$x_3 = s_1 = s_2 = 0.$
Add useful additional constraints
2
$Z_o = 213.01 \approx 213, x_1 = 5.47,$
$x_2 = 3.49, s_3 = 9.09,$
$Ø_c = 8.96, \lambda = 9.09$
$x_3 = s_1 = s_2 = 0.$
$Ø_c \leq 8$
$Ø_c \geq 9$
$Z_{1A} = 190.14 \approx 190, x_1 = 4.93,$
$x_2 = 3.07, s_1 = 12.79, s_2 = 22.00$
$s_3 = 21.79, Ø_c = 8.00, \lambda = 56.57,$
$x_3 = 0, h_0 = 213 - 190 = 23.$
3
Sub-Problem A Sub-Problem B
4
(infeasible)
$\lambda \leq 56, 190 \leq Z \leq 213$
Solving as an integer model
(Branch and bound takes **only 3** sub-problems to verify optimality)
5
$Z_{1A} = 190, x_1 = 5, x_2 = 3,$
$s_1 = 14, s_2 = 21, s_3 = 22,$
$Ø_c = 8, \lambda = 56, x_3 = 0.$
Optimal

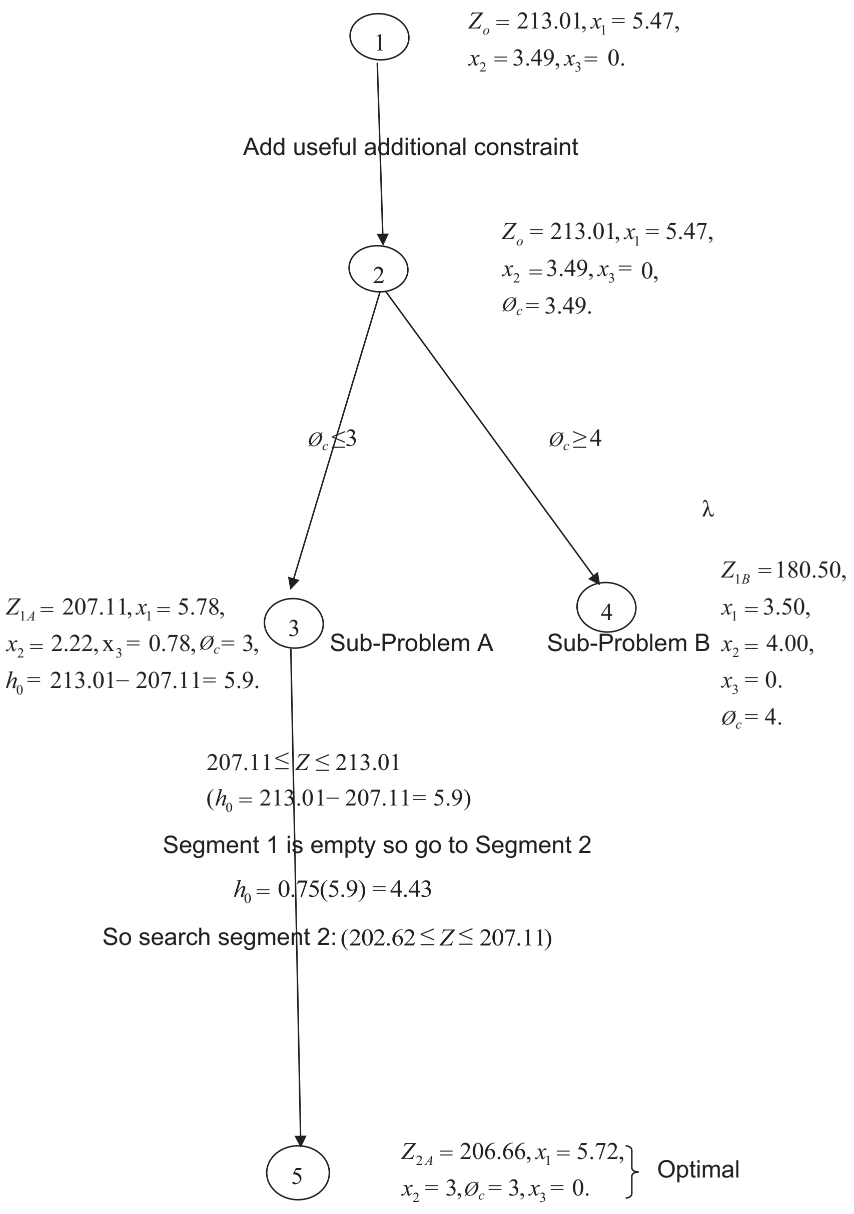
Figure 1.5: Flow chart - proposed solution process.

Solution by segment search

Useful additional constraint

Original variable sum constraint: $x_2 + x_3 - \phi_c = 0.$

With the useful additional constraint, the given problem becomes as in (1.32).



1
$Z_o = 213.01, x_1 = 5.47,$
$x_2 = 3.49, x_3 = 0.$
Add useful additional constraint
2
$Z_o = 213.01, x_1 = 5.47,$
$x_2 = 3.49, x_3 = 0,$
$Ø_c = 3.49.$
$Ø_c \leq 3$
$Ø_c \geq 4$
λ
3
$Z_{1A} = 207.11, x_1 = 5.78,$
$x_2 = 2.22, x_3 = 0.78, Ø_c = 3,$
$h_0 = 213.01 - 207.11 = 5.9.$
Sub-Problem A
4
Sub-Problem B
$Z_{1B} = 180.50,$
$x_1 = 3.50,$
$x_2 = 4.00,$
$x_3 = 0.$
$Ø_c = 4.$
$207.11 \leq Z \leq 213.01$
$(h_0 = 213.01 - 207.11 = 5.9)$
Segment 1 is empty so go to Segment 2
$h_0 = 0.75(5.9) = 4.43$
So search segment 2: $(202.62 \leq Z \leq 207.11)$
5
$Z_{2A} = 206.66, x_1 = 5.72,$
$x_2 = 3, Ø_c = 3, x_3 = 0.$
Optimal

Figure 1.6: Flow chart - proposed solution process for the mixed integer model.

Maximize

$$Z = 23x_1 + 25x_2 + 24x_3$$

Such that:

$$\begin{gathered} 6x_1 + 23x_2 + 24x_3 \le 113 \\ 29x_1 + 15x_2 + 13x_3 \le 211 \\ 18x_1 + 7x_2 + 16x_3 \le 132 \\ x_2 + x_3 - \phi_c = 0 \end{gathered} \tag{1.32}$$

Where ϕ_c is a non-negative integer.

This solution process for the mixed integer case is presented in Figure 1.6.

1.6 Conclusions

The segment search approach discussed in this chapter is very effective. Huge numbers of sub-problems that are required to verify optimality in a branch and bound framework are significantly reduced if the feasible region is searched segment by segment. The method accommodates other methods within its context. For example, more efficient approaches such as the branch and cut, branch and prize or branch cut and price can be used to search the segments once the additional constraints have been added. The segments are in descending order and the first feasible solution is optimal to the given problem.

It is our hope that the reformulation which was used to formulate useful additional constraints in this chapter will play an important role in search for an efficient algorithm for the integer model. More on reformulation can be found in Munapo and Kumar (2016).

References

Al-Rabeeah, M., Kumar, S., Al-Hasani A., Munapo, E. & Eberhard, A. 2019. Computational Enhancement in the Application of the Branch and Bound Method for Linear Integer Programs and Related Models, International Journal of Mathematical, Engineering and Management Sciences Vol. 4, No. 5, 1140–1153, 2019 https://dx.doi.org/10.33889/IJMEMS.2019.4.5-090

Taha, H.A. 2017. Operations Research: An Introduction, Pearson Educators, 10th Edition.

Winston, W.L. 2004. Operations Research Applications and Algorithms, Duxbury Press, 4th Edition.

Land, A.H., A.G. Doig. 1960. An Automatic method for solving discrete programming problems, *Econometrica* **28**, 497–520.

Dakin, R.J., 1965. A tree search algorithm for mixed integer programming problems. *The Computer Journal* **8**, 250–255.

Brunetta, L., M. Conforti, G. Rinaldi. 1997. A branch and cut algorithm for the equicut problem. *Mathematical Programming* **78**, 243–263.

Mitchell, J.E.,2001. Branch and cut algorithms for integer programming: In Floudas, C.A. and Pardalos, P.M. (Eds.), Encyclopedia of Optimization, Kluwer Academic Publishers.

Padberg, M., G. Rinaldi. 1991, A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review* **33**(1), 60–100.

Barnhart, C., E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance. 1998. Branch and price column generation for solving huge integer programs. *Operations Research* **46**, 316–329.

Salvelsbergh, M.W.P. 1997. A branch and price algorithm to solve the generalized assignment problem. *Operations Research* **45**, 381–841.

Barnhart, C., C.A. Hane, P.H. Vance. 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research* **48**, 318–326.

Fukasawa, R., H. Longo, J. Lysgaard, M. Poggi de Aragao, E. Uchoa, R.F. Werneck. 2006. Robust branch-and-cut-price for the Capacitated vehicle routing problem. *Mathematical Programming* Series A **106**, 491–511.

Ladányi, L., T.K. Ralphs, L.E. Trotter (Jr.). 2001. Branch, cut and Price: Sequential and Parallel, in Computational Combinatorial Optimization, Naddef, N. and Jüenger, M., eds, Springer, Berlin, 223.

Munapo, E. 2020. Improvement of the branch and bound algorithm for solving the knapsack linear integer problem, Eastern-European Journal of Enterprise Technologies, pp. 59–69.

Munapo, E. and Kumar, S. 2016. Knapsack constraint reformulation: A new approach that significantly reduces the number of sub-problems in the branch and bound algorithm, Cogent Mathematics, Vol. 3, 1162372, 2016.

Kumar, S., Munapo, E., and Jones, B.C. (2007) An integer equation controlled descending path to pure integer program, Indian Journal of Mathematics, Vol 49, No 2, pp 211–237.

Kumar, S., Luhandjula, M. K., Munapo, E., and Jones, B.C. (2010). Fifty years of Integer Programming: A review of solution approaches, Asia Pacific Business review, Vol 6, No 2, pp 5–15.

Kumar, S., and Munapo, E., (2012), Some lateral ideas and their applications for developing new solution procedures for a pure integer programming model, Proc of Herbal International conference of application of Mathematics and Statistics – Intelligent Solutions through Mathematics and Statistics, (Editor) Mariappan, Srinivasan and Amritraj, Excel India Publishers, pp 13–21.

Chapter 2
Improved solution method for the 0-1 GAP model

Abstract: This chapter presents an improved version of the transportation branch and bound algorithm for solving the generalized assignment problem (GAP). This improved version is based on the branch and bound technique for the GAP model that was proposed by Munapo et al. (2015) in which the sub-problems are solved by the available efficient transportation techniques rather than the usual simplex-based approaches. The improvement in this approach is that branches are not necessary. The transportation GAP model is balanced and formulated as a linear program (LP) and solved. An interesting feature of these relaxed constraints that result from the GAP model is that they are made up of zero and ones only as coefficients. The GAP model with relaxed constraints is solved and at every iteration the current solution is tested for feasibility using the original constraints. The violated original constraints are used to generate cuts that are added to the current problem and solved to get a new solution. If solution is feasible then it is optimal else process is repeated until a feasible and optimal solution is found.

Keywords: Generalized assignment problem (GAP), Transportation model, Linear programming (LP), Violated constraints and cuts

2.0 Introduction

The generalized assignment problem is a very difficult zero-one linear programming problem (LP) Fisher, et al. (1986), Guignard and Rosenwein, (1989), Martello and Toth, (1981), Pigatti, et al. (2005). Some of the applications of the generalized assignment problem include location, machine scheduling, supply chain, routing of vehicle and allocation of resource, Savelsburgh, (1997), Toth and Vigo (2001), Yagiura, et al. (2004, 2006). Munapo et al. (2015) proposed a transportation branch and bound algorithm for solving the generalized assignment problem. This is a branch and bound technique in which the sub-problems are solved by use of the available efficient transportation techniques rather than the usual simplex-based approaches. A technique for selecting branching variables is also available. Yagiura and Glover (2004) also discussed the GAP model.

The obvious weakness of the approach proposed by Munapo et al. (2015) is that the larger the generalized model, the larger will be the number branches required to verify optimality.

To alleviate this challenge of large numbers of sub-problems required to verify optimality, this chapter presents an improved version of this algorithm where there are no more branches but only a single transportation linear programming (LP)

https://doi.org/10.1515/9783110703023-002

problem, which increases only in size (i.e. increase in the number of rows) at every iteration.

The chapter has been organized in 7 sections. Sections 2.1 to 2.4 present the generalized assignment problem and the relaxation process. Sections 2.5 and 2.6 present the solution process of the generalized problem and finally 2.7 presents the conclusion.

2.1 Generalized assignment problem

In this chapter the generalized assignment problem (2.1) is considered.

Minimize

$$\begin{bmatrix} c_{11} & c_{12} & \ldots & c_{mn} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ \ldots \\ x_{mn} \end{bmatrix},$$

subject to

$$\begin{bmatrix} a_{i1} & a_{i2} & \ldots & a_{in} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \ldots \\ x_{in} \end{bmatrix} \le [b_i], \forall i = 1, 2, \ldots, m. \tag{2.1}$$

$$\begin{bmatrix} x_{1j} & x_{2j} & \ldots & 2_{mj} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \ldots \\ 1 \end{bmatrix} = [1], \forall j = 1, 2, \ldots, n.$$

where $x_{ij} = 0$ or 1,

$i = 1, 2, \ldots m,$ is a set of agents.

$j = 1, 2, \ldots n,$ is a set of tasks.

c_{ij} is the cost of assigning agent i to task j.

r_{ij} is the resource needed by agent i to do task j.

b_i is the resource available to agent i.

2.2 Relaxation process

The generalized assignment problem is relaxed by changing it into a transportation problem. The main reason is because transportation models are easier to solve than original linear programming version. Efficient techniques to solve the transportation models such as the MODI or network simplex method can be applied.

2.3 GAP model in relaxed form

The GAP constraints which are given in (2.2), can be replaced by clique inequalities given in (2.3).

$$\begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{in} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \dots \\ x_{in} \end{bmatrix} \leq [b_i], \forall i = 1, 2, \dots, m. \tag{2.2}$$

$$\begin{bmatrix} x_{i1} & x_{i2} & \dots & x_{in} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} \leq [\lambda_i], \forall i = 1, 2, \dots, m. \tag{2.3}$$

Where λ_i is integer and is obtained by solving the knapsack problem in (2.4).

$$\lambda_i = \text{Maximize}\begin{bmatrix} \mathrm{x}_{\mathrm{i}1} & \mathrm{x}_{\mathrm{i}2} & \dots & \mathrm{x}_{\mathrm{in}} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix},$$

Subject to

$$\begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{in} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \dots \\ x_{in} \end{bmatrix} \leq [b_i], \tag{2.4}$$

$$0 \leq x_{ij} \leq 1.$$

The optimal solution to this knapsack problem is obtained using simplex method.

2.4 The relaxed transportation model

The relaxed transportation problem is given in (2.5).

Minimize

$$\begin{bmatrix} c_{11} & c_{12} & \dots & c_{mn} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ \dots \\ x_{mn} \end{bmatrix},$$

Subject to

$$\begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{in} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \dots \\ x_{in} \end{bmatrix} \le [b_i], \ \forall i = 1, 2, \dots, m, \tag{2.5}$$

$$\begin{bmatrix} x_{1j} & x_{2j} & \dots & 2_{mj} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} = [1], \ \forall j = 1, 2, \dots, n.$$

The GAP in transportation relaxed form is given in Table 2.1.

Table 2.1: The relaxed transportation form.

					Supply
	c_{11}	c_{12}	. . .	c_{n1}	λ_1
	c_{21}	c_{22}	. . .	c_{n2}	λ_2
	…	. . .	. . .	. . .	. . .
	c_{m1}	c_{m2}	. . .	c_{m2}	λ_m
Demand	**1**	**1**	. . .	**1**	

The initial solution is obtained by solving the transportation problem given in Table 2.1.

The solution is tested for feasibility, if feasible then it is optimal else improve solution until it is optimal.

2.5 GAP transportation branch and bound algorithm

The transportation branch and bound algorithm for GAP is summarized by the following steps.

Step 1: Obtain a lower bound by relaxing the GAP model.
Step 2: From the most restricted row select the branching variables.
Step 3: Branch using the selected variables. Return to step 2 until a candidate solution is feasible.

Candidate solution: A solution is said to be the candidate solution if it is the smallest optimal solution available.

2.5.1 Optimality

From the k terminal nodes $Z_1^\tau, Z_2^\tau, \ldots, Z_k^\tau$ the optimal solution Z_o to the GAP problem is obtained as given in (2.6).

$$Z_o = \min\left[Z_1^\tau, Z_2^\tau, \ldots, Z_k^\tau\right]. \tag{2.6}$$

2.5.2 Numerical illustration

Use the transportation branch and bound algorithm to (2.7).

$$\operatorname{Min} Z_0 = \begin{bmatrix} 127 & 175 & 151 & 127 & 197 & 142 & 191 & 199 & 192 & 133 & 124 & 135 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \end{bmatrix}, \tag{2.7a}$$

Subject to:

$$\begin{bmatrix} 49 & 75 & 46 & 74 \\ 25 & 48 & 61 & 71 \\ 43 & 58 & 86 & 47 \end{bmatrix} \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ x_{14} & x_{24} & x_{34} \end{bmatrix} \leq \begin{bmatrix} 125 \\ 100 \\ 118 \end{bmatrix} \tag{2.7b}$$

$$\begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ x_{14} & x_{24} & x_{34} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{2.7c}$$

Where

$$\begin{gathered} x_{ij} = 0 \text{ or } 1 \\ i = 1, 2, \dots m, \\ j = 1, 2, \dots n, \end{gathered} \tag{2.7d}$$

Solving directly using the branch and bound algorithms we require 15 sub-problems to verify optimality and have (2.8) as the optimal solution.

$$Z_o = 520, x_{11} = x_{14} = x_{22} = x_{33} = 1. \tag{2.8}$$

From (2.9) we have (2.10).

$$\lambda_i = \text{Maximize} \begin{bmatrix} \mathrm{x}_{i1} & \mathrm{x}_{i2} & \mathrm{x}_{i3} & \mathrm{x}_{i4} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \ \mathrm{i} = 1, 2, 3.$$

Subject to

$$\begin{bmatrix} a_{i1} & a_{i2} & a_{i3} & a_{i4} \end{bmatrix} \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{in} \end{bmatrix} \leq [b_i], \tag{2.9}$$

$$\begin{gathered} 0 \leq x_{ij} \leq 1, \\ \forall i = 1, 2, 3. \end{gathered}$$

Solving the three LPs we have (2.10).

$$\lambda_1 = 2, \lambda_2 = 2 \,\&\, \lambda_3 = 2. \tag{2.10}$$

The relaxed transportation becomes as shown in Table 2.2.

Table 2.2: The relaxed transportation form for the illustration.

					Supply
	127	175	151	127	2
	197	142	191	199	2
	192	133	124	135	2
Demand	1	1	1	1	

Balancing the transportation problem, we have Table 2.3.

Table 2.3: Balanced transportation problem.

						Supply
	127	175	151	127	0	2
	197	142	191	199	0	2
	192	133	124	135	0	2
Demand	1	1	1	1	2	

Node 1

Solving as a transportation problem the optimal solution is given in Table 2.4.

Table 2.4: Optimal solution to balanced transportation problem.

						Supply
	127(1)	175	151	127(1)	0	2
	197	142	191	199	0(2)	2
	192	133(1)	124(1)	135	0	2
Demand	1	1	1	1	2	

Optimal solution is given in (2.11).

$$Z_o = 511, x_{11} = x_{14} = x_{32} = x_{33} = 1. \tag{2.11}$$

This solution is Node 0 and is infeasible. The basic variables x_{32} and x_{33} cannot be both basic. In other words the third constraint is violated, i.e.

$$43(0) + 58(1) + 86(1) + 47(0) \leq 118,$$
$$144 \leq 118. \tag{2.12}$$

This is not possible so either $x_{32} = 0$ or $x_{33} = 0$. These are the two initial branches of transportation branch and bound.

First Branch $x_{33} = 0$

When branching in this direction the transportation problem becomes Table 2.5.

Table 2.5: When $x_{33} = 0$.

						Supply
	127	175	151	127	0	2
	197	142	191	199	0	2
	192	133	10000	135	0	2
Demand	1	1	1	1	2	

Solving this we obtain the optimal solution which becomes our Node 2 as in Table 2.6.

Table 2.6: Node 2.

						Supply
	127(1)	175	151(1)	127	0	2
	197	142	191	199	0(2)	2
	192	133(1)	10000	135(1)	0	2
Demand	1	1	1	1	2	

The optimal solution is given in (2.13).

$$Z_1^T = 546, x_{11} = x_{13} = x_{32} = x_{35} = 1. \tag{2.13}$$

This solution is feasible and is a candidate solution.

Second Branch $x_{32} = 0$

When branching in this direction the transportation problem becomes Table 2.7.

Table 2.7: When $x_{32} = 0$.

						Supply
	127	175	151	127	0	2
	197	142	191	199	0	2
	192	10000	124	135	0	2
Demand	1	1	1	1	2	

Solving this we obtain the optimal solution which becomes the Node 3 as in Table 2.8.

Table 2.8: Node 3.

						Supply
	127(1)	175	151	127(1)	0	2
	197	142(1)	191	199	0(1)	2
	192	10000	124(1)	135	0(1)	2
Demand	1	1	1	1	2	

The optimal solution is given in (14).

$$Z_2^{\tau} = 520, x_{11} = x_{14} = x_{22} = x_{33} = 1. \tag{2.14}$$

This solution is feasible and is also a candidate solution.

The transportation branch and bound algorithm can be represented as a tree as shown in Figure 2.1.

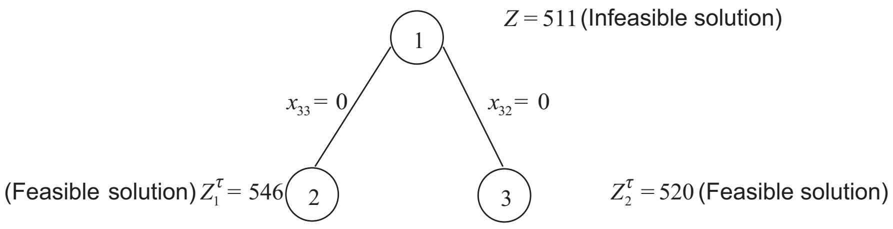

$Z = 511$ (Infeasible solution)
1
$x_{33} = 0$
$x_{32} = 0$
(Feasible solution) $Z_1^{\tau} = 546$
2
3
$Z_2^{\tau} = 520$ (Feasible solution)

Figure 2.1: Transportation branch and bound.

The optimal solution for the GAP illustration is given in (2.15).

$$Z_o = \min[546, 520] = 520. \tag{2.15}$$

2.6 Improved solution method for GAP

In this approach the relaxed GAP model is still changed into a transportation problem, balanced, and not solved directly. Instead, we use the relaxed GAP transportation problem to generate new constraints. These relaxed constraints are special and easier to solve than the original ones. An interesting feature of these relaxed constraints is that they are made up of zero and ones only as coefficients. The GAP model with relaxed constraints is solved and at every iteration the current solution is tested for feasibility using the original constraint. The violated original constraints are used to generate cuts that are added to the current problem and solved to get a new solution. If solution is feasible then it is optimal else process is repeated until a feasible and optimal solution is found.

2.6.1 Proposed algorithm

The improved GAP solution method is summarized as follows.

Step 1: Relax GAP to obtain a transportation model.

Step 2: Balance the transportation model and formulate a relaxed LP.

Step 3: Solve the relaxed LP model to obtain an optimal integer solution. If solution is feasible then it is optimal else go to Step 4.

Step 4: Use the violated original constraints to generate cuts and add these cuts to the current LP problem and return to Step 3. Generated cuts must have 0,-1 and 1 as coefficients only.

2.6.2 Strength of proposed algorithm

The problem remains one. The number of sub-problems are kept at minimal.

2.6.3 Reconsider the same numerical example

Solving the same numerical example using the improved gap solution we start with the balanced transportation given Table 2.3. From Table 2.3 the relaxed linear programming model is given in (2.16).

$$\text{Min} Z_0 = \begin{bmatrix} 127 & 175 & 151 & 127 & 197 & 142 & 191 & 199 & 192 & 133 & 124 & 135 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{24} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \end{bmatrix} \tag{2.16a}$$

Subject to:

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{21} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \le \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}, \tag{2.16b}$$

$$\begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ x_{14} & x_{24} & x_{34} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{2.16c}$$

Where

$$\left.\begin{aligned} x_{ij} &= 0 \text{ or } 1 \\ i &= 1, 2, 3 \text{ is a set of agents,} \\ j &= 1, 2, 3, 4 \text{ is a set of tasks.} \end{aligned}\right\} \tag{2.16d}$$

The flow diagram for the given numerical illustration using the improved solution method for GAP is given in Figure 2.2.

Iteration 1

Solving (2.16) as a linear integer problem we have Node 0.

Node 0: $Z_o = 511, x_{11} = x_{14} = x_{32} = x_{33} = 1$.

Violation: $\begin{bmatrix} 43 & 58 & 86 & 47 \end{bmatrix} \begin{bmatrix} x_{31} \\ x_{32} \\ x_{33} \\ x_{34} \end{bmatrix} \le [118]$.

Generated cut: $\begin{bmatrix} x_{31} & x_{32} & x_{33} & x_{34} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \le 1$. Note that the cut has coefficients 0s and 1s only.

Iteration 2

Solving (2.16) + cut as a linear integer problem we have Node 1.

Node 1: $Z_1 = 520, x_{11} = x_{14} = x_{22} = x_{33} = 1$.

Check violations: No violations and therefore this solution is feasible and optimal.

The search tree for the proposed algorithm can be represented in Figure 2.2.

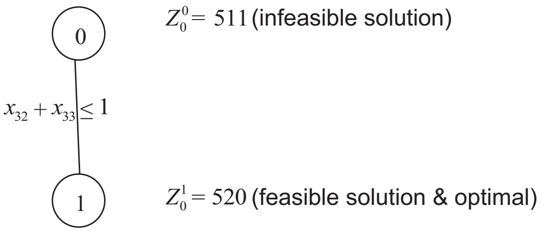

0
$Z_0^0 = 511$ (infeasible solution)
$x_{32} + x_{33} \le 1$
1
$Z_0^1 = 520$ (feasible solution & optimal)

Figure 2.2: Search tree for the proposed algorithm.

Note that the problem remains one and that the number of rows increase as cuts are added at every iteration.

2.7 Conclusions

The proposed approach has the advantage that the problem remains a single problem. The numbers of sub-problems created are kept at minimal thus reducing the chances of the GAP model exploding into large numbers of sub-problems as is the case with branch and bound related approaches. The relaxed coefficients of the relaxed formulated LP are made of zeros and ones only. This makes the sub-problems easier to solve than the original GAP constraints.

References

Fisher, M.L., Jaikumar, R., Van Wassenhove, L.N., (1986). A Multiplier Adjustment Method for the Generalized Assignment Problem. Mangement Science 32(9),1095–1103.

Guignard, M. Rosenwein, M., (1989). An Improved Dual-Based Algorithm for the Generalized Assignment Problem, Operations Research 37(4),658–663.

Martello, S., Toth, P., (1981). An algorithm for the generalized Assignment Problem. In Operations Research 81. J.P. Brans (ed.), North-Holland, Amsterdam, 589–603.

Munapo, E., Lesaoana, M., Philimon, N., & Kumar, S. (2015). A transportation branch and bound algorithm for solving the generalized assignment problem. International Journal of System Assurance Engineering and Management. 6. 217–223. 10.1007/s13198-015-0343-9.

Pigatti, A., Poggie de Aragao, M., Uchoa, E., (2005). Stabilized branch and cut and price for the generalized assignment problem. In 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics. Electronic Notes in Discrete Mathematics, Elsevier, Amsterdam, Vol. 19, 389–395.

Ross, G.T., Soland, R.M., (1975). A Branch and Bound algorithm for the Generalized Assignment Problem, Mathematical Programming 8, 91–103.

Savelsburgh, M., (1997). A branch and price algorithm for the generalized assignment problem, Operations Research 45(6),831–841.

Toth, P., Vigo, D., (2001). The Vehicle Routing Problem, Society for Industrial and Applied.

Yagiura, M., Ibaraki, T., Glover, F., (2004). An ejection chain approach for the generalized assignment problem, Informs Journal of Computing 16, 133–151.

Yagiura, M., Ibaraki, T., Glover, F., (2006). A path re-linking approach with ejection chains for the generalized assignment problem, European Journal of Operational Research 169, 548–569.

Chapter 3
A search for an optimal integer solution over the integer polyhedron – Two iterative approaches

Abstract: This chapter presents a search procedure for a pure integer programming model, which is a modification of the simplex method for linear programming. This modified search procedure moves on the integer polyhedron to find the optimal integer solution. This integer polyhedron is formed within the convex feasible space generated by the given linear constraints. Young (1965, 1968, 1969) developed, for the first time, a method that searched for the optimal solution over the integer polyhedron, but surprisingly this method has not received as much attention as other methods have. In this chapter, a new search procedure over the integer polyhedron has been described. This new method is by Munapo, Kumar and Khan (2010, 2012) and it is different to Young's approach and it was developed independently. This new method has been compared to the Young's approach and the method by Munapo, Kumar and Khan (2010, 2012) has potential applications, when information recycling in integer programs is required. Some illustrative examples are given.

Keywords: Integer polyhedron, Young's approach, Pure integer program, Gomory constraints, Integer polyhedron iterative approach

3.0 Introduction

An integer program is a linear program with integer restrictions on all variables. It is also known as a pure linear integer program (PLIP). These problems are usually difficult to solve as the required optimal integer point may be within the convex region generated by the LP constraints. An exception arises for totally uni-modular systems, where extreme points of the convex polyhedron are also integer points. Since the required optimal integer solution is not necessarily an extreme point of the convex space, the LP based approaches need appropriate modifications to detect the integer optimal solution. For a general pure integer programming (PIP) model, Young (1965, 1968, 1969) developed a simplex like search procedure. Young's approach moves on an integer polyhedron. He called this approach a primal integer programming method. It may be noted that although the method has been around for a long time, but not much has been written about its performance, application and usage by the practitioners, as the software developers did not pay attention to include Young's approach in the standard LP and ILP solvers. However, a search for a direct method remained in demand as can be seen from Schnijiver (1986). Motivated by Schnijiver (1986) statement, Munapo, Kumar and Khan (2010, 2012) developed a direct method

https://doi.org/10.1515/9783110703023-003

that finds an optimal integer point by moving along on the integer polyhedron. The approach by Munapo, Kumar and Khan (2010), is different to Young's approach and unlike the Young's approach, where recycling concept by Kumar (2005, 2006) is also applicable. Both approaches are discussed in this chapter. This chapter has the following objectives:

(a) Identify properties associated with the integer polyhedron.
(b) Discuss the Young's approach.
(c) Present the alternative search method by Munapo, Kumar and Khan (2010), which also moves around the integer polyhedron. It is hoped that it will revive interest in the direct search over the integer polyhedron. This new approach has been called **"Integer Polyhedron Search Procedure" (IPSP).**
(d) We hope, our approach will encourage researchers to experiment with these two direct search methods and software companies will add these procedures in the software library. We also hope researchers will evaluate performances of these two methods and compare them with other methods for solving a general linear integer program. As far as known to us, experiments with these direct search methods are lacking in the literature.

Other available approaches in integer programming rely on the strategy to generate a continuous LP optimal solution and from the optimal LP solution, these methods have developed strategies to move towards the required integer optimal solution (see Edelsbrunner (1987), Beasely (1996), Hillier and Lieberman (2005), and Karlof (2005). Other studies that may be of interest are: Nemhauser and Wolsey (1988); and Salkin and Mathur (1989). A review of various approaches has been presented by Kumar *et al.* (2010).

This chapter has been organized in six sections. Section 3.1 deals with various definitions and the mathematical background required for the development of this chapter. Young's (1965) approach has been described in Section 3.2. The integer polyhedron search procedure (IPSP) by Munapo, Kumar and Khan (2010) has been presented in Section 3.3. Numerical illustrations have been presented for both approaches are discussed in Section 3.4 and finally the chapter has been concluded in Section 3.5.

3.1 Background information

This section deals with the necessary background information needed for the two methods presented in this chapter.

3.1.1 Geometry of integer-points in a convex space defined by the linear constraints

Consider a m constraints and n variables PIP with integer data, i.e. A, b and C are integer element matrix and vectors, respectively. Let a PIP model be denoted by:

Max $x_0 = CX$

$$\text{Subject to } AX \leq b, X \geq 0 \text{ and integers.} \tag{3.1}$$

After introducing the slack variables, (3.1) can be expressed as:

Max $x_0 = CX$

$$\text{Subject to } AX = b, X \geq 0 \text{ and integers.} \tag{3.2}$$

The models (3.1) and (3.2) are equivalent, except for the dimensions of the matrix A and the vector X due to addition of slack variables in (3.2). Also an extreme point vertex and the associated basic feasible solution of (3.2) for the relaxed LP divides (n+m) variables into two subsets, m basic and n non-basic variables, where $A = (B, N)$ and $X = (X_B + X_{NB})$ so that

$$\mathrm{BX_B + NX_{NB} = b \text{ or } IX_B + (B^{-1}N)X_{NB} = B^{-1}b} \tag{3.3}$$

Similarly, all integer points in the feasible region also divide the (n+m) variables into two sub-sets: the set X represents the physical location of the point in the n dimensional space of (3.1), and the set including the slack variables represents a point in (n+m) dimensional space in (3.2). Thus, the two sets for all integer points do not have a sharp division of m and n as is the case for the LP extreme points.

Integer polyhedron is a convex space generated by joining the integer points in the feasible convex region of $AX \leq b, X \geq 0$ such that at least one higher integer point of the integer polyhedron is outside the convex region defined by (3.1). It means that if $x_j = \alpha_j, j = 1, 2, \ldots, n$ is an extreme point of the integer polyhedron, then there exists a point

$$x_j = \alpha_j + 1, \text{ or at least some } j, j = 1, 2, \ldots, n,$$

which is not in the LP feasible region. Let the LP convex space be denoted by S_{LP} and the integer-point convex space be denoted by S_{IP}. The integer polyhedron convex space will satisfy that it is a sub-space of the LP convex region, $S_{IP} \leq S_{LP}$. This inequality holds for all LP/IP models, including the uni-modular systems like transportation and assignment models, where $S_{LP} = S_{IP}$. The integer polyhedron has interesting properties. Proofs are given where necessary.

Property 3.1: All n dimensional integer points within the feasible region of the convex region defined by the constraints set $AX \leq b, X \geq 0$ of the given PIP, are such that the associated $(n + m)$ dimensional vectors with respect to (3.2) are also integer vectors.

Property 3.2: The search of an optimal integer point along the integer polyhedron should be free of fractional values. This follows immediately from Property 1 above.

Property 3.3: Given that matrix A is composed of integer elements, the selected pivot element a_{ij} is a non-negative integer element such that $a_{ij} \geq 1$. If $a_{ij} = 1$, the transformed matrix $\overline{A}$ as well as the RHS vector $\overline{b}$ will remain integer-valued elements after the pivoting operation. However, if the selected pivot element $a_{ij} > 1$, the transformed matrix after pivoting will contain fractional elements, hence it will not lead to an extreme point of the integer polyhedron. In that case, some modifications are required, as developed by two independent approaches, which are discussed in this Chapter.

Property 3.4: Fractional values can be avoided by using an additional constraint, described below. Assuming, that $b_i > 0$, let a pivot row, pivot element and pivot variable be given by:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n + s_i = b_i,\ a_{ij} \text{ and } x_j, \text{ respectively.}$$

When $a_{ij} > 1$, dividing the pivot row by a_{ij}, yields:

$$\frac{a_{i1}}{a_{ij}}x_1 + \frac{a_{i2}}{a_{ij}}x_2 + \ldots + x_j + \ldots + \frac{a_{in}}{a_{ij}}x_n + \frac{1}{a_{ij}}s_i = \frac{b_i}{a_{ij}} \tag{3.4}$$

Separating fractional and integer part in each term in (3.4), keeping the fractional parts positive, gives:

$$(d_{i1} + f_1)x_1 + (d_{i2} + f_2)x_2 + \ldots + x_j + \ldots + (d_{in} + f_n)x_n + (d_i + f_i)s_i = (d + f) \tag{3.5}$$

where $0 < f_1, f_2, \ldots, f_i, \ldots, f_n, f < 1$ and $d_{i1}, d_{i2}, \ldots, d_{in}, d_i, d$ are discrete constants. The element $d \geq 0$ in the RHS of (3.5), since $b_i > 0$. The relation (3.5) can be expressed as (3.6) as follows:

$$\begin{aligned} &d_{i1}x_1 + d_{i2}x_2 + \ldots + x_j + \ldots + d_{in}x_n + d_is_i - d \\ &= f - (f_1x_1 + f_2x_2 + \ldots + f_nx_n + f_is_i) \end{aligned} \tag{3.6}$$

Since LHS of (3.6) is an integer and the RHS will be less than the fractional part of the RHS of (3.5), it is easy to see that:

$$Max\,[f - (f_1x_1 + f_2x_2 + \ldots + f_nx_n + f_is_i)] = 0 \tag{3.7}$$

$$\Rightarrow d_{i1}x_1 + d_{i2}x_2 + \ldots + x_j + \ldots + d_{in}x_n + d_is_i - d \leq 0 \tag{3.7}$$

From (3.7), we get:

$$x_j \leq d - (d_{i1}x_1 + d_{i2}x_2 + \ldots + d_{in}x_n + d_is_i)$$

or equivalently,

$$x_j = d - (d_{i1}x_1 + d_{i2}x_2 + \ldots + d_{in}x_n + d_i s_i) - x'_j \tag{3.8}$$

where x'_j is also restricted to a nonnegative integer value. This constraint (3.8) have been used in different ways by Young (1965) and by Munapo, Kumar and Khan (2010). This additional constraint (3.8) allows us to move on the integer polyhedron. More on the application of (3.8) will be discussed in Sections (3.3) and (3.4).

It is important to note that the original pivot element which was greater than 1 is reduced to 1, hence fractions will not be generated.

Since the new variable x'_j may again be subjected similar situation as the variable x_j, it has been assumed that in general, x_j^{l+1} will be used to represent the new variable replacing the variable x_j^l, $l = 0, 1, 2, \ldots$ and $j = 1, 2, \ldots$ Note that $x_j \equiv x_j^0$.

Property 3.5: In LP, and as well as in PIP, each entry in a pivoting operation is a solution of the system of linear equations of the form $BX_B = P_i$ or b, which by Cramer's rule is given by $x_i = \det(B_i)/\det(B)$. Since in integer polyhedron case fractions do not arise, it leads to an interesting result that all bases for the integer polyhedron will be such that their $\det(B) = 1$, otherwise fractions will be generated.

3.1.2 Optimality of the solution

Since we are dealing with convex space generated by the linear constraints, the usual LP rules are applicable. In addition, the solution is required to satisfy the integer requirement. The selected pivot element may not satisfy the integer requirement. When the Property 3.3 is satisfied, the normal simplex iteration will be carried out, otherwise when fractions arise, one must develop a constraint (3.8) and carry out the iterations as discussed in the two approaches. When optimality conditions are satisfied, a search for the optimal solution over the integer polyhedron is complete.

3.2 Young's primal integer programming approach (1965)

Young (1965) selects the source column and row exactly in the same way as it is done for the LP model, but instead of pivoting on the selected pivot element, he developed a Gomory constraint and selected the pivot element from the Gomory constraint, which has the co-efficient value 1. Since in Young's approach the pivot element is always 1, the property 3 is satisfied. Hence Young's iteration transforms an integer matrix into another integer matrix, thus one is moving from one integer point to another integer point on the integer convex polyhedron. In this respect, the Young's method, and the method by Munapo, Kumar and Khan (2010) are seemingly similar, but are different as the substitution (3.8) retains the history of the iterative process. Since an integer point in n dimensional space in the positive quadrant will be an n element vector with value for each element ≥ 0, Young's tableau are subject to increase in dimension but the approach discussed in Section (3.3) does not increase dimension with each iteration.

3.2.1 Numerical Illustration of Young's primal approach

Consider a trivial example that was also used by Young (1965).
Max $x_0 = 3x_1 + x_2$,
subject to $2x_1 + 3x_2 \leq 6$,

$$2x_1 - 3x_2 \leq 3,$$

$$x_1, x_2 \geq 0 \text{ and integers.} \tag{3.9}$$

This problem can be easily analysed by graphical approach. Before attempting to Young's approach, let us investigate it graphically, as shown in Figure 3.1.

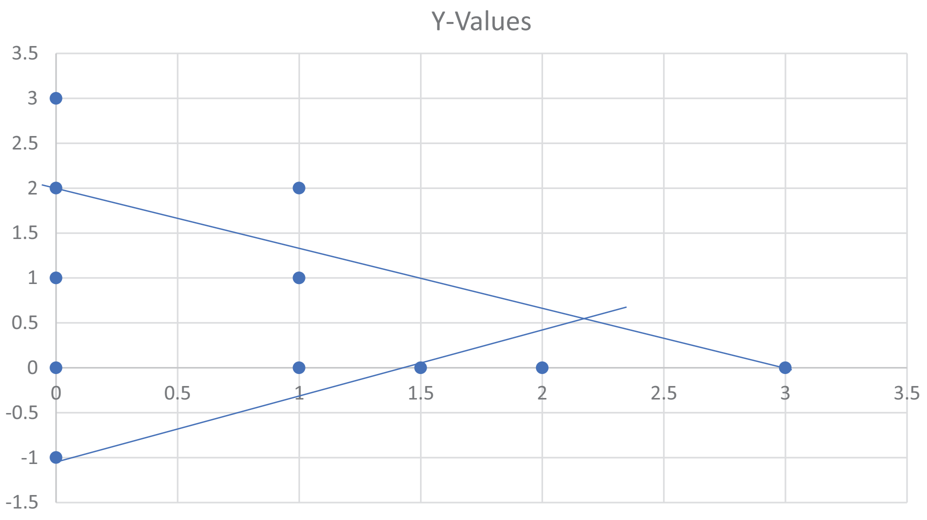


Figure 3.1: From the graph, one may note that the feasible space has only five integer points. This information is given in Table 3.1.

Now let us investigate the approach by Young (1965).

Let us define for convenience, the given variables are denoted by $x_j, j = 1, 2 \ldots, n$, (in the illustration, we have two variables.)

Table 3.1: Feasible points and value of the objective function.

Serial No	Feasible Pt	Z Value	Remarks
1	(0,0)	0	Not opt
2	(1,0)	3	Not opt
3	(0,1)	1	Not opt
4	(1,1)	4	Optimal
5	(0,2)	2	Not opt.

The slack variables are denoted by $s_i, i=1,2,\ldots m$ and slack variables required for the Gomory constraints be denoted by $t_j, j=1,2,\ldots,k$. We have two slack variables and other slack variables will unfold as we go through the search procedure.

The initial Tableau for the example (3.9) will be as shown in Table 3.2.

Table 3.2: Initial table for model (3.9).

	RHS	x_1	x_2	s_1	s_2
Z	0	−3	−1	0	0
s_1	6	2	3	1	0
s_2	3	2	−3	0	1

According to the usual simplex iterations, both variables x_1 and x_2 can enter the basis. If x_1 enters, the outgoing variable will be s_2 and similarly if x_2 enters, the variable to go out will be s_1.In both cases, the pivot element will be an element greater than 1, hence fractions will be generated and we will no longer be on the integer polyhedron. If arbitrarily, we select x_2 as the entering variable, we develop a constraint equivalent to (3.8), which is given by:

$$x_j = d - (d_{i1}x_1 + d_{i2}x_2 + \ldots + d_{in}x_n + d_i s_i) - x'_j$$

or for the above case, it will be:

$$\left[\frac{6}{3}\right] = \left[\frac{2}{3}\right]x_1 + \left[\frac{3}{3}\right]x_2 + \left[\frac{1}{3}\right]s_1 \Rightarrow 2 = x_2 + t_1.$$

This constraint is added to (3.9), resulting in the Table 3.3 given below.

Note that when x_2 enters the pivot element will be 1 as indicated by an asterisk mark and this pivot will take us to an improved integer point. The updated table is given in Table 3.4.

Table 3.3: After adding the additional constraint.

	RHS	x_1	x_2	s_1	s_2	t_1.
Z	0	−3	−1	0	0	0
s_1	6	2	3	1	0	0
s_2	3	2	−3	0	1	0
t_1.	2	0	1*	0	0	1

Table 3.4: Improved values after the pivot operation.

	RHS	x_1	x_2	s_1	s_2	t_1.
Z	2	−3	0	0	0	1
s_1	0	2	0	1	0	−3
s_2	9	2	0	0	1	3
x_2.	2	0	1	0	0	1

Table 3.5: Table after adding the extra constraint due to fractions.

	RHS	x_1	x_2	s_1	s_2	t_1	t_2
Z	2	−3	0	0	0	1	0
s_1	0	2	0	1	0	−3	0
s_2	9	2	0	0	1	3	0
x_2	2	0	1	0	0	1	0
t_2	0	1*	0	0	0	−2	1

Table 3.6: After the pivot operation.

	RHS	x_1	x_2	s_1	s_2	t_1	t_2
Z	2	0	0	0	0	−5	3
s_1	0	0	0	1	0	1*	−2
s_2	9	0	0	0	1	7	−2
x_2	2	0	1	0	0	1	0
x_1	0	1	0	0	0	−2	1

Table 3.4 indicates that we have not reached to an optimal solution. When variable x_1 enters, the outgoing variable will be s_1 and once again the pivot will lead to fractional values as the pivot element value is not equal to 1. This will need generating a new constraint and it will be $0 = x_1 - 2t_1 + t_2$. This constraint is added to the Table 3.4, resulting in Table 3.5 and after the pivot operation gives Table 3.6.

The solution in Table 3.6 is not yet optimal. The variable t_1 enters and s_1 will exit the basis. The pivot element is 1, shown by an * mark. No additional constraint is required. The pivot resulted in Table 3.7.

Table 3.7: Updated table.

	RHS	x_1	x_2	s_1	s_2	t_1	t_2
Z	2	0	0	5	0	0	−7
t_1	0	0	0	1	0	1	−2
s_2	9	0	0	−7	1	0	12*
x_2.	2	0	1	−1	0	0	2
x_1	0	1	0	2	0	0	−3

The pivot being not equal to 1, a new constraint will have to be added. It will give rise to new Table 3.8 given below:

Table 3.8: After adding the constraint.

	RHS	x_1	x_2	s_1	s_2	t_1	t_2	t_3
Z	2	0	0	5	0	0	−7	0
t_1	0	0	0	1	0	1	−2	0
s_2	9	0	0	−7	1	0	12	0
x_2.	2	0	1	−1	0	0	2	0
x_1	0	1	0	2	0	0	−3	0
t_3	0	0	0	−1	0	0	1*	1

The pivot operation give rise to Table 3.9.

Table 3.9: Updated table after the pivot operation.

	RHS	x_1	x_2	s_1	s_2	t_1	t_2	t_3
Z	2	0	0	−2	0	0	0	7
t_1	0	0	0	−1	0	1	0	2
s_2	9	0	0	5*	1	0	0	−12
x_2.	2	0	1	1	0	0	0	−2
x_1	0	1	0	−1	0	0	0	3
t_2	0	0	0	−1	0	0	1	1

It will give rise to one more additional constraint as shown in Table 3.10, which indicates the solution is not optimal. After the iteration, we get Table 3.11.

Table 3.10: Updated Table.

	RHS	x_1	x_2	s_1	s_2	t_1	t_2	t_3	t_4
Z	2	0	0	−2	0	0	0	7	0
t_1	0	0	0	−1	0	1	0	2	0
s_2	9	0	0	5	1	0	0	−12	0
x_2.	2	0	1	1	0	0	0	−2	0
x_1	0	1	0	−1	0	0	0	3	0
t_2	0	0	0	−1	0	0	1	1	0
t_4	1	0	0	1*	0	0	0	−3	1

Table 3.11: Final Tableau resulting in an optimal integer solution.

	RHS	x_1	x_2	s_1	s_2	t_1	t_2	t_3	t_4
Z	4	0	0	0	0	0	0	1	2
t_1	1	0	0	0	0	1	0	−1	1
s_2	4	0	0	0	1	0	0	3	−5
x_2.	1	0	1	0	0	0	0	1	−1
x_1	1	1	0	0	0	0	0	0	1
t_2	1	0	0	0	0	0	1	−2	1
s_1	1	0	0	1	0	0	0	−3	1

Solution summary is presented in Table 3.12

Table 3.12: Summary of all computations.

Reference Table	S. No	Solution x_1, x_2, s_1, s_2	Z	Remarks
T3.1	1	0,0,6,3	0	Not optimal
T3.3	2	0,1,0,9	2	Not optimal
T3.5	3	0,1,0,9	2	Degenerate
T3.7	4	0,1,0,9	2	Degenerate
T3.9	5	1,1,1,4	4	Optimal

Later Young expressed simplex tables as a function of non-basic variables as from all Tables, it may be noted that number of non-basic variables remain two. For example, the initial table will look like the Table 3.13, given below.

Table 3.13: Initial Table shown as a function of non-basic variables.

	1	$-x_1 \downarrow$	$-x_2$
X_0	0	−3	−1
X_1	0	−1	0
X_2	0	0	−1
S_1	6	2	3
S_2	3	2	−3
t_1	1	1*	−2

Note that S_1 and S_2 are the slack variables in the two inequalities. The variable X_1 is the entering variable and S_2 will be the outgoing variable by the conventional simplex rule. Since it will lead to fractions, Young developed a constraint, which results in the last row with pivot element as 1, resulting in no fractions by Property 3.

3.3 The Integer Polyhedron Search Algorithm (IPSA)

The Integer polyhedron search algorithm was developed by Munapo, Kumar and Khan (2010, 2012). Their approach, seemingly similar, to Young's approach; is different, and it was developed independently of the Young's approach. They also find the pivot element as is done in simplex iterations and when pivot element is not equal to 1, they develop a condition, and call it an upper bound constraint. This upper bound constraint is used to replace the variable and maintain integers values. Let us explain this point by using the example 3.1 and its initial Table 3.2.

As noted earlier, both variables x_1 and x_2 qualify for entry and both will lead to fractional values. If we let the variable x_1 enter the basis, the pivot row we have is:

$$2x_1 - 3x_2 + 0s_1 + s_2 = 3 \tag{3.10}$$

The relation (3.10) can be expressed as:

$$\left(\frac{2}{2}\right)x_1 - \left(\frac{3}{2}\right)x_2 + 0s_1 + \left(\frac{1}{2}\right)s_2 = \left(\frac{3}{2}\right)$$

Like in the Gomory constraint, separating integers and fractions, retaining fractional part positive, we have:

$$(1+0)x_1 + (-2+1/2)x_2 + 0s_1 + \left(0+\frac{1}{2}\right)s_2 = \left(1+\frac{1}{2}\right)$$

The above relation gives rise to an upper bound relation (3.11) given below:

$$x_1 - 2x_2 + x_1^1 = 1 \ or \ x_1 = 1 + 2x_2 - x_1^1 \tag{3.11}$$

When the variable x_1 in Table 3.2 is replaced by the relation (3.11), i.e. $x_1 = 1$ is replaced by x_1^1, it gives rise to Table 3.14.

Table 3.14: After replacing the variable x_1 by x_1^1 using (3.11).

	RHS	x_1^1	x_2	s_1	s_2
Z	3	+3	−7	0	0
s_1	4	−2	7*	1	0
s_2	1	−2	1	0	1

Not that table size remains unchanged, objective has improved, and we have an extra relation (3.11) in the memory. The integer point corresponding to Table 3.14 is given by:

$$x_1^1 = x_2 = 0, \ s_1 = 4, \ s_2 = 1, \text{ and } x_1 = 1.$$

The solution in Table 3.14 is not optimal. The variable x_2 enters and once again, pivot element is not equal to 1, we need to develop a new upper bound relation, which is given below:

$$-2x_1^1+7x_2+s_1=4\equiv\left(-\frac{2}{7}\right)x_1^1+x_2+\left(\frac{1}{7}\right)s_1=\left(\frac{4}{7}\right),$$

which leads to the relation (3.14) given by:

$$x_2=-x_2^1+x_1^1 \tag{3.14}$$

Substituting relation (3.14) in Table 3.14, we have Table 3.15.

Table 3.15: Obtained from Table 3.14 and relation (3.14).

	RHS	x_1^1	x_2^1	s_1	s_2
Z	3	−4	7	0	0
s_1	4	5*	−7	1	0
s_2	1	−1	−1	0	1

Now x_1^1 enters at its upper bound. It will give rise to a relation:

$$x_1^1=2x_2^1-x_1^2 \tag{3.15}$$

From the Table 3.14 and the relation (3.15), we get Table 3.15a.

Table 3.15a: After replacing the incoming variable at its upper bound.

	RHS	x_1^2	x_2^1	s_1	s_2
Z	3	4	−1	0	0
s_1	4	−5	3*	1	0
s_2	1	1	−3	0	1

The variable x_2^1 will be at its upper bound. The relation will be given by (3,16) and it will result in Table 3.16.

$$x_2^1=1+2x_1^2-x_2^2 \tag{3.16}$$

Table 3.16: Optimal solution.

	RHS	x_1^2	x_2^2	s_1	s_2
Z	4	2	1	0	0
s_1	1	1	−3	1	0
s_2	4	−5	3	0	1

The Optimal solution is given by:

$$S1 = 1,\ \ S2 = 4,\ \text{ and } x_1^2 = x_2^2 = 0 \tag{3.17}$$

When values in (3.17) are used in (3.16), we have $x_2^1 = 1$. These values are used in relation (3.15), we have $x_1^1 = 2$ and again go back to relation (3.14), we have $x_2 = 1$ and finally using the relation (3.11), we have $x_1 = 1$

Thus, the final optimal solution is:

$$x_1 = 1, x_2 = 1, s_1 = 1, and\ s_2 = 4, Z = 4 \text{ as was expected.}$$

3.3.1 Integer Polyhedron Search Algorithm (IPSA) by Munapo, Kumar and Khan (2010)

From the properties discussed in Section 3.2 and from the illustration given above, the integer polyhedron search algorithm can be summarized in the following steps:

Step 1: Assume that all $b_i \geq 0$. Select a pivot element as is selected in the simplex method. If the pivot element is >1, go to Step 2; else when it is l, go to Step 3.

Step 2: Develop a new upper bound constraint, similar to (3.8), i.e. $x_j = d - (d_{i1}x_1 + d_{i2}x_2 + \ldots + x_j + \ldots + d_{in}x_n + d_i s_i) - x_j^1$ and replace the pivot variable x_j by this upper bound relation. The number of variables will remain unchanged and the variable x_j is replaced by the variable x_j^1. Keep this linear relation in memory and go to Step 4.

Step 3: Apply the usual pivot operation used in the simplex procedure and go to Step 4.

Step 4: Check if the objective row has nonnegative elements? If yes, then an optimal integer solution has been obtained, else return to Step 1.

The flow diagram for the algorithm is shown in Figure 3.2.

Further reading on Gomory cuts can found in the work by Gomory (1965, 1969); Balas *et al.* (1996), Balas (1971), Salkin (1971), Gomory *et al.* (2003) and Kumar *et al.* (2008).

3.4 More numerical illustrations of IPSA

Consider two simple examples taken from Winston (2004). Example 3.4.1 illustrates that IPSA is insensitive with regard to the magnitude of fractions compared to the Branch and Bound (BB) approach, and Example 3.4.2 illustrates a degenerate case handled by the IPSA.

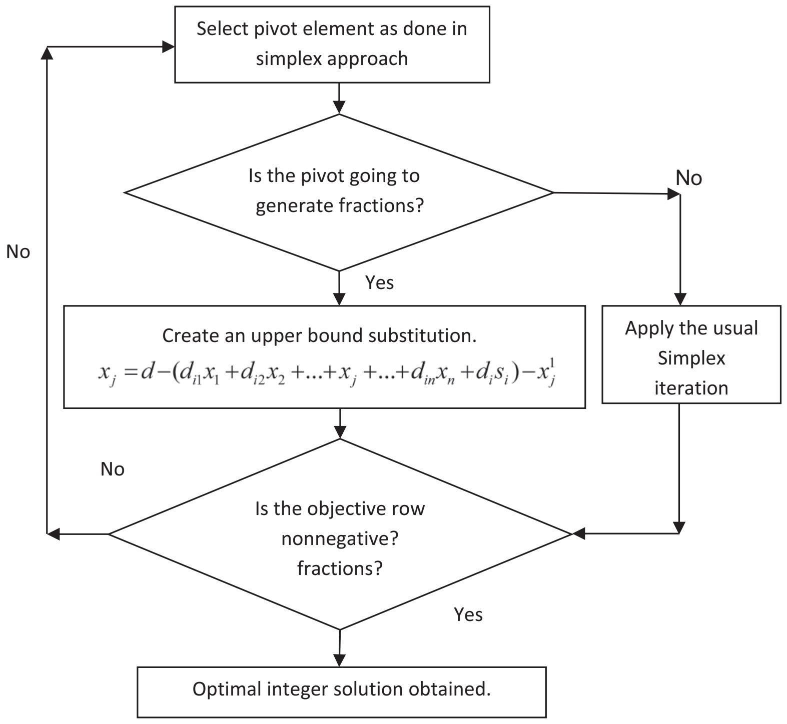

Select pivot element as done in simplex approach
Is the pivot going to generate fractions?
No
Yes
No
Create an upper bound substitution.
$x_j = d - (d_{i1}x_1 + d_{i2}x_2 + ... + x_j + ... + d_{in}x_n + d_i s_i) - x_j^1$
Apply the usual Simplex iteration
No
Is the objective row nonnegative? fractions?
Yes
Optimal integer solution obtained.

Figure 3.2: Integer Polyhedron Search Algorithm.

Example 3.4.1
Maximize $Z = 8x_1 + 4x_2$
Such that $x_1 + x_2 \leq 5$

$$9x_1 + 4x_2 \leq 40 \tag{3.18}$$

where $x_1, x_2 \geq 0$ and integer.

The problem (3.18) can be rearranged as a simplex tableau shown in Table 3.17.

As per the simplex method, one would select pivot element 9 in the last row. However, since it will result in fractional values, a corresponding upper bound constraint for the variable x_1 is developed. It is given by $x_1 \leq 4,$ or equivalently (3.19).

Table 3.17: Initial Tableau for LIP (3.18).

	x_1	x_2	s_1	s_2	RHS
Z	−8	−4	*0*	*0*	*0*
S_1	1	1	*1*	*0*	*5*
S_2	9*	4	*0*	*1*	*40*

$$x_1 = \mathbf{4} - x_1^1 \tag{3.19}$$

Here x_1^1 is a nonnegative integer variable. Substituting (3.19) in Table 3.17, generates Table 3.18.

Table 3.18: First Iteration of the Search.

	x_1^1	x_2	s_1	s_2	RHS
Z	8	−4	*0*	*0*	*32*
S_1	−1	1*	*1*	*0*	*1*
S_2	−9	4	*0*	*1*	*4*

From Table 3.18, the next pivot element to be selected has a tie. It can be 1 in the second row or 4 in in row 3. Since pivot is 1, it will not give rise to fractions; hence Step 2 will be applicable. The outcome is shown in Table 3.19.

Table 3.19: Second Iteration of the Search.

	x_1^1	x_2	s_1	s_2	RHS
Z	4	0	*4*	*0*	*36*
X_2	−1	1	*1*	*0*	*1*
S_2	−5	0	-4	*1*	*0*

This is an optimal solution to problem (3.19). It results in

$$x_1^1 = 0 \Rightarrow x_1 = 4, x_2 = 1, s_1 = s_2 = 0, Z = 36$$

The above simple example was experimented with the BB approach. It was noticed that if we change slightly some elements, the BB method requires relatively more effort before identifying the optimal solution. It is summarized in Table 3.20.

From Table 3.20, it is clear, that although optimality was achieved at iteration 2, its verification took longer, whereas the effort required in the IPSA is independent of minor changes in the RHS. More experiments are desired.

Table 3.20: Summary of Results from the BB Method.

Changes in RHS	Solution of sub-problem that resulted in the optimal solution	Optimality verified at iteration number	Iterations in the proposed method
No change – Numerical example with $b_1=5, b_2=40$	1	2	2
$b_1=6, b_2=40$	2	3	2
$b_1=6, b_2=41$	2	7	2
$b_1=6, b_2=42$	2	7	2
$b_1=7, b_2=42$	2	11	2

Example 3.4.2 Consider again one more example taken from Winston (2004).
Maximize $Z=x_1+2x_2$
Such that

$$\mathbf{x_1+x_2\le 10,\ \ 2x_1+5x_2\le 30} \tag{3.20}$$

$x_1,x_2\ge 0$ and integer.

The solution to Example 3.4.2 is summarized in Table 3.21.

Table 3.21: Summary of the Solution for Example 3.5.2.

Iteration number	Basis/ Non-basic	Value of basic variables	Pivot and its position	Upper bound constraint	Solution	Optimality of the solution
Initial	$Z,s_1,s_2/$ x_1,x_2	0, 10, 30	5 in row 3, col. 2	$x_2=6-x_2'$	$x_1=x_2=0$	Not optimal
Iteration 1	$Z,s_1,s_2/$ x_1,x_2'	12, 4, 0	2 in row 3 Col. 1	$x_1=3x_2'-x_1'$	$x_1=0,$ $x_2=6$	Not optimal
Iteration 2	$Z,s_1,s_2/$ x_1',x_2'	12, 4, 0	1 in row 3 Col. 2	Not required	Same as above due to degeneracy	Not optimal
Iteration 3	$Z,s_1,x_2'/$ x_1',s_2	12, 4, 0	3 in row 2 col. 1	$x_1'=1+s_2-x''_1$	Same as above due to degeneracy	Not optimal
Iteration 4	$Z,s_1,x_2'/$ x''_1,s_2	13, 7, 2	NA	NA	$x_1=5,x_2=4$ $Z=13$	Optimal

The above problem, when solved by the BB method, creates 8 sub-problems before optimality is reached.

3.5 Concluding remarks

The process has many interesting features.

a. Since one is not dealing with fractions, there are no round-off errors in the process.
b. The number of variables grows, but the dimension of matrix A remains unchanged.
c. There is neither a sub-problem nor any growth in the number of these sub-problems at any iteration, which is not true for the branch and bound, cutting plane and hybrid approaches.
d. Need for an initial good solution is not crucial.
e. The proposed algorithm makes a direct search over the integer polyhedron, it is not dependent on the continuous LP optimal solution.
f. The bounded variable constraint embedded within the approach executes an iteration faster, compared to a normal pivoting operation.
g. The approach is suitable for solving protean integer programs as information is never lost.
h. Improvements in computational efficiency, if any, will only be known after development of the software, testing and handling larger problems. This remains as a task for future investigations.

References

Balas, E., Ceria, S., Cornnejols, G. and Natraj, N. (1996) Gomory cuts revisited, Operations Research Letters, 19(1), pp 19 –39.
Balas, E., (1971) Intersection cuts – a new type of cutting planes for integer programming, Operations Research, 19(1), pp 1 –9.
Beasely, J.E., (Ed.) (1996) Advances in Linear and Integer programming, Oxford University Ptress.
Edelsbrunner, H., (1987) Algorithms in combinatorial Geometry, Springer-Verlag.
Gomory, R.E., (1965) On the relation between integer and non-integer solutions to a linear program, Proc. of the National Academy of Sciences, 53(2), pp 260 –265.
Gomory, R.E., (1969) Some polyhedral relations to combinatorial problems, Linear Algebra and its Applications, 2(4), pp 451–458.
Gomory, R.E., Jonson, E.L. and Evans, L. (2003) Corner polyhedral and their connections with integer programming, Mathematical programming, Ser B, 96, pp 321 –339.
Hillier, F. and Lieberman, S. (2005) Introduction to Operations Research, McGraw Hill, 8th Edition.
Karlof, L.K., (2005) Integer programming: Theory and Practice, CRC Press
Kumar, S., Luhandjula, M.K., Munapo, E., and Jones, B.C., (2010) Fifty years of Integer Programming: A review of Solution Approaches, Asia Pacific Business Review, Vol. 6, No. 2, pp 5–15.
Kumar, S., Munapo, E., Lesaoana, M. and Nyamugure, P., (2018) Some Innovations in OR Methodology: Linear Optimization, Lambert Academic Publishing, ISBN 978-613-7-38007-9.
Munapo, E., Kumar, S. and Khan, L. (2010). Pure integer programming on Integer Polyhedron, AIMS International Journal of Management, 4(2), pp 135–44.

Munapo, E., Kumar, S. and Khan, L. (2012). Information Recycling on Integer Polyhedron in Protean Environment, International Journal of Mathematical Modelling, Simulation and Applications, 5(1), pp 41–51.

Nemhauser, G.L. and Wolsey, L.A., (Eds.) (1988) Integer and Combinatorial Optimization, Interscience Series in Discrete Mathematics and Optimization, John Wiley and Sons

Salkin, H. M., and Mathur K., (1989) Foundations of Integer Programming, North Holland.

Schrijiver, A., (1986) Theory of linear and Integer Programming, John Wiley, New York.

Young, R.D. (1965) Primal integer programming algorithms, Journal of Research of National Bureau of Standards – B Mathematics and Mathematical Physics, Vol 69 B, No. 3, pp213–251.

Young, R.D. (1968) A simplified primal (all-integer) integer programming algorithm, J of ORSA, 16(4), pp750–782.

Young, R.D. (1969) Primal integer programming, Chapter in TC Hu, Integer programming and Network Flows, Addison Wesley, pp 287–309.

Chapter 4
Use of variable sum limits to solve the knapsack problem

Abstract: This chapter presents a technique for determining variable sum limits and then use these limits to solve the knapsack problem in a branch and bound related setting. Variable sum limits idea was used by Munapo (2020) to improve optimality verification of a knapsack model. In this chapter, variable sum limits of subsets of variables have been used. The strength of using variable sum limits is that determining a variable sum limit is a process that can be done independently. Taking advantage of the availability of massively parallel processors in modern computers, optimality in knapsack problems can be verified using a significantly small number of sub-problems.

Keywords: Knapsack problem, Variable sum limits, Sub-problems, Parallel processing

4.0 Introduction

A general linear integer problem (LIP) is one of those difficult problems that have been attracting active research for a very long time without a breakthrough. It is our belief that an effective general-purpose algorithm for this model has not been developed. Some researchers believe that such a general-purpose algorithm may not exist.

This chapter presents a technique for determining variable sum limits and then use these limits to solve the knapsack problem in a branch and bound related environment. The concept of variable sum limits has been used by Munapo (2020a, 2020b) for improving optimality verification. In this chapter variable sum limits of subsets of variables has been used. The strength of using variable sum limits is that these limits can be processed independently, talking advantage of the availability of massively parallel processors in modern computers, see Vasilchikov (2018). Using these limits, the optimality in knapsack problems can be verified using significantly a small number of sub-problems. The branch and bound algorithm was developed by Land and Doig (1960) and later it was improved to solve the mixed integer problems by Dakin(1965). Improvement on the branch and bound approach is an ongoing research, see Al-Rabeeah et al. (2019), Bhattacharjee and Sarmah (2014) and Munapo and Kumar (2016).

https://doi.org/10.1515/9783110703023-004

4.1 The knapsack model

Mathematical model of a knapsack problem is given by (4.1).

Minimize

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

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \geq b. \tag{4.1}$$

Where a_j, b and c_j are given nonnegative constants and $x_j \geq 0$, and integers.

4.2 Development of the variable range for a knapsack problem

4.2.1 Variable range

In general, limits on range of a variable are given in (4.2).

$$\ell_j^a \leq x_j \leq \ell_j^b. \tag{4.2}$$

Where ℓ_j^a and ℓ_j^b are lower and upper bounds for the variable x_j. In general, the lower limit for the variable $x_j, j = 1, 2, \ldots, n$ is as shown by (4.3).

$$(\ell_j^a = 0). \tag{4.3}$$

Similarly, the largest possible value for the variable x_j as its upper limit can be determined as (4.4), which can be obtained as.

$$\ell_j^b = I_j + 1. \tag{4.4}$$

Where $\frac{b}{c_j} = I_j + f$ and f is the fractional part of $\frac{b}{C_j}$.

These variable limits ℓ_j^a and ℓ_j^b for the variable x_j are two wide to show any advantage in their current form. However, there is a need to narrow it down and fortunately it is possible. The variable range for the variable x_j is given by (4.5).

$$R[x_j] = \ell_j^b - \ell_j^a. \tag{4.5}$$

4.2.2 Objective value upper bound

For a given knapsack problem, the variables, variable upper limits and values of the objective function can be summarized as given in Table 4.1.

Table 4.1: Variables, variable upper limits and values of the objective.

Variable (x_j)	x_1	x_2	...	x_n
Upper limit (ℓ_j^b)	ℓ_1^b	ℓ_2^b	...	ℓ_n^b
Objective value (Z_j)	$Z_1 = c_1\ell_1^b$	$Z_2 = c_2\ell_2^b$	...	$Z_n = c_n\ell_n^b$

In Table 4.1, the integral upper limits are used to calculate the *n* values of the objective function. The most likely basic variable can be identified using the minimum objective value (Z_j^{UB}) given by (4.6).

$$Z_j^{UB} = Min[Z_1, Z_2, ..., Z_n]. \tag{4.6}$$

The objective upper bound constraint can be formulated as given in (4.7).

$$c_1x_1 + c_2x_2 + ... + c_jx_j + ... + c_nx_n \leq Z_j^{UB}. \tag{4.7}$$

4.2.3 Objective value lower bound

The objective lower bound (Z^{LB}) is obtained by solving (4.8) and must be adjusted to an integer value.

Minimize

$$Z = c_1x_1 + c_2x_2 + ... + c_jx_j + ... + c_nx_n,$$

Such that:

$$a_1x_1 + a_2x_2 + ... + a_jx_j + ... + a_nx_n \geq b. \tag{4.8}$$

Where $x_j \geq 0$ and relaxed.

In this case (Z^{LB}) is rounded up and the resulting constraint is given in (4.9).

$$c_1x_1 + c_2x_2 + ... + c_jx_j + ... + c_nx_n \geq Z^{LB}. \tag{4.9}$$

Challenge

The region to be searched is given by (4.10).

$$Z^{LB} \leq c_1x_1 + c_2x_2 + ... + c_jx_j + ... + c_nx_n \leq Z_j^{UB}. \tag{4.10}$$

The obvious challenge is to search this whole convex region. For small-sized knapsack models, this may not be difficult but for large-sized models, it poses a challenge.

4.2.4 How to overcome this challenge?

A way to alleviate this challenge is to divide this convex region into two parts. Since we expect the optimal integer solution to be near the upper bound, Figure 4.1 may be a possible ratio of the distances of the optimal solution (Z) from upper bound and the continuous optimal point. Let the distance of the upper bound from the continuous optimal solution or lower bound be D.

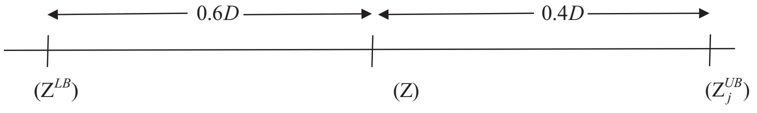


Figure 4.1: Distances from the lower bound.

In other words, the two regions that are searched separately are given by (4.11) and (4.12).

$$Z^{LB} \le Z \le Z^{LB} + 0.9D. \tag{4.11}$$

$$Z_j^{UB} - 0.1D \le Z \le Z_j^{UB} \tag{4.12}$$

The values can be easily adjusted to integers.

4.2.5 Variable sum bounds and subsets

Variable sum bounds

To determine variable bounds, we need all corner points of the convex region. This is a challenge as it is computationally expensive to calculate all corner points. To alleviate this difficult process, we can use the bounds of sums of variables instead. These are easier to calculate. Let the 0.6D convex region be region be denoted by A and the 0.4D region be denoted by B. Then the variable sum ranges can be found as given by (4.13) to (4.16).

Region A

Minimize

$$\lambda_1^A = x_1 + x_2 + \ldots + x_j + \ldots + x_n,$$

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \ge b. \tag{4.13}$$

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

$$Z^{LB} \le Z \le Z^{LB} + 0.6D.$$

Where λ_1^A is the lower variable sum bound.

Maximize

$$\lambda_2^A = x_1 + x_2 + \ldots + x_j + \ldots + x_n,$$

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \geq b. \tag{4.14}$$

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

$$Z^{LB} \leq Z \leq Z^{LB} + 0.6D.$$

Where λ_2^A is the upper variable sum bound.

Region B

Minimize

$$\lambda_1^B = x_1 + x_2 + \ldots + x_j + \ldots + x_n,$$

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \geq b. \tag{4.15}$$

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

$$Z_j^{UB} - 0.4D \leq Z \leq Z_j^{UB}.$$

Where λ_1^B is the lower variable sum bound.

Maximize

$$\lambda_2^A = x_1 + x_2 + \ldots + x_j + \ldots + x_n,$$

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \geq b. \tag{4.16}$$

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

$$Z_j^{UB} - 0.4D \leq Z \leq Z_j^{UB}.$$

Where λ_2^A is the upper variable sum bound.

Once the variable sum bounds for the two regions are available the two regions can be searched efficiently for the optimal solution.

4.2.6 Subsets of variable sum bound

In section 4.2.5 the sum of all variables i.e. $x_1 + x_2 + \ldots + x_j + \ldots + x_n$ is used in bounding. In subset variable sum bounding only a partial number of variables $x_1 + x_2 + \ldots + x_j$ are used in bounding. This is very important as some of the variables can be fixed resulting in unnecessary computations being avoided.

From (4.14) and (4.16), we can find the lower bound of a fraction of the variables, $x_1 + x_2 + \dots + x_j$. Let the lower bound be of this fraction of variables be from the Region A be λ_{f1}^{A} and that from the Region B be λ_{f1}^{B}. Thus for regions A and B we have (4.17) and (4.18) respectively as lower bounds.

$$0 \le \lambda_{f1}^{A}. \tag{4.17}$$

$$0 \le \lambda_{f1}^{B}. \tag{4.18}$$

The upper bound for λ_{f1}^{A} and λ_{f1}^{B} can be found using (4.19) and (4.20), respectively.

Maximize

$$Z_{f1}^{A} = x_1 + x_2 + \dots + x_j,$$

Such that:

$$a_1x_1 + a_2x_2 + \dots + a_jx_j + \dots + a_nx_n \ge b. \tag{4.19}$$

$$Z = c_1x_1 + c_2x_2 + \dots + c_jx_j + \dots + c_nx_n,$$

$$Z^{LB} \le Z \le Z^{LB} + 0.6D.$$

Maximize

$$Z_{f1}^{B} = x_1 + x_2 + \dots + x_j,$$

Such that:

$$a_1x_1 + a_2x_2 + \dots + a_jx_j + \dots + a_nx_n \ge b. \tag{4.20}$$

$$Z = c_1x_1 + c_2x_2 + \dots + c_jx_j + \dots + c_nx_n,$$

$$Z_j^{UB} - 0.4D \le Z \le Z_j^{UB}.$$

In other words the upper bound for λ_{f1}^{A} and λ_{f1}^{B} are Z_{f1}^{A} and Z_{f1}^{B},respectively. Thus, the selected fraction of variables can be bounded as given by (4.21) and (4.22).

$$0 \le \lambda_{f1}^{A} \le Z_{f1}^{A}. \tag{4.21}$$

$$0 \le \lambda_{f1}^{B} \le Z_{f1}^{B}. \tag{4.22}$$

4.3 Variable sum bounding algorithm

4.3.1 Algorithm

Given any knapsack problem of the form:

Minimize

$$Z = c_1x_1 + c_2x_2 + \dots + c_jx_j + \dots + c_nx_n,$$

Such that:

$$a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \geq b.$$

Where $x_j \geq 0$ and integer.

The variable sum bounding algorithm can be summarized as given below.

Step 1: Relax knapsack problem from integer to real values and calculate Z^{LB} and Z_j^{UB}.
Step 2: Use the calculated Z^{LB} and Z_j^{UB} values to determine λ_1^A and λ_2^A for Region A and λ_1^B and λ_2^B for Region B.
Step 3: Fix some variables with large Z values in Table 7 of Step 1.
Step 4: Search Region A using (4.23) and Region B using (4.24).

Minimize

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

Such that:

$$\begin{gathered} a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \geq b, \\ \lambda_1^A \leq x_1 + x_2 + \ldots + x_j + \ldots + x_n \leq \lambda_2^A, \\ Z^{LB} \leq Z \leq Z^{LB} + 0.6D. \\ 0 \leq \lambda_{f1}^A \leq Z_{f1}^A. \end{gathered} \tag{4.23}$$

Minimize

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

Such that:

$$\begin{gathered} a_1x_1 + a_2x_2 + \ldots + a_jx_j + \ldots + a_nx_n \geq b, \\ \lambda_1^B \leq x_1 + x_2 + \ldots + x_j + \ldots + x_n \leq \lambda_2^B, \\ Z_j^{UB} - 0.4D \leq Z \leq Z_j^{UB}. \\ 0 \leq \lambda_{f1}^B \leq Z_{f1}^B. \end{gathered} \tag{4.24}$$

Theorem: Any feasible integer solution in Region A is better than any feasible integer solution in Region B.

Proof: From $Z^{LB} < Z_A < Z_B < Z_j^{UB}$. Where Z_A and Z_B are optimal solutions in Region A and Region B, respectively.

4.4 Numerical illustration

Solve the knapsack problem given by (4.25) using the variable sum bounding algorithm.

Minimize

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897. \tag{4.25}$$

Where $x_j \geq 0 \forall j$ and integer.

Direct branch and bound algorithm

Solving directly by the branch and bound method the worst case takes 1901 sub-problems to verify optimality given by (4.26).

$$x_2 = 354, x_7 = 3, x_8 = 1, x_1 = x_3 = x_4 = x_5 = x_6 = 0{:}Z = 7884. \tag{4.26}$$

Solution by the proposed method - Variable sum bounding algorithm

Step 1: $Z^{LB} = 7881.50,$ which is adjusted to $Z^{LB} = 7882.$

Table 4.2: Objective values.

Z_j	10248	7898	13419	8376	28514	9768	7984	8298
x_j	244	359	497	349	538	264	308	461

$\therefore Z_j^{UB} = 7898.$

Step 2: Region A

From,

Minimize

$$\lambda_1^A = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8,$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897.$$

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$
$$7882 \leq Z \leq 7882 + 0.6(16).$$

$\therefore \lambda_1^A = 353.50,$ which is adjusted to $\lambda_1^A = 354.$

From,

Minimize

$$\lambda_2^A = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8,$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897.$$

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$
$$7882 \leq Z \leq 7882 + 0.6(16).$$

$\therefore \lambda_2^A = 360.62,$ which is adjusted to $\lambda_2^A = 360.$

Step 2: Region B

From,

Minimize

$$\lambda_1^B = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8,$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897.$$

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$

$7882 \leq Z \leq 7898$, resulting in D = 16.

$\therefore \lambda_1^B = 350.$

From,

Minimize

$$\lambda_2^B = x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8,$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897.$$

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$
$$7892 \leq Z \leq 7898.$$

$\therefore \lambda_2^B = 362.37,$ which is adjusted to $\lambda_2^B = 362.$

Step 3: From Table 4.2 obtained from Step 1, variables x_1, x_3, x_5 and x_6 are selected for fixing.

From Region A

Minimize

$$Z_{f1}^A = x_1 + x_3 + x_5 + x_6$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897.$$

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$
$$7882 \leq Z \leq 7891.$$

i.e. $Z_{f1}^A = 1.35,$ i.e. $0 \leq \lambda_{f1}^A \leq 1$ or $x_1 + x_3 + x_5 + x_6 \leq 1.$

From region B

Minimize

$$Z_{f1}^B = x_1 + x_3 + x_5 + x_6$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897.$$

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$
$$7892 \leq Z \leq 7898.$$

i.e. $Z_{f1}^B = 2.34$, i.e. $0 \leq \lambda_{f1}^B \leq 2$ or $x_1 + x_3 + x_5 + x_6 \leq 2$

Step 4: Searching Region A using (4.25) and Region B using (4.26).
Region A:
Minimize

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897,$$

$$350 \leq x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 362, \tag{4.27}$$
$$7882 \leq Z \leq 7891,$$
$$x_1 + x_3 + x_5 + x_6 \leq 1.$$

Optimal solution Region B.

$$x_2 = 354, x_7 = 3, x_8 = 1, x_1 = x_3 = x_4 = x_5 = x_6 = 0\text{: } Z = 7884.$$

A total of 313 sub-problems are required to verify this optimal solution.
Region B:
Minimize

$$Z = 42x_1 + 42x_2 + 27x_3 + 24x_4 + 53x_5 + 37x_6 + 26x_7 + 18x_8,$$

Such that:

$$53x_1 + 36x_2 + 26x_3 + 37x_4 + 24x_5 + 49x_6 + 42x_7 + 28x_8 \geq 12897,$$

$$350 \leq x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \leq 362. \tag{4.28}$$
$$7892 \leq Z \leq 7898.$$
$$x_1 + x_3 + x_5 + x_6 \leq 2.$$

Optimal solution Region B.

$$x_2 = 328, x_7 = 26, x_1 = x_3 = x_4 = x_5 = x_6 = x_8 = 0\text{: } Z = 7892.$$

A total of 155 sub-problems are required to verify this optimal solution.

By Theorem 1, we ignore optimal solution from Region B since there is a feasible one in Region A.

4.4.1 Optimality

The convex region is divided into two regions A and B and the searched separately. The two optimal solutions from the two regions are compared and the overall optimal solution selected.

4.5 Conclusions

The proposed approach is taking advantage of the current and future computer parallel processing power. Determining a variable sum limit is a process that can be done independently. More efficient hybrids of the branch such as branch & cut, branch & price and branch, cut and price can be used within the context of the proposed approach.

Reducing the number of sub-problems required to verify optimality from 1901 to (313+165) is an improvement, which is worth further considerations. More work to improve the proposed algorithm should be of interest to all researchers in the field.

References

Al-Rabeeah, M., Kumar, S., Al-Hasani A., Munapo, E. & Eberhard, A. 2019. Computational Enhancement in the Application of the Branch and Bound Method for Linear Integer Programs and Related Models, International Journal of Mathematical, Engineering and Management Sciences Vol. 4, No. 5, 1140–1153, 2019 https://dx.doi.org/10.33889/IJMEMS.2019.4.5-090

Bhattacharjee, K.K., Sarmah, S.P. 2014. Shuffled frog leaping algorithm and its application to 0/1 knapsack problem, Applied Soft Computing, Vol. 19. 252–263.

Dakin, R.J., 1965. A tree search algorithm for mixed integer programming problems. *The Computer Journal* 8, 250–255.

Land, A.H., A.G. Doig. 1960. An Automatic method for solving discrete programming problems, *Econometrica* 28, 497–520.

Munapo, E. 2020a. Development of a dummy guided formulation and exactsolution method for TSP, Eastern-European Journal of Enterprise Technologies, pp. 12–19.

Munapo, E. 2020b. Improving the Optimality Verification and the Parallel Processing of the General Knapsack Linear Integer Problem, Research Advancements in Smart Technology, Optimization, and Renewable Energy, Chapter 3, IGI Global.

Munapo, E. and Kumar, S. 2016. Knapsack constraint reformulation: A new approach that significantly reduces the number of sub-problems in the branch and bound algorithm, Cogent Mathematics, Vol. 3, 1162372, 2016.

Vasilchikov, V. 2018. On a Recursive-Parallel Algorithm for Solving the Knapsack Problem. Automatic Control and Computer Sciences. 52. 810–816. 10.3103/S014641161807026X.

Chapter 5
The characteristic equation for linear integer programs

Abstract: This chapter develops an innovative relation, called the characteristic equation (CE) for solving a pure integer program (PIP) and illustrates how the characteristic equation not only finds an optimal integer solution to a PIP but it also can find the ordered k^{th} *best*, $k \geq 2$ optimal solution, which is a feature not available to other approaches. The CE approach has been discussed for pure integer, binary integer, and mixed-integer models. A potential application of CE in context of a bi-objective and multi-objective models has been pointed out.

Keywords: Pure, Binary and Mixed integer programs, Descending hyperplane, The Characteristic equation, The k^{th}*best* optimal integer solution, Ordered tree search approach, Bi-objective and multi-objective integer problems

5.0 Introduction

In many real-life situations, integer restricted values are a natural consequence for interpretation and implementation of solutions obtained from mathematical models that are developed for analysis of complex industrial and business situations. In response to challenges created by industrial and real-life situations, many methods have been developed and discussed in the literature on integer programming, see Winston (2004), Kumar et al. (2010, 2018). Many of these methods along with their improved versions have been discussed in other chapters in this book. In this chapter we develop an innovative linear relation, that provides not only an optimal integer solution, but it can also provide $k^{th}best(k \geq 2)$, ordered optimal solution. This linear relation is called the characteristic equation (CE) for the linear integer linear programming model. The feature of determining the $k^{th}best(k \geq 2)$ solution **is** unique to the characteristic equation approach, which is not available to any other optimization method, including linear integer and related models. This feature of providing the $k^{th}best(k \geq 2)$ solution has an interesting application in determination of the non-dominated solution set for a multi-objective linear integer programming model.

Before presenting the characteristic equation approach, let us reexamine broadly the philosophy used by earlier researchers for finding an optimal linear integer solution to a given linear integer programming model. In these programming approaches, the authors have first identified the optimal LP solution and from that solution

https://doi.org/10.1515/9783110703023-005

they moved on towards the desired integer optimal solution in different ways, for example:

1. Gomory (1958) proposed cuts to reach the integer point from the LP optimal solution. Gomory cuts iteratively reduce the feasible space, making sure that no integer point is removed from the feasible space. These cuts develop a path from the LP optimal solution towards the required integer optimal solution. However, the problem dimension increases and guarantee for convergence is not provided. This approach forces the integer optimal point to become an extreme point of the modified convex space and that optimal point is reached by using the LP approach.
2. Another heavily used approach for determination of an optimal integer point is the branch and bound (B & B) approach. In this approach, once again the problem is solved as a LP and if the optimal solution involves some fractional values, integer bounds are introduced on variables that take fractional values out of the feasible region and create sub-problems by adding new constraints like: $x_j \leq I_j$ and $x_j \geq I_j + 1,$ where I_j represents the largest integer value of the fractional valued variable x_j. This approach once again, develops integer extreme point and recognizes optimality of the integer solution by using the linear programming technique. The branch and bound approach, once again, suffers from a disadvantage that problem size keeps on increasing with each iteration, and the number of problems also increases.
3. Many variations of the above approaches have been discussed in the literature and from time to time, many other heuristics were also developed using the above two ideas, see Hillier and Lieberman (2004), Kumar et al. (2018), Winston (2014).

The CE approach discussed in this chapter also uses the LP output of the simplex method, i.e. first the given PIP model is solved under a LP relaxed condition. It is also known that the LP optimal tableau obtained by using the simplex technique has many valuable information associated with various elements in the final output table. The CE approach uses some more untapped information for obtaining an optimal integer and other ranked-optimal solutions. This chapter explores that hidden information from the LP optimal simplex tableau and establishes one more useful linear relation from the output to identify not only the optimal integer solution but all other integer restricted $k^{th} best(k \geq 2)$ optimal solutions.

The output of the simplex tableau provides an insight on position of the integer points in the feasible LP space. The following observations are immediate by the nature of the linear inequalities:

1. A linear inequality constraint can be changed to an equation after adding a slack variable. Different values of this slack variable change the physical location of the given constraint, but the changed position with different values of the slack variable will generate a set of parallel hyperplanes.

2. Therefore, one can reach to an integer point within the feasible convex region defined by the LP constraints by assigning an appropriate value to the slack variables. Also note that the appropriate value to be assigned to the slack variable will also be an integer restricted value. When all slack variables have an appropriate integer values assigned to them, they will make all these constraints active at the integer point, one is looking for. Therefore, in one attempt, it becomes possible to reach the required integer point, unlike the Gomory or the B & B approach. Therefore, the problem size does not increase, as was the situation with Gomory, B & B and other approaches, which are the variant of these two approaches, developed as refinements.
3. The problem is to find the correct values for the non-basic variables for different constraints. The characteristic equation (CE) exactly fulfills this objective. Once the objectives are clearly understood, we are ready to understand, how we can find these exact values.

This chapter has been organized in 6 sections. In section 5.1, we discuss the characteristic equation for a pure linear integer model. Ordered tree search approach for an integer solution to CE was developed by Munapo et al. (2009) and it is discussed in section 5.2. In section 5.3, we have presented a CE for a binary integer problem and in section 5.4 we present an application of the CE in context of a bi-objective linear integer model. Finally, in section 5.5 we discuss CE for a mixed-integer model and conclude the chapter in section 5.6.

5.1 Development of a characteristic equation for a pure linear integer program

5.1.1 Analysis of a trivial example

The concepts associated with a CE can be appreciated by considering a simple trivial numerical illustration. We are using a slightly modified LP problem in two dimensions taken from Hiller and Lieberman (2005) given below:

$$\text{Max } z = 7x_1 + 4x_2$$

Subject to

$$\begin{aligned} &x_1 + x_2 \le 6 \\ &9x_1 + 5x_2 \le 45, \\ &x_1, x_2 \ge 0 \text{ and integer} \end{aligned} \tag{5.1}$$

For this trivial example, one can easily do an exhaustive search of all integer points in the LP feasible space and locate the optimal and other $k^{th} best (k \geq 2)$ solutions. This is given in Table 5.1, and Figure 5.1.

Table 5.1: All integer points in the feasible space for problem 5.1.

Sl. no	Int. pt.	Z value	Sl. no	Int. pt.	Z value	Sl. No	Int. pt.	Z value
1	(0,0)	0	2	(1,0)	7	3	(2,0)	14
4	(3,0)	21	5	(4,0)	28	6	(5,0)	35*1
7	(0,1)	4	8	(1,1)	11	9	(2,1)	18
10	(3,1)	25	11	(4,1)	32*3	12	(0,2)	8
13	(1,2)	15	14	(2,2)	22	15	(3,2)	29
16	(0,3)	12	17	(1,3)	18	18	(2,3)	26
19	(3,3)	33*2	20	(0,4)	16	21	(1,4)	23
22	(2,4)	30*4	23	(0.5)	20	–	–	–

By inspection of the values in the 'Z value' column, one can easily identify the optimal and other $k^{th} best (k \geq 2)$ solutions. We have marked a couple of these solutions by an asterisk mark in Table 5.1.

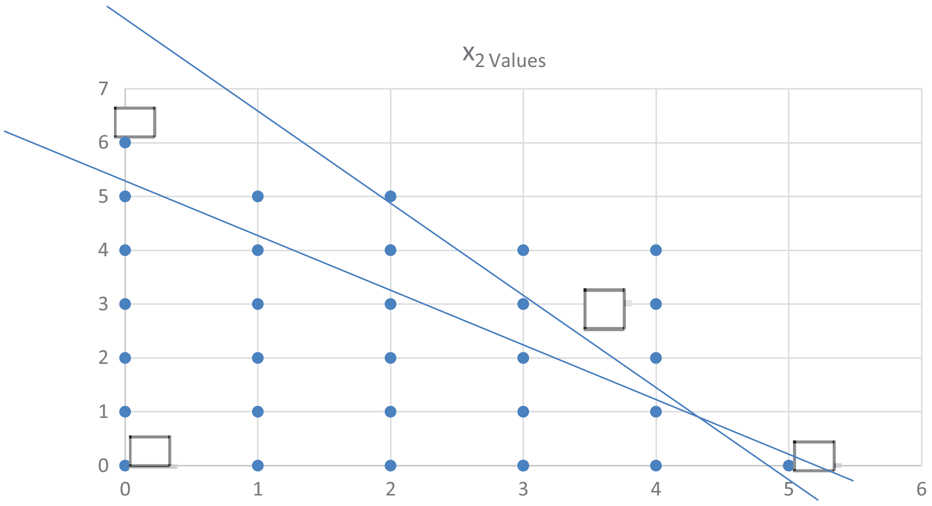


Figure 5.1: All feasible integer points for the problem (5.1).

From the Table 5.1, one can easily verify that the integer optimal solution is situated at the point (5,0) resulting in z value equal to 35. Similarly, we have also labelled a few more ordered solutions by an asterisk mark. Now we will develop a linear relation from the LP optimal solution and again identify all these solutions by using that relation.

The problem (5.1) is solved as a LP and the optimal solution is given in Table 5.2.

Table 5.2: LP optimal solution.

Z	x_1	x_2	S_1	S_2	RHS
1			0.25	0.75	35.25
		1	2.25	-.25	2.25
	1		-1.25	0.25	3.75

The LP solution from Table 5.2 is:

$$z = 35.25, x_1 = 3.75, x_2 = 2.25, S_1 = 0 \text{ and } S_2 = 0$$

The objective function z = 35.25 is an upper bound on the integer optimal solution. From Table 5.2, the objective function and the two basic variables x_1 and x_2 as a function of the non-basic variables are given by:

$$Z + (0.25)S_1 + (0.75)S_2 = 35.25$$

$$x_1 - 1.25s_1 + 0.25s_2 = 3.75$$

$$x_2 + 2.25s_1 - 0.25s_2 = 2.25 \tag{5.2}$$

The optimal z, x_1 and x_2 values under the condition that both non-basic variables S_1 and S_2 are equal to zero is given by 35.25, 3.75, and 2.25, respectively. Note that z = 35.25, x_1 and x_2 equal to 3.75 and 2.25 are not acceptable values when dealing with integer requirements. Under integer restrictions, all variables, basic or non-basic and the objective function will be integer restricted values.

Rewriting the objective function again with common denominator, we have:

$$z + \left(\frac{1}{4}\right)S1 + \left(\frac{3}{4}\right)S2 = (\frac{141}{4}) \text{ or}$$

$$\left(\frac{4}{4}\right)z + \left(\frac{1S1 + 3S2}{4}\right) = 35(\frac{4}{4}) + (\frac{1}{4})$$

The objective function will hold on to its integer value provided

$$S_1 + 3S_2 = 1 + 4i \tag{5.3}$$

Z can be an integer valued quantity only if the LHS of (5.3), i.e. $(S_1 + 3S_2)$ will be equal to 1 (when *i=0*) or 5 (when *i=1*) or 9 (when *i=2*) etc. Here 1 on the RHS of relation (5.3) is the remainder when 141 is divided by 4 and 4 is the common denominator. The LHS in a LP interpretation is zero but for the integer solution, we must increase values of non-basic variables to integer values in a controlled way i.e. the total sum of these values must be equal to 1 or 5 or 9 etc. For all other values, the objective function will continue to have fractional values. The role of $'i'$ on the RHS is to increase LHS in a controlled way for $i = 0, 1, 2, \ldots$.

For example, let us consider various possibilities from equation (5.3):

Iteration 1: For $'i = 0'$, $1S_1 + 3S_2$ =1 will have integer solution for S_1 and S_2 that can make the LHS =1. A feasible integer solution for non-basic variables is given by $S_1 = 1, S_2 = 0$, which will determine values of basic variables as below:

$$x_1 - \left(\frac{5}{4}\right)S_1 + \left(\frac{1}{4}\right)S_2 = \left(\frac{15}{4}\right) \text{ at } S_1 = 1 \text{ and } S_2 = 0, x_1 = 5$$

$$x_2 + \left(\frac{9}{4}\right)S_1 - \left(\frac{1}{4}\right)S_2 = \left(\frac{9}{4}\right) \text{ at } S_1 \text{ and } S_2 \text{ as above}, x_2 = 0.$$

This is the same point as was reflected as the optimal solution from Table 5.1.

For the next possibility consider:

Iteration 2: For $'i = 1'$, $1S_1 + 3S_2$ = 5 gives two alternative solutions: (1) S_1 =2 and S_2 =1 and (2) S_1 =5 and S_2 =0.

For the first solution, the values of S_1 and S_2 will alter the value of the basic variables which are also functions of non-basic variables. From the relation (5.2), we have

$$x_1 - \left(\frac{5}{4}\right)S_1 + \left(\frac{1}{4}\right)S_2 = \left(\frac{15}{4}\right) \text{ at } S_1 = 2 \text{ and } S_2 = 1, x_1 = 6$$

and

$$x_2 + \left(\frac{9}{4}\right)S_1 - \left(\frac{1}{4}\right)S_2 = \left(\frac{9}{4}\right) \text{ at } S_1 \text{ and } S_2 \text{ as above}, x_2 = -2$$

Hence the identified integer solution is given by: $S_1 = 2$ and $S_2 = 1$, resulting in $x_1 = 6$ and $x_2 = -2$, which is infeasible. Let us continue with the other solution.

For the second solution, we get:

$$x_1 - \left(\frac{5}{4}\right)S_1 + \left(\frac{1}{4}\right)S_2 = \left(\frac{15}{4}\right) \text{ at } S_1 = 5 \text{ and } S_2 = 0, x_1 = 10$$

$$x_2 + \left(\frac{9}{4}\right)S_1 - \left(\frac{1}{4}\right)S_2 = \left(\frac{9}{4}\right) \text{ at } S_1 \text{ and } S_2 \text{ as above}, x_2 = -9$$

Hence both solutions lead to infeasible solutions.

Iteration 3: For $'i=2'$, S_1+3S_2 = 9 gives more than one integer solutions again. These solutions are: (1) $S_1=0$ and $S_2=3$; (2) $S_1=3$ and $S_2=2$, (3) $S_1=6$ and $S_2=1$ or (4) $S_1=9$ and $S_2=0$. All these solutions are investigated separately.

The first solution $S_1=0$ and $S_2=3$ gives z = (141/4) – (9/4) + (1/4)0 = 33, now let us check values of the basic variables again.

$$x_1-\left(\frac{5}{4}\right)S_1+\left(\frac{1}{4}\right)S_2=\left(\frac{15}{4}\right)\text{ at }S_1=0\text{ and }S_2=3\text{ gives }x_1=3$$

and as above

$$x_2+\left(\frac{9}{4}\right)S_1-\left(\frac{1}{4}\right)S_2=\left(\frac{9}{4}\right)\text{ at }S_1\text{ and }S_2\text{ as above gives }x_2=3$$

Thus z = 33 at $x_1=$ x_2 =3 is the second-best solution, as was expected from the Table 5.1.

Similarly, other solutions need be investigated. Consider, for example the solution: $S_1=3$ and $S_2=2$.

This solution will give rise to $x_1=7,$ *and* $x_1=-4$ *a* infeasible solution.,

Similarly, other solutions will also lead to infeasible solutions.

For the 3rd best integer solution, we investigate for $'i=3'$, S_1+3S_2 = 13, one gets $S_1=1, S_2=4$, $x_1=$ 4 and x_2 =1, gives Z=32. This time again we can get a few more solutions, which will lead to infeasible answers.

If we keep increasing the RHS of (5.2), when *i*=35, z =0. The CE will be

$S_1+3S_2=141$ and Z = 0.

For increased values of *i*, the number of solutions will start increasing and computational load will increase.

The above discussion can be formulized in the form of a characteristic equation.

5.1.2 The characteristic equation

From the LP optimal solution, one gets

$$\left(\frac{D}{D}\right)Z+\frac{\beta_1 s_1+\beta_2 s_2+\ldots+\beta_k s_k}{D}=\frac{P}{D}=\frac{(R+iD)}{D} \tag{5.4}$$

where k represents the number of non-basic variables, $\beta_1,\beta_2,\ldots,\beta_k$ represent their integer coefficients, D is the lowest common factor and R is the remainder, also an integer value. From (5.4), one can establish a CE for that problem as given by equation (5.5):

$$\sum_{j=1}^{k}\beta_j s_j=R+iD, i=0,1,2,\ldots \tag{5.5}$$

Note that the RHS is a constant for a given value of *i*. The equation (5.5) is a moving hyperplane, where the objective function values are integer restricted values. If solution of the equation (5.5) gives integer non-basic values, it may also have integer values for basic variables. In other words, theses hyperplanes will pass through various integer points in the LP convex region.

5.1.3 Some interesting properties of the CE

Property 5.1.3.1
The characteristic equations for a multi-objective integer model will be more than one, as one CE will be obtained from each objective function. However, since solution space is the same defined by the given set of constraints; with respect to each objective these different CE's will pass through the same integer points in a different descending order. Therefore, we will have a choice of several CEs to identify these integer points. We can select the one with larger values of D and R or the one with minimum *i values*.

Property 5.1.3.2
Search for an integer point will be better if coefficients of non-basic variables are not equal to 0 or 1. Note that in the equation (5.3), the coefficient of $s_1 = 1$, which resulted in increased number of alternative solutions. In the case of a multi-objective model, if such cases can be avoided, it is computationally beneficial.

Property 5.1.3.3
The number of solutions starts to increase as *i* takes on larger values. This number further increases when some non-basic variables have 0 or 1 as their coefficient.

Property 5.1.3.4
The concept of CE will not be useful for models like transportation and assignment problems as these models do not deal with slack variables. However, if uni-modular problem can be approached in the conventional LP sense, the CE approach will be applicable. For more observations, see Kumar et al. (2020). Therefore, determination of the kth best of special transportation and assignment models create a challenge, that deserves more attention.

Property 5.1.3.5
The CE is a necessary but not a sufficient condition, for an integer solution. Note for developing a CE, we first restricted the objective function value to an integer value, then we restricted the non-basic variables to an integer value and use those values

to find the corresponding value of the basic variables. If these variables turn out to be a feasible integer values, we stop, else we continue the search.

5.1.4 An algorithm to find the k^{th} best optimal solutions, k ≥ 1 using the CE approach

The above properties give rise to an algorithm to find integer ordered-optimal solutions for a given PIP using the CE.
Initial Step 0:

0.1 Assign a value to K (i.e. the number of ordered optimal solutions required) and set k=1 (i.e. find the optimal integer solution).

Step 1:

1.1 Solve the LP relaxation of the given LIP model.

1.2 Write down the CE from the objective function row from the optimal solution in the form of an equation (5.4) and solve for integer values of $s_1, s_2 \ldots.$ from the equation for $i = 0$,

$$\sum_{j=1}^{k} \beta_j s_j = R + iD$$

If we find an integer solution, check the values of the basic variables. If these variables are integer valued quantities, go to Step 4.2, else go to Step 2.

Step 2:

2.1 Set $i = i + 1$ and go to 2.2

2.2 Solve the CE for $s_1, s_2 \ldots$

$$\sum_{j=1}^{k} \beta_j s_j = R + iD$$

If it results in integer solution, go to Step 3, otherwise go to Step 2.1.

Step 3:

Substitute the integer solution $s_j, j = 1, 2, \ldots, k$ in the expressions of the basic variables as a function of the non-basic variables obtained from the LP optimal solution. If all basic variables are integer, go to Step 4, otherwise return to Step 2.1.

Step 4:

4.1 Optimal solution for kth best has been obtained and one can find the integer value of the objective function also.

4.2 If $k < K$, *set* $k = k + 1$ go to Step 1.2, else go to Step 4.3.
4.3 Stop as all K optimal solutions have been obtained.

5.1.5 Features of the CE

There are many interesting features associated with the CE, which are presented as follows:

5.1.5.1 The solution of the equation (5.4) for any given value of *i* is independent of the knowledge of the previous set of solutions. If a solution for a particular value of *i* is infeasible with respect to the integer requirement, a desirable feature of the CE is that one can move on to the next value of the *i*. One does not have to have complete calculation with respect to the infeasible solution. This is not the case with the branch and bound and the cutting plane methods. In these two methods, the next iteration can commence only when the previous iteration is completed, be it a feasible or infeasible solution. In other words, even after detecting a fractional value of a variable, one is required to complete that iteration to proceed further with the next iteration. The CE finds the integral solution in two stages and subsequent search is independent of the previous searches. Therefore, the CE (5.4) is recycled again until the required solution has been determined. Two possibilities arise for the solution of the CE, which is comprised of the non-basic variables only.

a. The solution of the CE satisfies the integral requirement of the non-basic variables. One moves to test basic variables for integer values for those integer values of the non-basic variables. If the solutions are integer, the search ends else we reconsider the next value of *i* in the equation (5.4).
b. If the non-basic variables do not satisfy integral requirement, we need not test the basic variables. One can move immediately to the next value of *i*.

5.1.5.2 Parallel computing is a desirable feature for any method, particularly when larger problems are to be solved. The CE is suitable for parallel computing.

5.1.5.3 Approaches like branch and bound and cutting plane result in an increase of the size of the problem with each iteration. This is not the case with the CE.

5.1.5.4 The method relies on the property that the LP solution is unique. The application of CE can be bit more demanding when the coefficient of a non-basic variable in the LP final tableau has a zero or one in the objective function row. In some cases, it happens due to multiple solutions, and if this is not the case, it may be desirable to subject the objective function to a small perturbation to avoid an alternative LP solution or having a coefficient 1 in the final simplex tableau. We have given some numerical illustrations for these situations. This aspect has been discussed in more details by Kumar et al. (2020).

5.1.5.5 Generally it is claimed that the branch and bound works better on small problems, we have pointed out later that the branch and bound may have difficulties even on small size problems. See the problem discussed in section 5.1.7.

5.1.5.6 Since the CE is used again and again, rounding errors do not play any role in the CE approach.

5.1.5.7 Determination of the kth best optimal solution is a very difficult problem in any discrete optimization. The difficulty arises as there is no direct measure to test the kth best optimality of a solution. However, the presence of *i* in the CE (5.4) provides this feature as an indirect bonus and makes the kth best a very special feature of the CE approach that other methods do not have.

5.1.5.8 The procedure can be highly efficient if the coefficients of non-basic variables are factors of the RHS value of the equation (5.4).

5.1.5.9 For larger the value of the RHS, the complexity of possible solutions increases. Therefore, the CE approach is likely to work efficiently if required ordered-optimal integer solutions are in a close vicinity of the LP optimal solution.

5.1.6 A numerical illustration

Consider, once again a simple problem given by (5.6)

Maximize

$$Z = 7x_1 + 9x_2$$

Subject to

$$-x_1 + 3x_2 \le 6, 7x_1 + x_2 \le 35, x_1, x_2 \ge 0 \text{ and integers.} \tag{5.6}$$

The LP optimal solution is given in Table 5.3.

Table 5.3: LP optimal solution.

Basic	x_1	x_2	S_1	S_2	RHS
Z	0	0	28/11	15/11	63
x_2	0	1	7/11	1/22	7/2=3.5
x_1	1	0	-1/22	3/22	9/2=4.5

For the CE, we have from Table 5.3 the relation (5.7)

$$Z + \left(\frac{28}{11}\right)S_1 + \left(\frac{15}{11}\right)S_2 = 63 \tag{5.7}$$

Which give rise to the CE as shown in (5.8)

$$28S_1 + 15S_2 = 11i \tag{5.8}$$

Note R = 0 as the RHS was free of fractions and D = 11. From (5.8), it is easy to see that for a feasible solution, the RHS must be $\geq \min(28, 15)$ and multiple of 15 or 28 or some combination. Thus for $i = 0, 1$ *and* 2, there is no solution to (5.8).

For *i* =3, we have $28S_1 + 15S_2 = 33$, which also does not give any integer solution. Similarly, for *i* = 4, 5, 6 and 7 there are no feasible integer solutions.

When *i* = 8, we have $28S_1 + 15S_2 = 88$

Results into a solution $S_1 = 1$ *and* $S_2 = 4$. Since we have an integer solution for non-basic, we now test these values for the basic variables, given below from Table 5.3

$$x_2 + \left(\frac{7}{22}\right)S_1 + \left(\frac{1}{22}\right)S_2 = \left(\frac{7}{2}\right), \text{which at } S_1 = 1 \text{ and } S_2 = 4 \text{ gives } x_2 = 3.$$

Similarly

$$x_1 - \left(\frac{1}{22}\right)S_1 + \left(\frac{3}{22}\right)S_2 = \left(\frac{9}{2}\right), \text{which at } S_1 = 1 \text{ and } S_2 = 4 \text{ gives } x_1 = 4.$$

and

$$Z = 63 - \left(\frac{28}{11}\right)S_1 - \left(\frac{15}{11}\right)S_2, \text{ which at } S_1 = 1 \text{ and } S_2 = 4 \text{ gives } Z = 55$$

This concludes into an optimal integer solution. If we want to find more solutions, we continue the search for higher values of *i*. For example, let us find the 2^{nd} best integer solution. The CE for the next value of *i* = 9, will be given by:

$28S_1 + 15S_2 = 99$, for which the solution will be $S_1 = 3$ and $S_2 = 1$. For these integer values of the non-basic variables, we get fractional values of basic variables and the objective function. Hence it does not give us an acceptable feasible solution for the integer model. Next integer solution for non-basic variables is achieved when *i=15,* giving the RHS =165. A feasible integer solution is when $S_1 = 0$ and $S_2 = 11$. The value of the basic variables is obtained as detailed below.

$$x_2 + \left(\frac{7}{22}\right)S_1 + \left(\frac{1}{22}\right)S_2 = \left(\frac{7}{2}\right), \text{ which at } S_1 = 0 \text{ and } S_2 = 11 \text{ gives } x_2 = 3.$$

Similarly

$$x_1 - \left(\frac{1}{22}\right)S_1 + \left(\frac{3}{22}\right)S_2 = \left(\frac{9}{2}\right), \text{ which at } S_1 = 0 \text{ and } S_2 = 11 \text{ gives } x_1 = 3.$$

and

$$Z = 63 - \left(\frac{28}{11}\right)S_1 - \left(\frac{15}{11}\right)S_2, \text{ which at } S_1 = 0 \text{ and } S_2 = 11 \text{ gives } Z = 48$$

Similarly, more ordered optimal solutions can be investigated. See Figure 5.2.

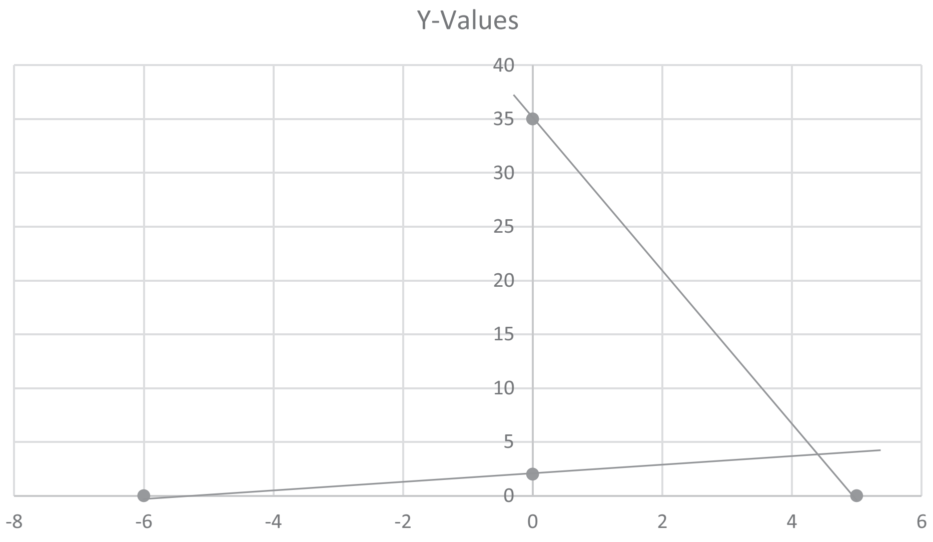


Figure 5.2: Graphical presentation of the problem (5.6).

5.1.7 An ill conditioned integer programming problem

Consider a 10-variable, 10-constraint problem, which when solved by the Branch and Bound algorithm took 1799 iterations for attaining the optimal solution. This problem was solved using TORA software. The same problem was attempted by the CE approach, it took 123 *i*-iterations. The problem is:

$$\text{Max } z = CX, \text{ subject to AX} \le b, X \ge 0 \text{ and integers,} \tag{5.9}$$

where

$$C = [5 \quad 90 \quad 12 \; 27 \quad 56 \quad 56 \; 23 \quad 36 \quad 8 \quad 178]$$

$$A = [a_{ij}] = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 56 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -5 & 1 & 1 & 1 & 16 & 20 & 1 & 1 \\ 5 & 1 & 1 & -8 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 19 & 1 & 1 & -1 & 3 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & -12 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 90 & 8 & -1 & 1 & 1 & 1 \\ 1 & 1 & 34 & 5 & 1 & 1 & 1 & -9 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 81 & 1 & 1 & -1 & 25 \\ 23 & 1 & 1 & 1 & 1 & 5 & 1 & 1 & 1 & -10 \end{pmatrix}$$

$$b = \begin{pmatrix} 100 \\ 5679 \\ 1990 \\ 450 \\ 670 \\ 80 \\ 8887 \\ 68 \\ 350 \\ 523 \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{pmatrix} \geq [0] \text{ and integer.}$$

For more ill condition pure integer programs, see Kumar *et al.* (2007).

5.1.8 Analogies of the characteristic equation with other systems and models

Here are a few more features of the CE.

5.1.8.1 Mapping of LP optimal solution on integer points

The CE is a kind of linear mapping of the LP optimal solution on to an integer point in a descending order. It is left for further attention from the readers.

5.1.8.2 A CE may be viewed as a modified knapsack problem

A multi-item knapsack problem deals with maximizing the value of collected items within the capacity of the given knapsack. However, the CE from equation (5.5) represents a knapsack with changing capacity, i.e. $(R, R+D, R+2D, \ldots)$, hence we call it a protean knapsack, see Kumar (1994) and the other difference is that we want the knapsack is full to its capacity. Since the same equation is solved again and again with different values in the RHS, information recycling by Kumar (2005, 2006) may be useful here.

5.1.8.3 Integer solution search strategies

The CE is comprised of LP non-basic variables and integer coefficients and integer RHS value. If these integer coefficients are arranged either in increasing or decreasing order, one can minimize the efforts required for searching integer solutions. One such method – 'The ordered tree method', developed by Munapo et al. (2009) is discussed here in Section 5.2.

5.2 The ordered tree method for an integer solution of a given CE by Munapo et al. (2009)

The ordered tree method was developed to solve a knapsack problem, however, since there is similarity between the knapsack and the CE, the method can be modified to solve the CE also. To understand the method, some terms are necessary. They are defined, and when required illustrated below:

1. **An ordered branch and ordered tree**: The non-negative linear expression in CE may be used to construct an ordered branch and subsequently an ordered tree. An **ordered branch** is set of emanating links from the same node and having the coefficients of the non-basic variables as link lengths. The branch is ordered because link lengths are arranged in ascending order. The **ordered tree** is developed by adding ordered branches to the smallest node of the tree. An ordered branch is shown in Figure 5.3, where $C_1 < C_2 < \ldots < C_n$.

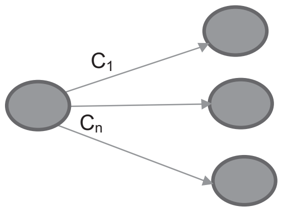


Figure 5.3: An ordered branch of an ordered tree.

2. **Node value:** Each node is assigned a value that is a sum of the links from the root note of the tree to that node. The value of the root node is assumed zero. In Figure 5.4, if an ordered branch is added to a node, the value of the nodes will be as shown in Figure 5.4.
3. **Smallest node:** is a node with smallest node value.
4. **Node merging:** is a process of joining nodes with the same value by a dummy link, denoted by a dotted line with link length zero. This node merging process reduces tree expansion and it indicates existence of multiple solutions to the

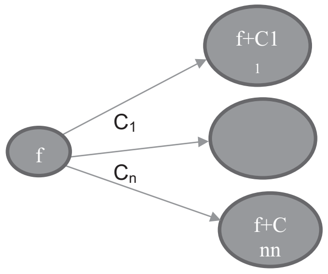


Figure 5.4: Value of the nodes.

CE. The concept of node merging can be explained by a simple illustration. For example, consider the following CE.

$$2s_1 + 3s_2 + 4s_3 = 2 + 2i, i = 0, 1, 2 \ldots \tag{5.10}$$

For $i = 0,$ the equation (5.10) will become $2s_1 + 3s_2 + 4s_3 = 2$. Since the coefficients in equation (5.10) are in an increasing order, no changes are required. The tree, at the initial stage, will have 3 leaves i.e. three links of lengths 2, 3 and 4. Let us investigate it for three stages. Two stages leaves are shown in Figure 5.5.

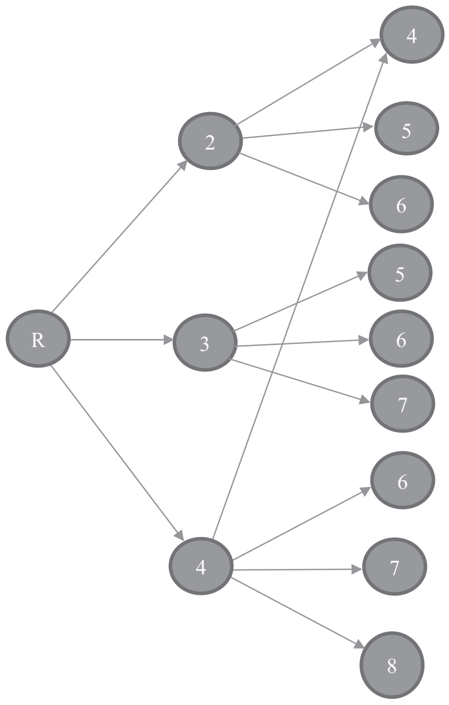


Figure 5.5: The ordered tree for $i = 0, 1, 2$.

Since the nodes with same value do not give rise to more information, the concept of node merger helps in reducing the number of leaves without any loss of information. The solution of the CE (5.10) for $i = 0$ is $s_1 = 1$ *and for* $i = 1$, the CE has two solutions, i.e. $s_1 = 2, s_2 = s_3 = 0$ and $s_1 = s_2 = 0, s_3 = 1$.

Similarly, more node mergers will take place if we proceed with the next stage.

5. **Fathomed node:** A node in an ordered tree is fathomed, when the no further ordered branch are required to be added on to it. A node is fathomed when the node value is greater than the RHS value. For example, for $i = 0$, the RHS of the CE (5.10) was 2 and the node value of one leaf was equal to 2 and other nodes were greater than 2. Hence one solution for $i = 0$ was concluded. However, for $i = 1,$ the RHS became 4 and we obtained two solutions, one from the node with value equal to the RHS and the other from the merger. All other nodes were fathomed.
6. **Fathomed tree:** An ordered tree is fathomed when the smallest node is fathomed.
7. **An integer solution to the CE:** An integer solution to the CE is obtained when node value is exactly equal to the RHS. The solution is easily traced by the path from the root to that node.
8. **Level of search:** A tree is fathomed at a given level, when no ordered branch can be added to it and no node has the value equal to the RHS. Alternatively, if the node value is equal to the RHS, an integer solution for non-basic variables has been obtained and then one must check for an integer basic solution.

5.2.1 A Numerical illustration of the ordered tree search technique

Let us reconsider the problem (5.6) for which the LP relaxation solution is given in Table 5.3 and the CE was given by equation (5.8). This equation and relations between the basic and non-basic variables are reproduced below as (5.11).

The CE was $28S_1 + 15S_2 = 11i$ and the

$$x_2 + \left(\frac{7}{22}\right)S_1 + \left(\frac{1}{22}\right)S_2 = \left(\frac{7}{2}\right)$$

$$x_1 - \left(\frac{1}{22}\right)S_1 + \left(\frac{3}{22}\right)S_2 = \left(\frac{9}{2}\right). \tag{5.11}$$

For developing an ordered tree, we rewrite the CE as:

$$15S_2 + 28S_1 = 11i$$

The ordered branch for (5.11) will be given by Figure 5.6

The tree in Figure 5.6 is fathomed for $i = 0$ *and* 1. It will start to grow for i = 0, 1, 2, $3, 4, 5, 6, 7$ *and* 8. Tree for i=8 will be complex when the CE has a solution. This solution is given by the slack variable values as $s_1 = 1$ *and* $s_2 = 4$. These values

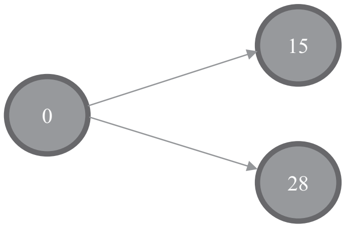


Figure 5.6: The ordered branch for the CE (5.11).

when substituted in (5.11) give $x_1 = 4, x_2 = 3$ *and* $Z = 55$. Once again, it has resulted in the same answer, as was expected.

It may be noted that combinatorial explosion is a serious issue with the approach. The node merger idea does help to reduce the bulk of combinations. Solution of a CE is an interesting problem that requires more attention.

5.3 The CE for the binary integer program

Steps for the CE approach for the binary integer problem are similar to the earlier discussion with a few specific differences.

1. The binary variables are restricted to 0 or 1 value. However, for its LP relaxation these variables are replaced by $0 \le x_j \le 1$, which can be handled by using the LP upper bounded variable technique.
2. The slack variables with respect to the binary variables are also binary i.e. they restricted to 0 or 1 value. However, the slack variables arising from each given constraint is not restricted to a binary value.

5.3.1 Numerical illustration of a binary program

$$Max\ Z = 3x_1 + 4x_2 - 2x_3 - x_4 + 2x_5$$

Subject to:

$$2x_1 - x_2 + x_3 + x_4 + 3x_5 \le 4$$

$$-x_1 + 3x_2 + 4x_3 - x_4 + 2x_5 \ge 5$$

$$2x_1 + 2x_2 - x_3 + 6x_4 + 2x_5 \le 8$$

$$x_j = 0 \vee 1, \forall j \text{ and } j = 1, 2, \ldots, 5. \tag{5.11}$$

Using the upper bound technique, the continuous optimal solutions is given in Table 5.4, where variable $\overline{x_j} = 1 - x_j, j = 2$ *and* 5.

From the Table 5.5, the solution to the model (5.11) is given by:

$x_1 = 8/9,\ \overline{x_2} = 0 \equiv x_2 = 1, x_3 = \frac{2}{9},\ x_4 = 0, \overline{x_5} = 0 \equiv x_5 = 1,\ s_1 = 0,\ s_2 = 0, s_3 = 22/9.$

Table 5.4: LP Optimal solution using the upper bound technique.

x_1	$\overline{x_2}$	x_3	x_4	$\overline{x_5}$	s_1	s_2	s_3	RSH
0	$\frac{67}{9}$	0	$\frac{26}{9}$	$\frac{2}{9}$	$\frac{10}{9}$	$\frac{7}{9}$	0	$\frac{74}{9}$
0	$-\frac{5}{9}$	1	$-\frac{1}{9}$	$-\frac{7}{9}$	$\frac{1}{9}$	$-\frac{2}{9}$	0	$\frac{2}{9}$
1	$\frac{7}{9}$	0	$\frac{5}{9}$	$-\frac{10}{9}$	$\frac{4}{9}$	$\frac{1}{9}$	0	$\frac{8}{9}$
0	$-\frac{37}{9}$	0	$\frac{43}{9}$	$-\frac{5}{9}$	$-\frac{7}{9}$	$\frac{4}{9}$	1	$\frac{22}{9}$

Writing the basic as a function of non-basic and the objective function row gives:

$$x_3 = \frac{2 - \left[-5\overline{x_2} - x_4 - 7\overline{x_5} + s_1 - 2s_2\right]}{9}$$

$$x_1 = \frac{8 - \left[-7\overline{x_2} - 5x_4 - 10\overline{x_5} + 4s_1 - s_2\right]}{9}$$

$$s_3 = \frac{22 - \left[-37\overline{x_2} + 43x_4 - 5\overline{x_5} - 7s_1 - 4s_2\right]}{9}$$

$$Z + \frac{\left[67\overline{x_2} + 26x_4 + 2\overline{x_5} + 10s_1 + 7s_2\right]}{9} = \frac{74}{9} \tag{5.12}$$

Resulting in the CE given by:

$$67\overline{x_2} + 26x_4 + 2\overline{x_5} + 10s_1 + 7s_2 = 2 + 9i \tag{5.13}$$

Where $\boldsymbol{i} = 0, 1, 2 \ldots$

Solution of the CE (5.13)

Iteration 1: Put i=0, the RHS becomes 2. The only possible solution will be $\overline{x_5}$=1. This value results in values of basic variables as follows: $x_3,\ x_3 = 1, x_1 = 2$ *and* $s_3 = 3$. Since variables were binary restricted, the above solution is binary infeasible.

Iteration 2: Put i=1, the CE becomes

$67\overline{x_2} + 26x_4 + 2\overline{x_5} + 10s_1 + 7s_2 = 11,$ and therefore the CE for integer solution can be simplified to $2\overline{x_5} + 10s_1 + 7s_2 = 11$, and has a solution $\overline{x_5} = 2,$ and $s_2 = 1,$ which is again binary infeasible.

Iteration 3: Put i=2, the CE becomes $67\overline{x_2} + 26x_4 + 2\overline{x_5} + 10s_1 + 7s_2 = 20$, which on simplification becomes $2\overline{x_5} + 10s_1 + 7s_2 = 20$. Many binary infeasible solutions are possible, for example: (1) $\overline{x_5} = 10,\ \overline{x_5} = 3$ and $s_2 = 2$. However, the solution $s_1 = 2$ is a feasible solution as s_1 is not a binary restricted variable. From the relations (5.11), one can easily verify that $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 0$, $x_5 = 1$, $s_1 = 2$, $s_2 = 0$, *and* $s_3 = 4$; *giving* $Z = 6$

The solution can also be traced by developing a binary tree as discussed by Kumar, Munapo and Jones (2009). We develop this tree for the CE when i=2. We know that $\overline{x_2}=0, x_4=0,$ *and equation reduces to* $2\overline{x_5}+10s_1+7s_2=20$. The binary tree is shown in Figure 5.7.

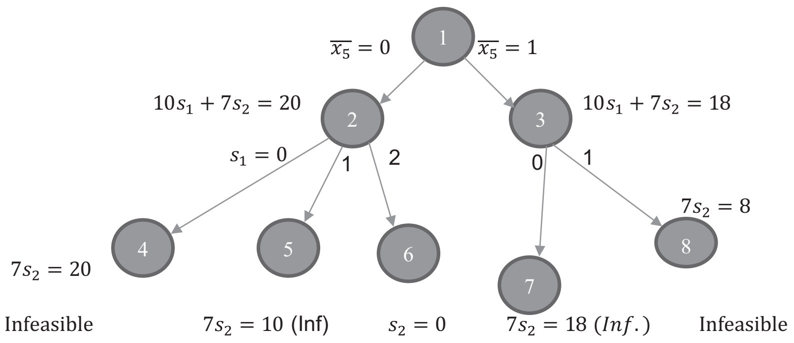


Figure 5.7: Binary tree.

Node 6 leads to a feasible solution with $s_1=2,\ s_2=0,$ and all other values will be the same as were obtained earlier.

It may be noted that the tree will get larger and larger as the value of the RHS increases for higher values of $i \geq 3$.

5.4 CE applied to a bi-objective integer programming model

Multi-objective optimization models are a natural modelling tool for analyzing many practical situations arising in business and industry. Accordingly, the field of multi-objective optimization has demanded researchers' attention, for example, see Antunes et al. (2016), Ehrgott and Gandibleux (2000), Ehrgott (2006). The central problem in bi-objective and multi-objective cases is to find the non-dominated point set unlike the optimal solution for single-objective models. Al-Rabeeah, et al. (2019) used the CE approach to find the set of non-dominated point set for a given bi-objective model, which is presented here.

5.4.1 Numerical illustration for bi-objective model

Consider that the bi-objective model is given by:

$$Max\ Z_1=3x_1+2x_2$$

$$Max\ Z_2=x_1+3x_2$$

$$S.t.\ 2x_1 + 3x_2 \leq 11$$

$$x_1 \leq 4,\ x_2 \leq 3,\ x_1,\ x_2 \geq 0,\ x_1,\ x_2 \in Z \tag{5.14}$$

For developing the CEs for the above model (5.14), we consider the given two objectives Z_1 and Z_2.

For these two objectives the LP final simplex tables are given below as Tables (5.5) and (5.6), respectively.

Table 5.5: Final simplex table for the objective Z_1.

	x_1	x_2	s_1	s_2	s_3	rhs
x_2	0	1	$\frac{1}{3}$	$-\frac{2}{3}$	0	1
x_1	1	0	0	1	0	4
s_3	0	0	$-\frac{1}{3}$	$\frac{2}{3}$	1	2
$z_j - c_j$	0	0	$\frac{2}{3}$	$\frac{5}{3}$	0	14

The CE with respect to the objective Z_1 from the above table is given by:

$2s_1 + 5s_2 = 3i, i = 0, 1, 2, \ldots 14$ and relations between basic and non-basic variables are given by:

$$x_2 + \left(\frac{1}{3}\right)s_1 - \left(\frac{2}{3}\right)s_2 = 1$$

$$x_1 + (0)s_1 + s_2 = 4$$

$$s_3 - \left(\frac{1}{3}\right)s_1 + \left(\frac{2}{3}\right)s_2 = 2 \tag{5.15}$$

For $i = 0$, $s_1 = s_2 = 0$ and we obtain $x_1 = 4, x_2 = 1$ and $s_3 = 2$ giving the optimal solution as was expected as LP solution was free of fractions.

For $i = 1$, there is no solution. For $i = 2$, $s_1 = 3, s_2 = 0$, $s_3 = 2, x_1 = 4, x_2 = 0$, giving $z_1 = 12$ is the second best solution.

Similarly for $i = 3$, we have $s_1 = 2, s_2 = 1, s_3 = 2, x_1 = 3, x_2 = 1, z_1 = 11$. For $i = 4$, we have $s_1 = 1$, $s_2 = 2$, $s_3 = 2$, $x_1 = 2$, $x_2 = 2$, $z_1 = 10$, *and* for $i = 5$, we have $s_1 = 0$, $s_2 = 3$, $s_3 = 0$, $x_1 = 1$, $x_2 = 3$, $z_1 = 9$.

Similarly, the final simplex tables and CE's for the objectives Z_2 is given below.

Once again, the CE and relations in basic and non-basic will be as follows:

$$s_1 + 3s_3 = 2i, i = 0, 1, \ldots, 10$$

$$x_1 + (1/2)s_1 - (3/2)s_3 = 1$$

Table 5.6: Final simplex table for the objective Z_2.

	x_1	x_2	s_1	s_2	s_3	rhs
x_1	1	0	$\frac{1}{2}$	0	$-\frac{3}{2}$	1
s_2	0	0	$-\frac{1}{2}$	1	$\frac{3}{2}$	3
x_2	0	1	0	0	1	3
$z_j - c_j$	0	0	$\frac{1}{2}$	0	$\frac{3}{2}$	10

$$\begin{aligned} s_2 - (1/2)s_1 + (3/2)s_3 &= 3 \\ x_2 + (0)s_1 + s_3 &= 3 \end{aligned} \tag{5.16}$$

Once again, for $i = 0$, the LP solution will be the optimal solution as it is free of fractions. For $i = 1$, we have $s_1 = 2, s_2 = 4, s_3 = 0, x_1 = 0, x_2 = 3, z_2 = 9$.

For $i = 2$, we have two possible solutions. One of them results into an infeasible solution and the other one gives: $s_1 = 1, s_2 = 2, s_3 = 1, x_1 = 2, x_2 = 2, z_2 = 8$. For $i = 3$, we have three solutions. Two of them lead to infeasible solutions and the remaining one gives, $s_1 = 0, s_2 = 0, s_3 = 2, x_1 = 4, x_2 = 1, z_2 = 7$.

These integer ordered-optimal solutions identified with respect to the objectives z_1 can also be traced from z_2 and vice a versa. For example, the optimal solution with respect to the objective z_1 is the same as the point with respect to $z_2 = 7$. Therefore, in multi-objective situations, no need to scan with respect to each objective. An integer point in the convex space is independent of the objective. Once we have values of the objective functions, one can easily check each point for membership of the non-dominated point set, see Kumar et al. (2020).

5.5 Characteristics equation for mixed integer program

5.5.1 Mathematical developments

A mathematical model of a general mixed integer program is given by (5.17):

$$Max\ z = CX$$

Subject to $$AX \le b, X \ge 0, X_1 \in Z \tag{5.17}$$

Here $C = (c_1, c_{2,} \ldots, c_n), X^T = (x_1, x_2, \ldots, x_n) = [X_1, X_2]$

$$A_{ij} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, i = 1, 2, \dots, m; j = 1, 2, \dots, n, b^T = (b_1, b_2, \dots, b_m)$$

The given n variables form two kinds of variables, $n_1 < n$ are integer restricted non-negative variables and the remaining $(n - n_1)$ are non-negative real variables. In the model (5.17), the n_1 variables are represented by the vector X_1 and $(n - n_1)$ variables by a vector X_2.

From the mixed-integer model (5.17), one can generate two more models represented by (5.18) and (5.19), which forms a linear and a pure integer program, respectively.

The LP model is given by:

$$Max\ z_{LP} = CX$$

Subject to $$AX \le b, X \ge 0. \tag{5.18}$$

The pure integer program is given by:

$$Max\ z_{PIP} = CX_1$$

Subject to $$AX_1 \le b, X_1 \ge 0 \text{ and integer}. \tag{5.19}$$

Once a PIP solution to (5.19) is obtained, a new linear program can be developed by replacing the n_1 integer-restricted variables by their values from the PIP solution and the new LP becomes a problem of $(n - n_1)$ variables. These variables are restricted non-negative real variables. This new LP can be solved for these $(n - n_1)$ variables.

The above three models (5.17), (5.18) and (5.19), and the modified LP have strong interdependence properties, which are useful for establishing a solution of the given mixed integer program. These properties are presented below.

An upper bound relation

The LP optimal solution to (5.18) will be an upper bound to the given MIP model (5.17).

Feasibility and a lower bound relation

Since an optimal solution to the PIP model (5.19) will also satisfy all constraints of the MIP model (5.17), this optimal value will be a lower bound for the MIP.

Resource distribution relation
The MIP model deals with an optimal distribution of the given resources represented by the vector b to integer restricted n_1 variables and the remaining resources to the remaining $(n - n_1)$ real variables. The distribution of the resource vector b to integer restricted variables can be obtained by using the characteristic equation and the distribution of the remaining resources to continuous variables is obtained by solving the LP model.

Nature of an infeasible solution point
Let the optimal solution to the pure integer program be denoted by: $\{x_1 = \beta_{1,\dots,} x_j = \beta_j, \dots, x_n = \beta_n\}$. This optimal solution has a property that any solution point denoted by: $\left\{x_1 = \beta_{1,\dots,} x_j = \beta_j + 1, \dots, x_n = \beta_n\right\}$ for any j will always lead to an infeasible solution point.

Proof: Assume that the LP solution involves fractional values. The search for an optimal solution by using the CE approach is such that the part of the convex region from the optimal z_{LP} to the optimal z_{PIP} is comprised of only fractional values. Hence the next integer points with respect to any j i.e. for $x_j = \beta_j + 1$ must lie in the infeasible region.

5.5.2 A characteristic equation approach to solve a mixed-integer program

Using the characteristic equation, one can solve a mixed-integer model also. It involves the following:

Step 1: Solve the LP model (5.18). Its optimal value will be an upper bound for the mixed integer program (5.17). It is denoted by: Z_{LP}^{Ub}.

Step 2: Develop a CE from the final simplex tableau of the LP solution and find the PIP optimal solution. This PIP solution will be a lower bound for the given MIP. It is denoted by Z_{PIP}^{LB}.

Step 3: Substitute the integer values for the integer-restricted variables obtained from Step 2 into the original MIP model (5.17) and develop a modified LP in $(n{-}n_1)$ variables.Solve the LP in $(n{-}n_1)$ real variables. Denote the component of the objective function based on n_1 integer restricted values by z_{In1} and the optimal solution of the modified LP by $z_{LP(n-n1)}$.
The new lower bound for the given MIP (5.17) will be given by: $z_{In1} + z_{LP(n-n1)}$.

Step 4: If $\left(Z_{LP}^{UB} - Z_{PIP}^{LB}\right) or \left(Z_{LP}^{UB} - \ (z_{In1} + z_{LP(n-n1)})\right) = \ D_{LP-MIP}$ is equal to zero, or approximately equal to zero stop, the two part solution will constitute an optimal solution to the mixed-integer program. Else repeat these calculations with the next best PIP solution. If the difference D_{LP-MIP} starts to increase, stop further calculations and conclude the best solution as an optimal solution to the MIP.

5.5.3 Numerical illustration – MIP 1

Consider a slight modified version of an example taken from Hillier and Lieberman (2001).

$$\text{Max } 5x_1 - 2x_2 + 8x_3 - 2x_4$$

subject to:

$$\begin{gathered} x_1 + 5x_3 \le 10 \\ x_1 + x_2 - x_3 \le 1 \\ 6x_1 - 5x_2 \le 0 \\ -x_1 + 2x_3 - 2x_4 \le 3 \\ x_1, x_2, x_3 \ge 0 \text{ and } \in Z0 \\ x_4 \ge 0 \end{gathered} \tag{5.20}$$

The LP solution is given below, and the final simplex tableau is shown in Table 5.7.

$$x_1 = \frac{5}{4}, x_2 = \frac{3}{2}, x_3 = \frac{7}{4}, s_4 = \frac{5}{4}, x_4 = s_1 = s_2 = s_3 = 0, Z_{LP} = \frac{69}{4}$$

Here $s_1, s_2, s_3, and\, s_4$ are the slack variables in the 4 constraints in (5.20)

Table 5.7: Final Table of the simplex iterations for (5.20).

Basic	x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	RHS
x_2	0	1	0	0	1/10	1/2	-1/10	0	3/2
s_4	0	0	0	-2	-17/60	7/12	7/60	1	3/4
x_1	1	0	0	0	1/12	5/12	1/12	0	5/4
x_3	0	0	1	0	11/60	-1/12	-1/60	0	7/4
Obj	0	0	0	2	101/60	5/12	29/60	0	69/4

From the Table 5.7, the LP optimal value giving an upper bound = 69/4 = $\frac{69}{4} = 17 + \frac{1}{4}$ for the MIP model (5.20). We again reconsider the model (5.20), and solve it as a PIP, i.e. all variables as non-negative integer restricted variables. The CE from Table 5.7 will be given by (5.21):

$$120x_4 + 101s_1 + 25s_2 + 29s_3 = 15 + 60i, i = 0, 1, 2 \ldots \tag{5.21}$$

The non-basic variables x_4, s_1, s_2 and s_3 have to move from the current value zero to some non-negative integer value such that (5.21) remains satisfied for the minimum i, and also converts the current basic variables from the real values to integer values. From Table 5.7, the relations between the basic and non-basic variables are given by:

$$\begin{aligned} x_1 &= \frac{5}{4} - \frac{1}{12}s_1 - \frac{5}{12}s_2 - \frac{1}{12}s_3 \\ x_2 &= \frac{3}{2} - \frac{1}{10}s_1 - \frac{1}{2}s_2 + \frac{1}{10}s_3 \\ x_3 &= \frac{7}{4} - \frac{11}{60}s_1 + \frac{1}{12}s_2 + \frac{1}{60}s_3 \\ s_4 &= \frac{3}{4} + 2x_4 + \frac{17}{60}s_1 - \frac{7}{12}s_2 - \frac{7}{60}s_3 \end{aligned} \tag{5.22}$$

The CE (5.21) has no solution for $i = 0$. For $i = 1$, the CE (5.21) becomes:

$$120x_4 + 101s_1 + 25s_2 + 29s_3 = 75 \tag{5.23}$$

The equation (5.23) has a solution given by: $s_2 = 3$, which results in $x_1 = 0$, $x_2 = 0$, $x_3 = 2$, $x_4 = 0$, $s_1 = 0$, $s_2 = 3$, $s_3 = 0$, $s_4 = -1$, which is not an acceptable integer solution. We try the next value i.e., $i = 2$. For this value, the RHS of (5.23) becomes 135, which again has no solution. The next value is $i = 3$ which gives equation (5.24).

$$120x_4 + 101s_1 + 25s_2 + 29s_3 = 195 \tag{5.24}$$

Equation (5.24) has an integer solution, which is given by (5.25):

$$x_4 = 1,\ s_1 = 0,\ s_2 = 3,\ s_3 = 0 \tag{5.25}$$

At the solution given by (5.25), basic variables become: $x_1 = 0$, $x_2 = 0$, $x_3 = 2$, $s_4 = 1$, $z_{PIP} = 14$

Therefore, the optimal solution for the MIP is bounded by the values
$14 \le z_{MIP} \le 17.25$

From the PIP integer solution, a modified LP is developed by substituting values of the integer restricted variables, which are:$x_1,\ x_2,\ x_3 \ge 0$ and $\in Z$

Substituting the integer values for the integer restricted variables from the above solution $(x_1 = x_2 = 0,\ x_3 = 2)$ into MIP model, a modified LP as a function of remaining real variables, which is the variable x_4 becomes as given by (5.26).

$$Max\ z = 16 - 2x_4$$

Subject to

$$4 - 2x_4 \le 3, x_4 \ge 0 \tag{5.26}$$

Solving (5.26) trivially gives $x_4 = \frac{1}{2}$

Combining the integer solution and the above continuous solution, we have an improved solution to the given MIP model, which is given by:

$$x_1 = 0, x_2 = 0, x_3 = 2, s_4 = 1, z_{MIP} = 15 \tag{5.27}$$

The above solution can be concluded as an optimal solution to the given MIP. Note that the only variation possible is to increase the values of the variables *x1 or x3*, but the solution will exceed the current UB. Similarly, an increase in the values of the remaining variables will worsen the current solution. Hence an optimal solution has been obtained and the search is terminated.

MIP illustration 2

Consider the problem:

$$Max\ z = 3x_1 + 5x_2$$

Subject to:

$$\begin{gathered} x_1 + 3x_2 \geq 7 \\ x_1 - 3x_2 \leq 0 \\ x_1 + 2x_2 \leq 13 \\ 2x_1 + x_2 \leq 15 \\ 3x_1 + x_2 \geq 15, x_1 \geq 0\ and\ integer, x_2 \geq 0 \end{gathered} \tag{5.28}$$

The LP solution when $x_1,\ x_2 \geq 0$, The LP output table is as given in Table (5.8)

Table 5.8: Final simplex output for the LP relaxed model (5.28).

Basic	x_1	x_2	s_1	s_2	s_3	s_4	s_5	RHS
x_1	1	0	0	0	-1/3	2/3	0	17/3
s_2	0	0	0	1	7/3	-5/3	0	16/3
s_5	0	0	0	0	-1/3	5/3	1	17/3
x_2	0	1	0	0	2/3	-1/3	0	11/3
s_1	0	0	1	0	5/3	-1/3	0	29/3
$z_j - c_j$	0	0	0	0	7/3	1/3	0	106/3

The CE will be given by (5.29)

$$7s_3 + s_4 = 1 + 3i \tag{5.29}$$

The relations among basic and non-basic variables are given by (5.30)

$$\begin{aligned} x_1 &= \frac{17}{3} + \frac{1}{3}s_3 - \frac{2}{3}s_4 \\ s_2 &= \frac{16}{3} - \frac{7}{3}s_3 + \frac{5}{3}s_4 \\ s_5 &= \frac{17}{3} + \frac{1}{3}s_3 - \frac{5}{3}s_4 \\ x_2 &= \frac{11}{3} - \frac{2}{3}s_3 + \frac{1}{3}s_4 \end{aligned}$$

$$s_1 = \frac{29}{3} - \frac{5}{3}s_3 + \frac{1}{3}s_4 \tag{5.30}$$

The CE (5.29) has a solution for i = 0 gives

$$s_4 = 1,\ s_3 = 0, x_1 = 5, s_2 = 7, s_5 = 4, x_2 = 4, s_1 = 10, \mathit{and}\ 35 \tag{5.31}$$

For the given MIP (5.28), the upper bound is given by: $Z_{LP}^{UB} = 106/3 = 35 + 1/3$

Substituting $x_1 = 5$ in the model (5.28), gives a LP as a function of a single variable x_2 It is given below as model (5.32).

$$Max\ z = 15 + 5x_2$$

Subject to:

$$\begin{gathered} 3x_2 \geq 2 \\ -3x_2 \leq -5 \\ 2x_2 \leq 8 \\ x_2 \leq 5 \\ x_2 \geq 0,\ x_1 = 4 \end{gathered} \tag{5.32}$$

Which has a trivial solution given by: $x_2 = 4$

Hence MIP and PIP solutions are identical in this case.

5.6 Concluding remarks

In this chapter the characteristic equation approach has been discussed for solving integer, binary integer, multi-objective linear integer and mixed-integer programs. We find that:

- An integer point can be approached by many methods and techniques yielding either an exact or an approximate solution. Many solution methods have been developed in the literature, see Kumar et al. (2010).
- The complexity of an IP can increase with the size of the model (i.e. the number of variables and the number of constraints.)
- The special class of integer models like assignment, transportation, transshipment, vehicle routing and related problems play an important role in real life problems. There are many special methods to deal with these special problems. However, determination of the k^{th} *best for* $k \geq 2$ deserve further attention. One possible approach is based on random search has been discussed by Kumar et al. (2020b)
- An effective general-purpose method and proper software support for many recently developed approaches are not available.
- Efforts have been recently devoted to incorporation of randomness and fuzziness within the IP models.

For future developments in the field of IP, the following needs attention:
- Production of user-friendly software for these recently developed PIP solution approaches is desirable.
- Investigation of computational efficiency for these PIP approaches need be examined.
- Description of high-quality case studies to demonstrate the usefulness of more recent approaches will be desirable.

The information-recycling concept has been discussed by Kumar (2005, 2006) and applied to Linear and Geometric programming. Munapo *et al.* (2010) have applied information recycling to the integer polyhedron to solve protean PIP problems. Information recycling methods attempt to find a solution to a problem from available information of a similar but different problem, which was solved earlier by applying any known method. This concept may be useful for *i*-iterations for the CE approach.

References

Al-Rabeeah, M., Kumar, S., Al-Hasani, A., Munapo, E., and Eberhard, A., (2019). Bi-objective integer programming analysis based on the characteristic equation, International Journal of System Assurance Engineering and Management, DOI 10.1007/s13198-019-00824-7 (Online published 05 July 2019), Vol 10, No. 5, pp 937–944.
Antunes C.H., Alves M.J., and Climaco, J. (2016), Multiobjective linear and integer programming, Speringer, Berlin.
Erhgott, M., (2006), A discussion of scalarization techniques for multiple objective integer programming, Ann Oper Res, 147(1),343–360.
Ehrgott, M., and Gandibleux, X., (2000) A survey and annotated bibliography of multi-objective combinatorial optimization, OR Spektrum, Vol 22, No 4, pp 425–460.
Gomory, R.E., 1958. Outline of an algorithm for integer solutions to linear programs, Bulletin of the American Mathematical Society, 64 (5), pp 275–78.
Gomory, R.E., 1965. On the relation between integer and non-integer solutions to linear programs, Proc. Of the National Academy of Sciences, 53(2), pp 260–65.
Hillier, F. and Lieberman, S., 2004. Introduction to operations Research, McGraw-Hill, 8th Edition.
Kumar, S. (1994), Optimizations of protean systems: A review, APORS'94, Fukuoka, (Eds. Masanori Fushimi and Kaoru Tone, World Scientific Publishers, pp 139–46.
Kumar, S. (2005), Information recycling mathematical methods for protean systems: A path-way approach, South African Journal of Industrial Engineering, Vol 16, No 2, pp 81–101.
Kumar, S. (2006), Information recycling mathematical methods for protean systems: A path-way approach to a Geometric Program, South African Journal of Industrial Engineering, Vol 17, No 2, pp 127–143.
Kumar, S., Munapo, E. and Jones, B.C., 2007. An integer equation controlled descending path to a protean pure integer program, Indian Journal of Mathematics, Allahabad, 49(2), pp 211–237.
Kumar, S., Munapo, E., and Jones, B.C. (2009), Descending hyper-plane approach for a binary integer program, International Journal of Mathematical Modelling, Simulation and Applications, Vol. 2, No 1, pp 53–62.

Kumar, S. Luhandjula, M.K., Munapo, E. and Jones, B.C. 2010. Fifty years of integer programming: A review of solution approaches, Asia Pacific Business Review, 6(2), pp 5–15.

Kumar, S. and Munapo, E., 2012. Some lateral ideas and their applications for developing new solution procedures for a pure integer programming model, Keynote address, Proc. Of the Herbal International conference on Applications of Mathematics and Statistics -Intelligent solutions through Mathematics and Statistics, (Eds) Mariappan, Srinivasan and Amritraj, Excel India Publisher, pp 13–21.

Kumar, S., Munapo, E., Lesaoana, M. and Nyamugure, P., (2018) Some innovations in OR Methodology: Linear Optimization, Lambert Academic Publishing. ISBN: 978-613-7-38007-9

Kumar, S., Al-Hasani, A., Al-Rabeeah, M., Eberhard, A. (2020), Journal of Physics, Conference Series, ICoMPAC 2019, 1490(2020) 012063, doi:10.1088/1742-6596/1490/1/012063.

Kumar, S., Munapo, E. and Nyamugure, P. (2020), An insight into characteristic equation for an integer program, to appear in International J, of Mathematical, Engineering and Management'

Munapo, E., Jones, B.C., and Kumar, S., 2009, Ordered tree method for an integer solution of the characteristic equation arising in PIP, International Journal of Mathematical Modelling, Simulation and Applications, Vol. 1, No. 4, pp 24–37.

Winston, W.L., 2004. Operations Research: Applications and Algorithms, Duxbury.

Wolsey, L.A., 1980. Heuristic analysis, linear programming and branch and bound in Combinatorial optimization II, pp 121–134, Springer, Berlin, Heidelberg.

Chapter 6
Random search method for integer programming

Abstract: This chapter presents development of the random search method in context of the linear integer programming model, mixed-integer programming model and the extreme point mathematical programming model. The random search method is an approximation technique, where optimality of a solution is ascertained by a given probability, which is controlled by the investigator. Higher the probability, better results are achieved but computational load also increases. Each situation is illustrated by a numerical example.

Keywords: Random search method, Pure integer programming model, Mixed-integer programming model, Extreme point mathematical programming model

6.0 Introduction

Analytical solution procedures in integer programming have made considerable progress to identify the optimal integer or mixed-integer solutions to integer programming models arising in modern commercial and industrial situations, see [Hiller and Lieberman (2004), Kumar, et al. (2018), Nemhauser and Wolsey (1988, 89), Salkin (1971), Taha (2006, 2014) and Winston (2004)]. Many of these analytical solution methods have also been discussed elsewhere in this book. However, approximation methods have always played an important role in situations where mathematical conditions of optimality are either not easy to define or finding a solution that can satisfy the defined conditions is computationally very demanding. In such situations, approximation techniques are handy to develop approximate solutions within a reasonable time frame. The random search method for integer programming is one such technique, which is discussed in this chapter. Many different situations have been discussed in this chapter, where random search is meaningful for providing approximate answers in some realistic time.

The chapter has been organised in 6 sections. The random search method for a pure integer program has been discussed in Section 6.1, which was developed by Huynh, Kumar and Connell (1987). The random search method for a mixed-integer programming model is presented in Section 6.2, which was developed by Huynh, Connell and Kumar (1987). A computationally difficult type problem, known as an extreme point mathematical programming problem (EPMP) has been discussed in Section 6.3. After discussing the random search methods for integer and mixed-integer cases, general characteristics of the random search technique are presented in Section 6.4, and based on these characteristics, a random search method for an

https://doi.org/10.1515/9783110703023-006

EPMP problem has also been discussed in this section. This method was developed by Huynh and Kumar (1988). Finally, the chapter is concluded in Section 6.5.

6.1 The random search method for an integer programming model

In this section, a random search method for integer programming is discussed. This approach was developed by Huynh, Kumar and Connell (1987).

6.1.1 Integer linear program, notation, and definitions

Consider a linear integer programming model of the form:

$$Max\ Z = C.X$$

Subject to:

$$A.X \leq b, X \geq 0 \ and \ integer \tag{6.1}$$

Furthermore, X may have some defined lower and upper bounds given by $L \leq X \leq U$.

Note that A is a matrix m *by* n, b *is* m *by* 1, X, L *and* U *are* n *by* 1, *and* C *is* 1 *by* n.

Let,

$p =$ Probability that a search point under investigation is the required integer point.

If we are searching for an optimal integer point, then p is the probability that the randomly selected point is the required optimal integer solution. In other situations, the probability p and the solution may have some different interpretation in different situations. These different interpretations are seen in other sections of this chapter.

$ns =$ the required number of random searches for determination of the point under investigation. It may be an optimal solution or some other point of interest.

$Prob =$ The probability that at least one search point from the ns number of random search points is the required optimal integer point or some other point of interest.

$H =$ The set of integer points in the hyper-box determined by the bounds $(U - L)$, where U is the upper bound and L is the lower bound for each variable with origin as the point L. The integer points in the box are determined by the given constraints in (6.1).

$S =$ The set of feasible integer points defined by the given constraints

$A.X \leq b, X \geq 0$ *and integer*. Note that S *is a subset of* H $(S \subset H)$.

$Z_{max} =$ Is an upper bound on the maximum value of the objective function.

$x_{j\ max} =$ Is the upper bound on the value of the variable $x_j, j = 1, 2, \ldots, n$.

$x_{j\ min} =$ Is equal to the lower bound on the value of the variable $x_j, j = 1, 2, \ldots, n$. In the initial position, the origin, which is equal to 0 for all $x_{j,}$ is the lower bound.

$Z_{CP} =$ The value of the objective at the current feasible point.

Successful solution: It is a feasible solution (X, Z_{CP}) obtained after a previous feasible solution X', Z'_{CP}, which is said to be a successful solution if $Z_{CP} \geq Z'_{CP}$.

Bounds on the variable x_j can be obtained by solving two linear programming problems:

U_j for the variable x_j is determined by solving the LP, Max $x_j, ST\ AX \leq b, X \geq 0$ and the lower bound L_j for the same variable x_j is obtained by solving the LP, Max ($-x_j$), ST $AX \leq b, X \geq 0, j = 1, 2, \ldots, n$ *i.e.* this step is repeated for each variable x_j.

6.1.2 The random search method for integer programming

The random search method has some essential elements, which are discussed here with reference to the integer programming model:

First find the number of integer points in the set H. Let this number be represented by Product, where its value is given by (6.2)

$$Product = \prod_{j=1}^{n} \left(U_j + 1 - L_j\right) \tag{6.2}$$

Note $\left(U_j + 1 - L_j\right)$ is the number of integer points on the j^{th} side of the box represented by the set H, $j = 1, 2, \ldots, n$.

The set S is empty initially, as no solution is available.

Assume that the problem has only one optimal solution. The probability, p, that a search point is the required optimal solution is given by:

$$p = \frac{1}{Product} \tag{6.3}$$

The probability that the current search point is other than the required optimal integer point is given by: $(1 - p)$.

The probability of not obtaining an optimal solution in *ns* number of searches is = $(1 - p)^{ns}$.

Therefore, probability that at least one search point in the *ns* number of random searches is the required optimal solution is given by $(1 - (1 - p)^{ns})$.

We, therefore, can control the quality of our search by assigning a value to this probability and denote it by PROB. Therefore, we have:

PROB $= (1 - (1 - p)^{ns})$. This relation in PROB and *ns* can be used to find the value of the number of random searches for a given PROB and p. It is given by:

$$ns = \frac{\ln(1 - PROB)}{\ln(1 - p)} \tag{6.4}$$

Note that the probability 'Prob' and 'p' are known probabilities, hence from the relation (6.4), we can find the maximum number of random searches to find the required optimal integer solution for that given p and PROB. As we increase the value of ‘PROB’, the given assurance factor, the value of ‘*ns*’ will increase.

Note that we assumed that the optimal integer solution was unique. However, if the number of optimal solutions is more than one and assume it is given by k, $k \geq 1$. The probability will be denoted by:

$$p_k = \frac{k}{PRODUCT} > \frac{1}{PRODUCT} = p \rightarrow p_k > p$$

Since we have no knowledge of the number of optimal solutions to the given model, by using the *ns* number of searches under the condition of one optimal solution, we are likely to detect one of the required optimal points more rapidly, when number of optimal solutions are more than one.

6.1.3 Reduction in the region for search

After obtaining an initial solution X_0, let value of the objective at this point be given by Z_0. Suppose after obtaining the initial solution, if we get a solution, call it X'_0 *and* Z'_0 such that $Z'_0 > Z_0$. We call it a successful solution.

Note the given objective $Z = CX$ will increase in the direction of the normal to the objective function, which is defined by the vector C. The objective will increase only when it moves in the direction of the vector C. Thus after each successful solution, the search region is reduced, when $Z'_0 > Z_0$ is added to the given integer programming model.

6.1.4 The algorithm

Assume that X_0 is an initial feasible solution, the values of the objective Z_0 and PROB are known. Steps of the algorithm are as follows: all upper and lower bounds can be re-evaluated by using the ILP multi-objective model (6.5):

$$Max\{Min(x_1, \ldots, x_n); Max(x_1, \ldots, x_n)\} \equiv Max\{-X^T, X^T\}$$

Subject to:

$$A.X \leq b, C.X \geq Z_0 \; and \; X \geq 0. \tag{6.5}$$

This results in the following steps of the random search method.

Step 1: Using the relations (6.2), (6.3) and (6.4), find for the given problem lower and upper bounds, the value of the PRODUCT, *p and ns* and an initial solution X_0 and calculate $Z_0 = C.X_0$. Set the counter s:=1 and go to Step 2.

Step 2: Update# the value of p and go to Step 3.
Step 3: If $p = 1$, then Go to Step 7, otherwise calculate $ns = Int\left[\left[\frac{\ln(1 - PROB)}{\ln(1 - p)}\right]\right]$ and set s=0, s is a counter. Go to Step 4.
Step 4: Set s:= s+1.
If s $\geq ns$ then Go to Step 7.
Otherwise, go to Step 5.
Step 5: Calculate $Z = CX.If\ Z > Z_0$ and $Z \leq Z_{max}$, go to Step 6, otherwise go to Step 4.
Step 6: Check for feasibility of the solution, i.e. if $A.X \leq b$, then replace $Z_0 = Z$ *and* $X_0 = X$. Go to Step 2. Otherwise go to Step 4.
Step 7: Stop, the optimal solution is Z_0 *and* X_0.

#The update requires localizing the search region, finding the Lower and Upper bounds on each variable, and calculate p *and* ns values.

6.1.5 Numerical illustrations for an integer program

Example 6.1 Consider a trivial example
Maximize $Z = 3x_1 + 13x_2$

Subject to

$$2x_1 + 9x_2 \leq 40, 11x_1 - 8x_2 \leq 82, x_1, x_2 \geq 0 \text{ and integer}. \tag{6.6}$$

For the above LIP model, we let $X_0 = 0, PROB = 0.99$ *and* $Z_{max} = 58.8$ is the LP value of the objective function.

Iterative steps of the random search method are given in Table 6.1.

Table 6.1: Iterative output by the random search method.

Iteration No	Variables Value x_1, x_2	L Bound Values L1, L2	Upper Bound Values U1, U2	Z value
1	1,0	0,0	9,4	3
2	4,1	0,0	9,4	25
3	7,2	0,1	9,4	47
4	8,2	0.2	9,4	50
5	2,4	0,2	9,4	58

We have reached the optimal solution as the upper bound by the LP solution was 58.8. There is no integer value >58 and <58.8.

Figure 6.1: Steps of random search process for the example 6.1

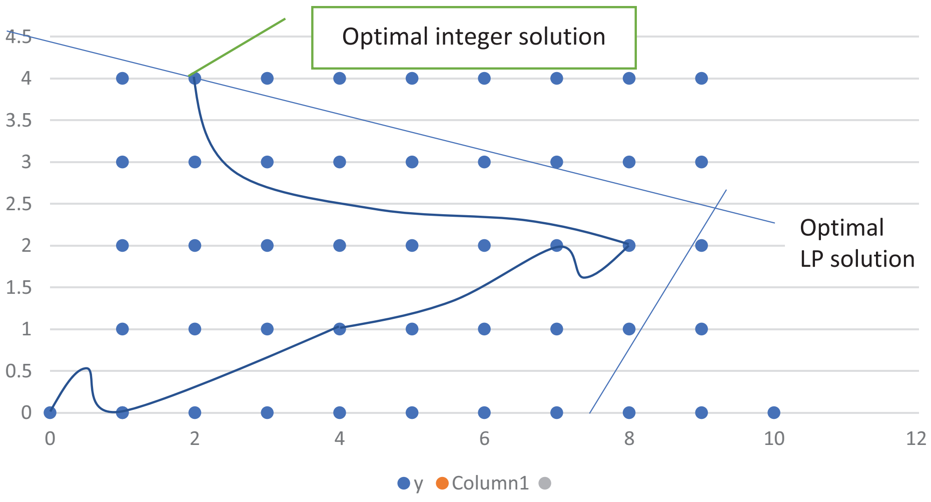


Figure 6.1: Random search.

Example 6.2 Consider the following problem.

$$Max\ Z = 3x_1 + x_2 + 2x_3 + x_4 - x_5$$

Subject to:

$$\begin{aligned} 25x_1 - 40x_2 + 16x_3 + 21x_4 + x_5 &\le 300 \\ x_1 + 20x_2 - 50x_3 + x_4 - x_5 &\le 200 \\ 60x_1 + x_2 - x_3 + 2x_4 + x_5 &\le 600 \\ -5x_1 + 4x_2 + 15x_3 - x_4 + 65x_5 &\le 700 \\ x_1, x_2, x_3, x_4, x_5 \ge 0 \ &and\ integer. \end{aligned} \tag{6.7}$$

The linear programming solution of problem (6.7) is $Z_{max} = 321.48, x_1 = 4.04, x_2 = 88.52, x_3 = 34.52,$ $x_4 = 151.78, x_5 = 0$

For application of the random search method, let the initial solution be $X_0 = 0$ and PROB =0.90. The ILP solution obtained by the random search method resulted in $Z_{ILP} = 316,\ x_1 = 4,\ x_2 = 87,$ $x_3 = 34,\ x_4 = 149,\ x_5 = 0$

A few iterations are summarized below in the Table 6.2

Table 6.2: Iterative values for Example 6.2.

Iteration Number	Value of x_1, x_2, x_3, x_4, x_5	Lower bound of x_1, x_2, x_3, x_4, x_5	Upper bound of x_1, x_2, x_3, x_4, x_5	Z value
1	5, 29, 31, 93, 0	0, 0, 0, 0, 0	10,88, 45, 153, 11	229
. . .	. . .	. . .	. . .	. . .
5	2, 85, 33, 133, 0	0, 61, 26, 82, 0	6,88, 38,153, 2	290
6	4, 87, 34, 149, 0	0, 77, 31, 122, 0	5, 88, 36, 153, 1	316
Opt. Solution	4, 87, 34, 149, 0	3, 87, 34, 147, 0	4, 88, 34, 152, 0	316

6.2 Random search method for mixed-integer programming

The random search method for mixed-integer programming was developed by Huynh, Connell and Kumar (1987) is presented in this section.

6.2.1 The mixed-integer programming problem, notation and definitions

A mixed-integer programming model is given by (6.8).

$$Max\ Z = C_1X_1 + C_2X_2$$

Subject to:

$$A_1X_1 + A_2X_2 \le b, X_1 \ge 0 \text{ and } X_2 \ge 0 \text{ and integer}$$

and

$$L \le X_2 \le U. \tag{6.8}$$

Note that *L and U are the lower and upper bounds on the integer varaiables* X_2.

Here let the real and integer variable be denoted by: $x_1, \ldots, x_{n_1}, x_{n_1+1}, \ldots, x_{n_1+n_2}$, *where* $n_1 + n_2 = n$.

In other words, n_1 number of variables are real and n_2 are integer restricted variables.

Following similar notation as were used for the random search method for the integer programming model, we have:

H = Set of integer points in the hyper-box determined by the integer variables in X_2. Assume an integer value of all variables can be determined randomly, and if that random value is feasible, one can locate value of the objective by solving a problem given by (6.9), which is:

$$Max\ Z = C_1X_1 + [C_2X_2]_{at\ known\ random\ integer\ point}$$

Subject to

$$A_1X_1 \le b - A_2X_2, X_1 \ge 0 \tag{6.9}$$

Therefore, for the model (6.8), various parameters are given by:

The number of points in the set H is given by:

$$\prod_{j=n_1+1}^{j=n_1+n_2} \left[U_{n_{1+j}} + 1 - L_{n_1+j}\right]$$

The

$$p = \frac{1}{\prod_{j=n_1+1}^{j=n_1+n_2} \left[U_{n_{1+j}} + 1 - L_{n_1+j}\right]},$$

and

$$ns = \frac{\ln(1 - PROB)}{\ln(1 - p)}$$

6.2.2 Steps of the random search algorithm

The random search approach for the mixed-integer program is similar to the approach that was used for the random search method for the integer programming model discussed in Section 6.2. These steps are:

Step 1: Calculate $Z_0 = C_1X_{1(initial)} + C_2X_{2(initial)}$

Step 2: Find the value of the number of points in the set H.

Step 3: Calculate the value of

$$p = \frac{1}{\prod_{j=n_1+1}^{j=n_1+n_2} \left[U_{n_{1+j}} + 1 - L_{n_1+j}\right]}$$

Step 4: If $p = 1,$ *then go to Step* 7,
Otherwise, calculate $ns = \dfrac{\ln(1 - PROB)}{\ln(1 - p)}$
And set s=0 (s is a counter)

Step 5: Set s=s+1, if $s \ge ns$ then go to Step 7. Otherwise randomly generate X_2 in the region H,

Step 6: Solve the problem (6.9), using the simplex method. If it is a successful solution (X_0, Z_0), and go to step 2 otherwise to Step 5.

Step 7: The process is terminated and conclude the optimal solution.

6.2.3 Numerical illustration

Example 6.3
Consider the following situation:

$$Max\ Z = x_1 + x_2$$

Where $x_1, x_2 \geq 0$ *and must satisfy* at least two constraint sets of the following three sets of constraints:

$$\begin{aligned}
&\text{Set 1: } -2x_1 + x_2 \leq 0, 4x_1 + x_2 \leq 16 && (\text{call these constraints 1 and 2})\\
&\text{Set 2: } -x_1 + x_2 \leq 0, 3x_1 - x_2 \leq 9 && (\text{call these constraints 3 and 4})\\
&\text{Set 3: } x_1 \leq 6, 2x_2 \leq 5 && (\text{call these constraints 5 and 6})
\end{aligned} \tag{6.10}$$

The constraints of the model (6.10) will generate a non-convex feasible space, see Figure 6.2. However, the problem can be reformulated as a mixed-integer model by introducing two binary variables as follows

$$Max\ Z = x_1 + x_2$$

Subject to:

$$\begin{gathered}
-2x_1 + x_2 - My_1 \leq 0,\\
4x_1 + x_2 - My_1 \leq 16\\
-x_1 + x_2 - My_2 \leq 0,\\
3x_1 - x_2 - My_2 \leq 9\\
x_1 - M(1 - y_1 - y_2) \leq 6,\\
2x_2 - M(1 - y_1 - y_2) \leq 5\\
y_1 + y_2 \leq 1,\\
x_1, x_2 \geq 0 \textit{ and real}\\
\textit{and } y_1, y_2 \geq 0 \textit{ and binary integer},\\
M \textit{ is a large quantity}
\end{gathered} \tag{6.11}$$

The feasible space, which is non-convex, is shown in Figure 6.2

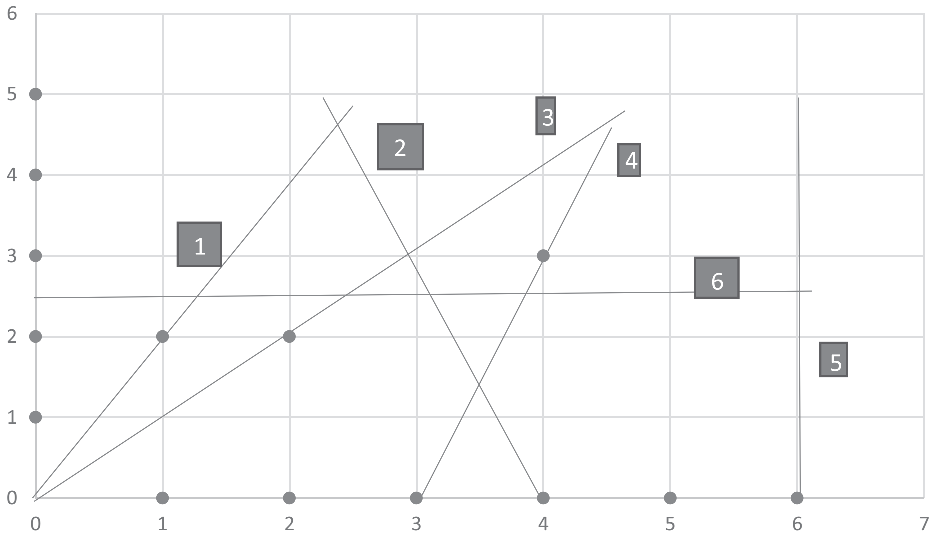


Figure 6.2: Three regions defined by constraints (1) to (6).

The solution of (6.11) was obtained by the random search method with $x_1 = x_2 = 0$, $y_1 = y_2 = 0$, PROB =0.99, and M=100. The output is given in Table 6.3.

Table 6.3: Sample output of the random search.

Iteration	Real variables x_1, x_2	Integer variables y_1, y_2	Lower bound l_1, l_2	Upper bound u_1, u_2	Objective value
1	3.8333,2.5	1,0	0,0	1,1	6.3333
2	3.2, 3.2	0,0	0,0	1,0	6.4
Optimal	3.2, 3.2	0,0	0,0	0,0	6.4

6.3 An extreme point mathematical programming problem

In this section, we present a model that is difficult to solve by analytical approaches, although attempts have been made by many researchers. It is known as an extreme point mathematical programming problem, which is a linear programming problem where an optimal solution must be feasible for a set of linear constraints and be an extreme point of another set of linear constraints. Many problems in this category can be converted as a mixed-integer program, hence their discussion in this book is appropriate. Historically, Kirby and Scobey (1970) considered a problem of manufacturing K products, each of these products could be processed by the same machine and the machine could process any *N* out of *K* products, simultaneously. Products were processed on the machine's *N* rings of dies *(N < K)*. To change the production from one set of *N* products to another set of *N* products, the entire machine is shut down while dies on the rings were changed. This problem, when formulated as a mathematical model, became a special model, which later they called as an extreme point mathematical programming problem (EPMP). To find a solution for the EPMP model, these authors (Kirby and Scobey) discovered that Charnes and Cooper (1961) had also developed an EPMP model, however, they did not provide any solution procedure for it, at that time.

6.3.1 Mathematical formulation of an extreme point mathematical programming model

A mathematical model of the EPMP problem can be stated as follows:

$$Max\ Z = C.X$$

Subject to $AX \leq B$, and X is an extreme point of the convex set H,

Where

$$H = \{X : DX \leq E,\ \ and\ X \geq 0\}. \tag{6.12}$$

Here *A is m_1 by n, B is m_1 by 1, C is 1 by n, D is m_2 by n, E is m_2 by 1, and X is n by 1.*

Solution to the model (6.12) were proposed by Kirby, Love and Swarup (1972). The basic ideas in their approach was:

(i) To rank the extreme points of the convex region defined by: $DX \leq E$ and $X \geq 0,$ and
(ii) To test the elements of a vector representing a solution for linear independence with respect to $DX \leq E$.

Kumar and Wagner (1979) further developed an algorithm for solving the model (6.12). Features of the method developed by Kumar and Wagner (1979) included:

(i) Ranking of the extreme points as was done by Kirby, Love and Swarup (1972).
(ii) However, they avoided the test for linear independence and established upper and lower bounds using a restricted base entry in the modified linear programming model:

$$Max\ Z = C.X$$

Subject to $AX \leq B, DX \leq E, and\, X \geq 0.$

Ranking of extreme point solutions was a common feature of both approaches i.e. Kirby, Love and Swarup (1972) and that by Kumar and Wagner (1979). The ranking of solutions can be time consuming, computationally difficult and inefficient.

(iii) Sen (1982), Sen and Sherali (1985), and Sherali and Sen (1985) used ideas like the cutting plane technique but these ideas also blow up the size of the problem and once again solution to the extreme point mathematical programming model remained unsatisfactory.

6.3.2 Problems that can be reformulated as an extreme point mathematical programming model: Some applications

Before we develop another method for solving an EPMPP, we discuss a few applications, where a given situation can be transformed as an extreme point mathematical programming model.

Application 6.1: A single source transportation model

Single source of transportation problem is an ordinary transportation problem where each demand is restricted to be supplied by a single source as described by Negelhout and Thompson (1980). These situations arise in military movements, where it is a common practice that troops going on a mission are preferred from the same unit. Similarly, large supermarket chains orders are filled from a single warehouse, and in computer networks, it is a common requirement, that all computations relating to a single job are performed by a single computer. Mathematical model of a single source transportation problem can be stated as the following mathematical model.

$$Min\ Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Subject to:

$$\sum_{j=1}^{n} x_{ij} \le a_i, i = 1, 2, \ldots, m$$

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

And x_{ij} is an extreme point of $x_{ij} \le b_j,\ x_{ij} \ge 0 \forall (i, j)$.

From the equality constraints, we may eliminate variables $x_{m1}, x_{m2}, \ldots, x_{mn}$ since

$$x_{mj} = b_j - \sum_{i=1}^{i=(m-1)} x_{ij}, j = 1, 2, \ldots, n. \tag{6.13}$$

The model (6.13) can be expressed as in (6.14)

$$Min\ Z = \sum_{i=1}^{m-1} \sum_{j=1}^{n} (c_{ij} - c_{mj}) x_{ij} + \sum_{j=1}^{n} c_{mj} b_j$$

Subject to

$$\sum_{j=1}^{n} x_{ij} \le a_i, i = 1, 2, \ldots, (m-1)$$

$$\sum_{i=1}^{m-1} \sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} b_j - a_m$$

$$\sum_{i=1}^{m-1} x_{ij} \le b_j, j = 1, 2, \ldots, n.$$

$\{x_{ij}\}$ is an extreme point of

$$x_{ij} \le b_j, x_{ij} \ge 0 \,\forall (i, j) \; for \; i = 1, 2, \ldots, (m-1) \; and \; j = 1, 2, \ldots .n. \tag{6.14}$$

Application 6.2: Zero-one integer linear programming model
The zero-one linear programming problem was expressed by Kirby, Love and Swarup (1972) as an extreme point mathematical programming model as follows:

$$Max \; z = C.X$$

Subject to

$$AX \le B$$

Where X is an extreme point of

$$x_j \le 1, j = 1, 2, \ldots, n \; and \; X \ge 0. \tag{6.15}$$

Thus zero-one integer linear programming is a class of the extreme point mathematical problem. There are several other applications described by William (1978).

Application 6.3: Critical path problem under assignment constraints
Chandrasekhar, Kumar and Wagner (1975) considered the critical path problem under assignment constraints and formulated a mathematical model which was in the form of an extreme point mathematical programming model. They assumed that:
(i) There are precedence constraints among activities of a project.
(ii) There are n men who can accomplish each of these activities.
(iii) The time of performing these activities depend upon men-job combination, and
(iv) The precedence constraints are assumed to be independent of assignments.

Mathematical model for the above problem
A project consisting of *n* activities with specified precedence constraints is given. In order to have the project completed, it is required that all the activities must be

accomplished. There are m men who can perform any one of these activities, but they differ in their skills, i.e. the time needed to perform any one of these activities depends on a man-activity combination. However, if the number of available persons is less than the number of activities, i.e., $m < n$, it follows that a person will have to be assigned more than one activity, but, of course, he can perform only on one activity at a time. In other words, if the person is busy with on an activity, work on another activity will have to wait for completion of activity in hand. Thus, work will have to be delayed to-commence work of the other activity until work on the activity on hand is completed.

Let the n activities be denoted by (ij), where i and j are nodes (events) of the network ($j > i$). Let $t_{ijk} \geq 0$, is time required to complete the activity (ij), *if it is assigned to person k.*

$$(ij) = 1, 2, \ldots, n \ and \ k = 1, 2, \ldots, m$$

Let $T_i =$ *Time to reach the event i or node i.* Note the given n activities will result in a network, that can be written as a mathematical model of the form (6.16).

$$Min\ z = T_L \qquad (1)$$

Subject to:

$$\left.\begin{aligned} & T_j - T_i - \sum_{k=1}^{m} t_{ijk} x_{ijk} \geq 0 \forall (ij),\ j > i && (2) \\ & \sum_{k=1}^{m} x_{ijk} = 1, \forall (ij) && (3) \\ & x_{ijk} = 0 \ or \ 1 && (4) \end{aligned}\right. \qquad (6.16)$$

And if $x_{ijk} = 1$,

then the kth person cannot be assigned to any other activity during the time he is performing the activity. Here

$$i, j = 1, 2, \ldots, L;\ (ij) = 1, 2, \ldots, n;\ k = 1, 2, \ldots, m \qquad (5)$$

The model (6.16) has five sets of components denoted by (1), (2), . . ., (5). All these relations (1) to (5) form the model (6.16).

Note that (1) represents the objective function and the inequality (2) is due to precedence constraints imposed by the structure of the network. The set of constraint (3) indicates that all the activities must be accomplished. The condition (4) indicates that all events are attended to. However, it is not easy to mathematically express the condition (5).

In all such cases, when analytical approaches are difficult and computationally demanding, sometimes it is convenient to apply the random search method, where optimality is assured with an associated probability.

6.4 Development of the random search method for the EPMP model

It is comprised of the following process.

6.4.1 Randomly generated solution

A solution may have different meaning in different situations. For example, in the case of a linear program, $Max\ g = FX, subject\ to\ DX \le E\ and\ X \ge 0,$ we know the solution is an extreme point of the feasible convex set $DX \le E\ and\ X \ge 0$. Furthermore, it is assumed that the feasible space generated by the linear constraints is bounded and non-empty. The simplex method search takes us to the extreme point that is farthest and in the direction of the vector *F*. Thus one can change the direction by randomly generating directions from a uniform distribution between two known values, for example, say it is *U(-15, 15)*. For each *F* value, the method will generate a different extreme point.

6.4.2 The number of search points

As explained, the maximum number of vertices *nv* for an *n*-dimensional polyhedron with *m* facets is given by *u(m,n)* as given by McMullen (1970). Then the probability of a vertex under investigation is the required answer is given by $p = 1/nv$.

Klee (1964, 1974) and McMullen (1970) have established that the maximum and minimum number of vertices of an n-dimensional polytope with m-facets are given by

$$U(m,n) = \binom{m - \left[\frac{n+1}{2}\right]}{m-n} + \binom{m - \left[\frac{n+2}{2}\right]}{m-n}, \text{ and } l(m,n) = (n-1)m - (n-2)(n+1) \quad (6.17)$$

In (6.17), $\lfloor k \rfloor$ denotes the greatest integer $\le k$. If we make *ms* number of such searches, the probability at least one them is correct is given by:

$$Prob = 1 - (1-p)^{ms}$$

Equivalently, the number of searches, $ms = \dfrac{\ln(1 - Prob)}{\ln(1-p)}$.

6.4.3 Reduction in search region or a successful solution

Given a feasible solution, one should be able to define another feasible solution, which is better compared to the previous one. For example, for a feasible solution

(X_0, z_0), another feasible solution (X, z) is a successful solution if $z > z_0$.Thus one can reduce the feasible region from $H = \{X : DX \leq E, X \geq 0\}$ to $H = \{X : DX \leq E, X \geq 0$ and $CX \leq z_0\}$.

From the above it is clear, that after each successful solution, the search region will be reduced, and search procedure will be accelerated.

6.4.4 A feasible pivot for an EPMP

The EPMP can be arranged as shown in Table 6.4.

Table 6.4: Initial Table of the EPMP problem.

Basic\nonbasic	$x_1 x_2 x_3 \ldots\ldots x_n$	**RHS**
s_1	**D**	**E**
s_2	**−C**	**−Z**
$\vdots$		
s_{md}		
s_{md+1}		
$Z_j - C_j$	**−F**	**Z**
R	**A**	**B**
$z_j - f_j$	**−C**	**0**

Note that the random objective is denoted by *z=FX*. Let N_b be the set of indices for basic variables and let N_g denote the set of indices for non-basic variables when $z_j - f_j \leq 0$. The entering variable is selected from the set N_g and the outgoing variable is selected in the usual way of simplex calculations. It means if d_{rt} is a pivot element, it satisfies the usual condition:

$$\frac{e_r}{d_{rt}} = Min\left\{\frac{e_i}{d_{it}} \geq 0\right\} for\ i = 1, 2, \ldots, m_d.$$

A pivot element d_{rt} for each $t \in N_g$.

An element d_{rt} is said to be feasible pivot for EPMP if and only if the simplex operation on that pivot yields a feasible solution X with respect to $AX \leq B$.

It is clear, that such a pivot exists only if the following conditions are satisfied.

(i) N_g is a non-empty set.

(ii) $\frac{b_i - e_r a_{it}}{d_{rt}} \geq 0\ for\ all\ i\ and$ t $\in N_g.$

If there are more than one feasible pivot, we select the best feasible pivot d_{uv}, which gives a maximum increase in the value of the objective function.

6.4.5 Algorithmic steps

The algorithmic approach may be described as follows:

Step 1: Arrange the given problem as arranged in Table 6.4. Initially F = C. Find the optimal solution to $Max\ Z = CX, subject\ to\ DX \le E, X \ge 0$ by using the usual simplex iterations.

Step 2: Check feasibility of the optimal solution for the constraints set Table 6.4: $AX \le b, X \ge 0$. If the solution is also feasible to $AX \le b, X \ge 0,$ stop and go to Step 5, else go to Step 3.

Step 3: If solution is not feasible, check all previous iterations, and find the extreme point which corresponded to a feasible solution to $AX \le b, X \ge 0$. Let the value of the objective CX at this feasible extreme point of $DX \le E, X \ge 0$ be denoted by z_f. Add a constraint $CX \ge z_f$.

Step 4: Generate randomly the objective vector F and rearrange the problem as in Table 6.4.

Step 5: Terminate the process and conclude the extreme point solution.

These steps are illustrated for a numerical example.

6.4.6 Illustrative example 6.4

Consider the following EPMP problem:

$$Max\ Z = x_1 + 2x_2 + x_3$$

Subject to:

$$x_1 \le 1, x_2 \le 3, x_3 \le 4, which\ is\ the\ set\ of\ constraints \equiv AX \le B$$

Where x_1, x_2, x_3 is an extreme point of:

$$\begin{gathered} 3x_1 + 2x_2 - x_3 \le 6, \\ 3x_1 + 2x_2 + 4x_3 \le 16, \\ 3x_1 - 4x_3 \le 3, \\ 2.25x_1 + 4x_2 + 3x_3 \le 17, \\ x_1 + 2x_2 + x_3 \le 10, \\ x_1, x_2, x_3 \ge 0. \end{gathered} \tag{6.18}$$

This second set of constraints is equivalent to the constraints set denoted by $DX \le E, X \ge 0$.

The second constraints set in (6.18) has been taken from Balinski (1961) and they called the convex polyhedron generated by these constraints as a 'Baby' convex polyhedral. All vertices of these polyhedron are shown in Table 6.5

Table 6.5: Exhaustive search for the optimal solution.

List of vertices x_1, x_2, x_3	Values of Z	Inspection of the set EP set of (16.8)
0, 0, 0	0	Feasible
0, 0, 4	4	Feasible
0, 2, 3	7	Optimal
0, 7/2, 1	8	Infeasible
0, 3, 0	6	Feasible
1, 0, 0	1	Feasible
1, 3/2 0	4	Feasible
4/3, 2, 2	22/3	Infeasible
7/3, 0, 1	10/3	Infeasible
8/3, 0, 2	14/3	Infeasible

Now let us reconsider this problem by using the random search method.

The given problem (6.18) is arranged in Tabular form as shown in Table 6.6.

Table 6.6: Initial table of the given problem (6.18).

Basis\N Basis	x_1	x_2	x_3	RHS
x_4	3.00	2.00 (Pivot)	−1.00	6.00
x_5	3.00	2.00	4.00	16.00
x_6	3.00	0.00	−4.00	3.00
x_7	2.25	4.00	3.00	17.00
x_8	1.00	2.00	1.00	10.00
x_9	−1.00	−2.00	−1.00	0.00
$z_j - c_j$	−1.00	−2.00	−1.00	0.00
r_1	1.00	0.00	0.00	1.00
r_2	0.00	1.00	0.00	3.00
r_3	0.00	0.00	1.00	4.00
$g_j - f_j$	−1.00	−2.00	−1.00	0.00

Note that the variables $x_4, \ldots, x_8$ are the slack variables and x_9 is the slack variable for the additional constraint $x_1 + 2x_2 + x_3 \geq 0$. The pivot operation will give rise to Table 6.7, indicating x_2 basic and the slack variable x_4 as a non-basic variable. These calculations are shown with respect to the given objective function.

Table 6.7: Results after the pivot operation.

Basis\N Basis	x_1	x_4	x_3	RHS
x_2	1.50	0.50	−0.50	3.00
x_5	0.00	−1.00	5.00	10.00
x_6	3.00	0.00	−4.00	3.00
x_7	−3.75	−2.00	5.00 (Pivot)	5.00
x_8	−2.00	−1.00	2.00	4.00
x_9	2.00	1.00	−2.00	6.00
$z_j - c_j$	2.00	1.00	−2.00 $\uparrow$	6.00
r_1	1.00	0.00	0.00	1.00
r_2	−1.50	−0.50	0.50	0.00
r_3	0.00	0,00	1.00	4.00
$g_j - f_j$	2.00	1.00	−2.00	6.00

For an interpretation of the outcome of the pivot operation in Table 6.7, note that the current value of the objective has not reached its optimum as $(z_j - c_j)$ is equal to -2 for the variable x_3. Hence variable x_3 must enter the basis. Consequently, the outgoing variable will be x_7 as shown in Table 6.7. Further it may also be noted from the Table 6.7 that $r_1 = 1$, $r_2 = 0$ and $r_3 = 4$ indicates that the basic solution in Table 6.7 is also a feasible solution to the given constraints $AX \leq B$, which is given by $x_1 = 0$, $x_2 = 3$ and $x_3 = 0$.

If the pivot operation was carried out, the variable x_3 will enter and x_7 will go out of the basis. One can easily verify updated values of the basic variables will be as given in Table 6.8.

Table 6.8: Giving only partial information resulting from the pivot operation.

basic	x_2	x_5	x_6	x_3	x_8	x_9	z_j-c_j	r_1	r_2	r_3	$g_j - f_j$
value	3.5	5	7	1	2	8	8	1	−0.5	3	8

This clearly is an infeasible solution with respect to given constraints $AX \leq B$. Note that $r_2 = -0.5$, *means that* $x_2 = 3.5$. Hence it is not the required optimal solution, we were looking for. By the application of the random search method, we generate three random numbers as the coefficients of three variables in the random objective function and resolve the problem. Note that the simplex procedure takes to the farthest point in the direction of the normal to the objective plane, Munapo and Kumar (2013, 2014). The randomly generated coefficients of the objective function are likely to change the direction of the normal, hence likely to reach to some other extreme point of the same polyhedron.

The updated model to be analysed will be given by (6.19):

$Max\ Z = x_1 + 2x_2 + x_3$ and the random objective considered $Max\ Z = x_1 + 3x_2 + 6x_3$

Subject to:

$$x_1 \le 1, x_2 \le 3, x_3 \le 4, which\ is \equiv AX \le B$$

Where x_1, x_2, x_3 is an extreme point of:

$$\begin{gathered} 3x_1 + 2x_2 - x_3 \le 6, \\ 3x_1 + 2x_2 + 4x_3 \le 16, \\ 3x_1 - 4x_3 \le 3, \\ 2.25x_1 + 4x_2 + 3x_3 \le 17, \\ x_1 + 2x_2 + x_3 \le 10, \\ x_1, x_2, x_3 \ge 0. \end{gathered} \tag{6.19}$$

Note that a new constraint has been added to the set of constraints $DX \le F$, as the objective value 6 was a feasible and 8 was an infeasible solution. True solution will be bounded by the condition $6 \le z < 8$.

The model (6.19) is again solved and the final simplex is given in Table 6.9.

Table 6.9: Optimal solution to model (6.19).

Basis\N Basis	x_1	x_5	x_7	RHS
x_2	0	−3/10	2/5	2.0
x_4	15/4	1	−1	5.0
x_6	6	8/5	−4/5	15.0
x_9	−1/4	−1/5	3/5	1.0
x_8	1/4	1/5	−3/5	3.0
x_3	3/4	2/5	−1/5	3.0
$z_j - c_j$	5/4	3/5	2/5	26.00

Table 6.9 (continued)

Basis\N Basis	x_1	x_5	x_7	RHS
r_1	1.00	0.00	0.00	1.00
r_2	−1.50	−0.50	0.50	1.00
r_3	0.00	0,00	1.00	0.00
$g_j - f_j$	2.00	1.00	−2.00	6.00

This solution indicates that when x_9=1, it means the given objective value is 7, and values of other variables is given by $x_2 = 2, x_3 = 3,$ which is also feasible to the given constraints $AX \le b$. The new bounds for the objective value is $7 \le z < 8$.

If the model is resolved, random objective will not be able to change the extreme point as there is only one extreme point in the region when $CX \ge z_f = 7$ is added. Thus, the current solution is optimal, which is the same as was given by the exhaustive search results in Table 6.5.

6.5 Conclusion

The random search method discussed in this chapter is a useful tool for managing situations that are otherwise difficult to deal with. The application of the random search method is not a set of specified steps, but this method can be applied with care and creative thinking. The optimality of a solution obtained by the random search approach is ascertained with a specified probability. Many more applications of the random search methods arise in context of a multi-objective linear integer programming problem, see Kumar et al. (2018a), where the random search method has been developed to find the ordered optimal solutions for an assignment problem, see Kumar et al. (2020) and in Al-Hasani et al. (2020), a method has been developed for generating non-dominated point set for a multi-objective model.

References

Al-Hasani, A., Al-Rabeeah, M., Kumar, S., and Eberhard, A. (2020), Rank-based Solution Methods and their applications in Determination of non-dominated Points set for a Multi-Objective Integer Programming Model, International Journal of Mathematical, Engineering and Management Sciences, Vol. 5, N0 6, 1249–1269. http://doi.org/10.33889/IJMEMS.2020.5.6.093.

Kumar S., Al-Hasani, A., Al-Rabeeah, M., Eberhard, A. (2020), A random search method for finding K>=2 number of ranked optimal solutions to an assignment problem, J. of Physics, Conf Series, 1490 (012063), doi: 10.1088/1742-6596/1490/1/2063, pp 13.

Balinski, M.L. (1961) An algorithm for finding all vertices of a convex polyhedral sets, J. of Soc. Industrial and Applied Maths. 9, 72–88.

Chandrasekaran, R., Kumar, S. and Wagner, D. (1975) Critical path problem under assignment constraints, Department of Operations Research, Case Western Reserve University, Cleveland, Technical Memorandum No 357.

Charnes, A. and Cooper, W.W. (1961) Mathematical models and industrial applications of Linear Programming, Vol 1 and 2, John Wiley & Sons, Ne York.

Hillier, F. and Lieberman, S. (2004). Introduction to Operations Research, Tata McGraw-Hill Education.

Huynh, H.N., Kumar, S. and Connell, H.J. (1987). Random search method for integer programming, Proc of the International Conference on Optimization Techniques, (Editor) K.L Teo, H. Paul, K.L. Chew, and M.C. Wong, 1987, pp 601–610, University of Singapore.

Huynh, H.N., Connell, H.J. and Kumar, S. (1987). Random search Method for Mixed-integer programming, ASOR 1987 Proc of the OR Conference, (Editor) Kumar, S. pp109–118.

Huynh, H.N. and Kumar, S. (1988). A random search method for extreme point Mathematical programming, Asia Pacific Journal of Operations Research, Vol. 7, pp30–45.

Kirby, M.J.L. and Scobey, P.F. (1970) Production scheduling on n identical machines, Journal of Canadian Operations Research, Vol 8, pp14–27.

Kirby, M.J.L., Love, H.R. and Swarup, K. (1972) Extreme point mathematical programming, Management Science, Vol 18, pp540–49.

Klee, V. (1964) On the number of vertices of a convex polytope, Canadian J Maths 16, pp701–720.

Klee, V. (1974) Polytopes pairs and their relationships to linear programming, ACTA Mathematica, 133, pp1–25.

Kumar, S., Al-Hasani, A., Al Rabeeah, M., and Ebehard, A. (2018) A random search method for finding number of ranked optimal solution to an-assignment problem. In http://www.optimization-online.org/DB_HTML/2018/11/6942.Html. 2018a.

Kumar, S., Munapo, E., Lesaoana, M., and Nyamugure, P. (2018). Some innovations in OR Methodology: Linear optimization, Lambert Academic Publishing, ISBN: 978-613-7-38007-9.

Kumar, S and Wagner, D. (1979) Some algorithms for solving extreme point mathematical programming problems, Journal of New Zealand Operational Research, Vol 7, pp127–49.

McMullen, P. (1970) The maximum number of faces of a convex polytope, Mathematica, 17, 127–149.

Munapo, E., and Kumar, S., (2013) Solving large-scale linear optimization problem with non-negative coefficients by transforming n-variable LP into two-variable LP problem, ASOR Bulletion, Vol 32, No 1, pp1–12.

Munapo, E., and Kumar, S., Leasaoana, M., and Nyamugure, P., (2014) Solving a large-scale LP model with non-negative coeficients: Anhybrid search over the extreme points and the normal direction to the given objective function, ASOR Bulletion, Vol 33, Issue 1, pp 11–23.

Naglehout, R.V., and Thompson, G.L. (1980) A single source transportation algorithm, Computing and Operations Research, 7, pp185–198.

Nemhauser, G.L. and Wolsey, L.A. (1988). Integer and combinatorial optimization, Inter Science Series in Discrete Mathematics and Optimization, John Wiley and Sons.
Nemhauser, G.L. and Wolsey, L.A. (1989). Chapter 6, Integer Programming, HND Book in Operations Research and Management Sciences, pp 447–527.
Salkin, H.M. (1971). Foundations of integer programming, North Holland.
Sen, S. and Sherali, H.D. (1985) A branch and bound algorithm for extreme point mathematical programming problem, Discrete Applied Mathematics, Vol 11, pp 265–280.
Sen, S. (1982) The extreme point mathematical programming problem, PhD Thesis, Virginia Polytechnic Institute and State university.
Sherali, H.D. and Sen, S. (1985) A disjunctive cutting planes for extreme point for the mathematical programming problem, Opsearch, 22, pp83–94.
Taha, H.A. (2006). Operations Research: An Introduction, Pearson Educators, 7th Edition.
Taha, H.A. (2014). Integer programming: theory, applications and computations, Academic Press.
Williams, H.P. (1978) Model building in mathematical programming, Wiley and Sons, New York 1978.
Winston, W.L. (2004). Operations Research: Applications and Algorithms, Duxbury.

Chapter 7
Some special linear integer models and related problems

Abstract: The assignment and transportation problems are well known, and they have applications in production planning, telecommunication, scheduling, military operations etc. This Chapter presents a See-Saw approach for solving the assignment and transportation problems. In this approach two columns are paired and the See-Saw movement is done in such a way that when moving up in one column then we have to move down in the other column and vice versa. The current solution is improved by See-Saw moves until optimality is reached. The See-Saw moves are very simple. The (n-1) 'See-Saw' moves for a fixed column can be calculated at the same time by parallel processors. Even the n various columns can also be handled at the same time by parallel processing. This is not the case with both the Hungarian and transportation simplex methods. Every see-saw move occupies two new cells. We have also discussed a method to find a very a good starting solution for the transportation problem and briefly discussed the problem of ranked optimal solution for an assignment problem.

Keywords: Assignment problem, Transportation problem, Sew-Saw methods, Transportation simplex algorithm, North-west corner method, Least cost method, Vogel's approximation method, Transportation simplex method, Network simplex method, Ranked optimal solution

7.0 Introduction

The linear programming has played a major role in quantitative models arising in real-life industrial situations. The class of special linear integer models include problems like assignment, transportation, and transshipment models, which have been discussed in many books, see Hillier and Lieberman (2015), Taha (2017), Winston (2004). In this chapter, a See-Saw approach has been developed for solving the assignment and transportation models. This new approach is suited for parallel computing. This chapter has been organized as follows:

1. General discussion on the assignment model is discussed in section 7.1.
2. A 'See-Saw' approach to an assignment problem has been presented in section 7.2.
3. In sections 7.3, we relook at the starting solution for the transportation problem and develop a method for improved starting solution of the transportation problem.
4. In section 7.4, we develop a 'See-Saw' approach for the transportation model.

https://doi.org/10.1515/9783110703023-007

5. In section 7.5 we discuss the problem of finding ranked optimal solutions for an assignment problem, and finally make a few concluding remarks in section 7.6.

7.1 The assignment problem

The assignment problem is a well-known linear integer model, which is a special case of the general transportation and linear programming problem. Since it is a degenerate problem, its solution by standard LP method is not suited for this model. In the case of an assignment problem, each supply point has exactly one unit to supply in each row and the demand at each demand point is also one for each column. Solution to an assignment model have been developed by Aboli, et al. (2020), Niv, et al. (2020), Oncan, et al. (2019), Zhang, et al. (2019). There are many available exact methods for solving an assignment model, which include, the Hungarian method of assignment, (see Date and Nagi (2016); Edmonds and Karp (1972); Kuhn (1955); Kumar, Ncube, and Munapo (2003); Munkres (1957); Quddoos and Rabbani (2019); Tomizawa (1971)). Other approaches are the transportation simplex method, linear programming approach and network simplex method. Of these approaches, the network simplex method is the most efficient one.

The ‘Sew-Saw’ approach for solving the assignment problem is discussed in section 7.2. In this approach two-column paired movement is done in such a way that when moving up in one column then we must move down in the other column and vice a vera. The solution is verified for optimality and if optimal then algorithm stops. If not, then use the transportation simplex to move to the optimal solution. This method is suited for parallel computing.

Let an assignment problem be given as shown by Table 7.1.

Table 7.1: Assignment problem in general.

						Source
	c_{11}	c_{12}	c_{13}	…	c_{1n}	1
	c_{21}	c_{22}	c_{23}	…	c_{2n}	1
	c_{31}	c_{32}	c_{33}	…	c_{3n}	1
	…			…		…
	c_{m1}	c_{m2}	c_{m3}	…	c_{mn}	1
Demand	1	1	1	…	1	D

Where c_{ij} is the cost of assignment when j^{th} job is assigned to the i^{th} source? This is a balanced problem, i.e., number of rows (n) is equal to the number of columns (m), means $D = n = m$. The objective is to minimize the total assignment cost.

In mathematical form the assignment problem can be represented as given in (7.1).

$$\text{Minimize } Z = \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{n} c_{ij}x_{ij},$$
$$\sum_{j}^{m} x_{ij} = 1, i = 1, 2, \ldots, n,$$
$$\sum_{i}^{n} x_{ji} = 1, j = 1, 2, \ldots, n$$
$$x_{ij} \geq 0. \tag{7.1}$$

Solving (7.1) directly as a linear programming model will give the optimal solution, unfortunately (7.1) is a highly degenerate case and this makes the simplex method inefficient for solving the model (7.1).

7.1.1 Features of the assignment model

Feature 7.1: Subtracting or adding a constant to a row or column does not change optimality of the solution.

In general subtracting or adding a constant to a row or column does not change the optimal solution of an assignment problem. Let p_i be a constant subtracted from row i and q_j be constant subtracted from column j. Thus, the cost element c_{ij} changes to $\bar{c}_{ij}$ as given in (7.2).

$$\bar{c}_{ij} = c_{ij} - p_i - q_j. \tag{7.2}$$

This means:

$$\begin{aligned}
\text{Total cost} &= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \;=\; \sum_{i=1}^{n}\sum_{j=1}^{m} \left(c_{ij} - p_i - qj\right)x_{ij},\\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} - \sum_{i} p_i\left(\sum_{j} x_{ij}\right) - \sum_{j} q_j\left(\sum_{i} x_{ij}\right),\\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} - \left(\sum_{i} p_i\right)(d_j) - \left(\sum_{j} q_j\right)(a_i)\\
&= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} - \text{constant}.
\end{aligned} \tag{7.3}$$

7.1.2 Kuhn-Tucker conditions

Since (7.1) has equality constraints, a Lagrangian function is constructed as given by (7.4).

$$L = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} + u_j\left(1 - \sum_{j}^{n} x_{ij}\right) + v_i\left(1 - \sum_{i}^{n} x_{ij}\right) \tag{7.4}$$

Where u_i and v_j are Lagrangian multipliers.

The necessary first order optimality conditions are given in (7.5).

$$\begin{aligned}\frac{\partial L}{\partial x_{ij}} &= \frac{\partial\left[\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}\right]}{\partial x_{ij}} + u_j\frac{\partial\left[1-\sum_{j}^{n} x_{ij}\right]}{\partial x_{ij}} + v_i\,\frac{\partial\left[1-\sum_{i}^{n} x_{ij}\right]}{\partial x_{ij}}\\ \frac{\partial L}{\partial u_i} &= 1 - \sum_{i}^{n} x_{ij} = 0\\ \frac{\partial L}{\partial v_j} &= 1 - \sum_{j}^{n} x_{ij} = 0\end{aligned} \tag{7.5}$$

7.1.3 Transportation simplex method

In this approach the Lagrangian multipliers are used to improve a given starting solution. The current solution is improved until it becomes optimal. In this approach the multipliers are calculated from current basic cells. An index for each of the unoccupied cell is calculated from the determined multipliers and that cell with largest index is the best candidate for occupation. Optimality stops when there is no more candidate for occupation.

7.1.4 Hungarian approach

The Hungarian method was developed and published by Kuhn (1955). Kuhn named his approach as the ‘Hungarian method of assignment’ because his approach was based on the works of two Hungarian mathematicians Kőnig (1931) and Egerváry (1931). This work of the two Hungarian mathematicians are summarized as follows.

Kőnig’s theorem

In a bipartite graph $G = (S, T; E)$, the minimum number of elements of S exposed by a matching is equal to the maximum of the deficit $h(X)$ over the subsets of S where $h(X) := |X| - |\Gamma(X)|$ and $\Gamma(X)$ denotes the set of elements of T having a neighbour in X. Particularly, there is a match covering S if and $|\Gamma(X)| \geq |X|$ holds for every subset $X \subseteq S$.

Egerváry's theorem
In the same year of 1931, Kőnig's theorem was extended to Egervary's theorem

Let $G = (S, T; E)$, be a complete bipartite graph with $|S| = |T|$ and let

$c: E \rightarrow Z_+$ be a nonnegative integer-valued weight function. The maximum weight of a perfect matching of G is equal to the minimum weight of a nonnegative, integer-valued, weighted-covering of c where a weighted-covering is a function $\pi: S \cup T \rightarrow Z_+$ for which $\pi(u) + \pi(v) \geq c(uv)$ for every $uv \in E$ and the weight of π is defined to be $\sum [\pi(v): v \in SUT]$.

7.1.5 Tsoro and Hungarian hybrid approach

This is a hybrid of the Hungarian method and a Tsoro winning strategy, Kumar, Ncube, and Munapo (2003). Tsoro is a two-person game which is widely played in Southern African countries. The idea of the game is to optimize (maximize or minimize) the number of playing stones depending on the objective of the game.

7.2 See-Saw rule and its application to an assignment model

The see-saw rule is derived from the see-saw game. In a see-saw game two people sit on rod fixed in the middle and when one person goes up the other goes down and vice versa.

The see-saw principle can be used to solve an assignment problem. In this case two columns are paired at a time. Movement is done in one column such that when one goes up then in the other column the movement goes down in such a way the total cost is decreased. The see-saw game is shown in Figure 7.1 and the initial assignment problem is as given by Table 7.2.

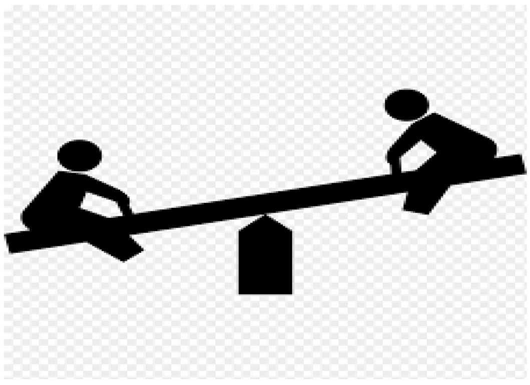


Figure 7.1: The see-saw game.

Table 7.2: Initial position.

c_{1j_1}	…	c_{1j_2}
c_{2j_1}	…	c_{2j_2}
…	…	…
$c_{(n-1)j_1}$	…	$c_{(n-1)j_2}$
c_{nj_1}	…	c_{nj_2}

Suppose the current allocation is shown in yellow in Table 7.2. The current combined cost for columns j_1 *and* j_2 is given by $(ccc_{j_1j_2})$. For the data in Table 7.2, it is given in (7.6).

$$ccc_{j_1j_2} = c_{2j_1} + c_{(n-1)j_2} \tag{7.6}$$

Movement is possible if the new combined cost is less or equal to current combined cost. Suppose the allocation in the two columns is changed as shown in Table 7.3.

Table 7.3: New position.

c_{1j_1}	…	c_{1j_2}
c_{2j_1}	…	c_{2j_2}
…	…	…
$c_{(n-1)j_1}$	…	$c_{(n-1)j_2}$
c_{nj_1}	…	c_{nj_2}

The new current cost is given by:

$$cc_{j_1j_2} = c_{(n-1)j_1} + c_{2j_2}$$

See-Saw movement is done in such a way that rows are exchanged. This is done to ensure feasibility of the new solution. The see saw movement is desirable only if and only if (7.7) is satisfied.

$$c_{2j_1} + c_{(n-1)j_2} - \left(c_{(n-1)j_1} + c_{2j_2}\right) \le 0 \tag{7.7}$$

Pairing of columns

Pairing of columns is done from the left to the right i.e. select first column and pair it with all the (n-1) on the right and select the best see-saw move. Pairing of columns can be done from right to left and still get the same optimal solution. In this chapter columns are paired but pairs can be developed in any way one likes. We will get the same optimal solution.

7.2.1 Starting solutions

Just like the transportation simplex method this method requires a starting solution. Its efficiency depends on how near the starting solution is to the optimal. In this chapter we recommend the Least Cost (LC) method and Vogel's Approximation (VA) method for an initial starting solution.

The least cost method finds a starting solution by targeting the least cost cells. Once a cell is allocated, the corresponding row and column is crossed out. If there is a tie, it is resolved arbitrarily. The procedure is repeated until all rows or columns are satisfied.

In Vogel's approximation method both row and column penalties are used in allocating cells. The cell with the largest row and or column penalty is selected for allocation, ties are broken arbitrarily, and the process is repeated until all rows and columns are satisfied.

7.2.2 See-Saw algorithm

The See-Saw Algorithm for assignment problem is summarized as follows.

Step 1: Determine a stating solution for the assignment problem

Step 2: Pair the columns starting from the left. Select the best see-move. Repeat procedure until there is no see-saw move possible and go Step 3.

Step 3: Current solution is optimal.

7.2.3 Numerical Illustration 1

This algorithm is illustrated on the following 6X6 assignment problem given in Table 7.4.

Table 7.4: Given example.

							Row
	20	16	6	18	48	26	1
	28	48	4	64	36	24	2
	88	32	38	44	30	38	3
	4	4	6	2	2	2	4
	62	64	8	86	56	82	5
	50	124	4	58	92	44	6
Column	1	2	3	4	5	6	

Using the See Saw rule we have the following steps:

Step 1: Starting solution using the least cost method is given in Table 7.5.

Table 7.5: Least cost starting solution.

20	**16**	6	18	48	26
28	48	**4**	64	36	24
88	32	38	44	**30**	38
4	4	6	**2**	2	2
62	64	8	86	56	82
50	124	4	58	92	**44**
c_1	c_2	c_3	c_4	c_5	c_6

Where $c_1 - c_2$ stand for columns 1 to 6.

Total cost =158.

Fixing c_1

$c_1 \& c_2 : c_{11} + c_{25} - (c_{15} + c_{21}) = 20 + 64 - (62 + 16) = +6 > 0$. See-Saw move is impossible.

$c_1 \& c_3 : c_{12} + c_{35} - (c_{15} + c_{32}) = 28 + 8 - (62 + 4) = -30 \leq 0$. See-Saw move is possible.

$c_1 \& c_4 : c_{14} + c_{45} - (c_{45} + c_{15}) = 4 + 86 - (2 + 62) = +26 > 0$. See-Saw move is impossible.

$c_1 \& c_5 : c_{13} + c_{55} - (c_{15} + c_{53}) = 88 + 56 - (62 + 30) = +52 > 0$. See-Saw move is impossible.

$c_1 \& c_6 : c_{15} + c_{65} - (c_{15} + c_{66}) = 50 + 82 - (62 + 44) = +26 > 0$. See-Saw move is impossible. (7.8)

Selecting See-Saw move with the largest negative (-30) we have Table 7.6.

Table 7.6: First See-Saw move.

20	**16**	6	18	48	26
28	48	4	64	36	24
88	32	38	44	**30**	38
4	4	6	**2**	2	2
62	64	**8**	86	56	82
50	124	4	58	92	**44**
c_1	c_2	c_3	c_4	c_5	c_6

Total cost = 128.

We now fix column c_2 and repeating the process for $c_3 - c_6$ we can note that there is no See-Saw move that is possible.

7.2.4 Proof of optimality

The See-Saw algorithm uses a current solution to examine the possibility of moving into unoccupied cells.

Possible to have a See-Saw move, if: $c_{2i} + c_{(n-1)j} - (c_{(n-1)1} + c_{2j}) \le 0$

See-Saw move is undesirable, if: $c_{2i} + c_{(n-1)j} - (c_{(n-1)1} + c_{2j}) > 0$

The See-Saw algorithm examines all unoccupied cells.

Algorithm terminates if See-Saw moves are impossible to all unoccupied cell.

When no move is possible, it implies that the current solution cannot be improved and hence is optimal.

In conclusion, it may be stated that the See-Saw moves are simple. The (n-1) See-Saw moves for a fixed column can be calculated at the same time by parallel processors. Even the n various columns can also be handled at the same time. This is not the case with both the Hungarian and transportations simplex methods. Every see-saw move occupies two new cells. The assignment model has application in the Weapon Target Assignment (WTA) problem. This is the problem of assigning weapons to targets while considering the maximum probability of kill, see Kline, Ahner and Hill (2019). The assignment problem is also used in scheduling problem large hospitals, transport industry, manufacturing, and production Lan et al. (2019).

7.3 The transportation problem

Let any transportation problem be given by Table 7.7.

Table 7.7: Transportation problem in general.

						Source
	c_{11}	c_{12}	c_{13}	$\dots$	c_{1n}	s_1
	c_{21}	c_{22}	c_{23}	$\dots$	c_{2n}	s_2
	c_{31}	c_{32}	c_{33}	$\dots$	c_{3n}	s_3
		$\dots$			$\dots$	$\dots$
	c_{m1}	c_{m2}	c_{m3}	$\dots$	c_{mn}	s_m
Demand	d_1	d_2	d_3	$\dots$	d_n	D

Where c_{ij} is the cost of transporting one unit from the source s_i to the destination d_j. It is a balanced transportation problem, if $d_1 + d_2 + \dots + d_n = D = s_1 + s_2 + \dots + s_m$.

7.3.1 Existing methods to find a starting solution for the transportation problem

Transportation model in linear programming has many interesting features, such as the LP formulation has a coefficient matrix that is unimodular implying that the continuous optimal solution is also the integer solution. The main challenge is that the LP model is degenerate resulting in the simplex method being inefficient for this kind of problems. The interior point algorithms are not affected by degeneracy and are good for very large transportation problems. Comparing with the interior point algorithms the simplex method is very good for small and medium sized LP models. There are improved versions of the simplex method specifically developed to handle transportation problem. These are the transportation simplex method and the network simplex method. Of these two, the network simplex method is more recent and more efficient. These two approaches require a starting solution and there is need to improve the starting solutions. The efficiency of the transportation and network simplex method depend on the starting solution. There are three main methods available for determining the starting solutions. These are north-west, least coast and Vogel's methods. In section 7.4, we discuss a method for determining a good starting solution quickly. The method is called method of subtractions as it is made of subtraction operations only.

The existing methods to find a starting solution for the transportation problem are briefly presented below:

7.3.1.1 North-West corner method

The **North-West Corner** method computes a starting solution of a transportation problem. In this approach, the basic variables are selected from the extreme top left corner to develop a feasible solution. This approach is simple and reliable, and it is easy to compute. The obvious disadvantage is that it does consider the transportation cost, therefore, it can result in a high cost starting solution. In terms of iterations to move towards an optimal solution can be time consuming and is generally not preferred.

7.3.1.2 Least Cost method

The Least Cost Method starts obtaining a feasible solution by starting from the least-cost cells. The justification is that if more lower cost cells are chosen in the initial starting solution, the lower the total transportation cost will be and lesser number of iterations will be required to move to an optimal solution. This method has the advantage of giving a near optimal solution to the transportation problem and it is very simple and easy to implement to find a starting solution. In the presence of ties, this method can be challenging to select the cell which is likely to give the total lower cost.

7.3.1.3 Vogel's Approximation Method (VAM)

Vogel's Approximation Method considers cost like the least cost method. The difference is that the allocation starts in the cell with the largest difference between the two smallest or lowest costs in rows and columns. The advantage of this method is that it is highly accurate and as result it takes fewer iterations to reach to the optimal solution. The disadvantage is that it is tedious when it is implemented to large-sized problems.

Since each method has its limitations, it is necessary to generate a simple and an accurate approach to come up a good starting solution.

7.3.2 Transportation simplex method

The transportation simplex method is also known as the modified distribution (MODI) method and the steps are summarized as follows.

Step 1: From the starting solution with $(m+n-1)$ occupied cells, calculate u_i and v_j such that $c_{ij}=(u_j+v_j)$ and that either u_i or v_j=0.

Step 2: From the unoccupied cells calculate Δ_{ij} such that $\Delta_{ij}=c_{ij}-(u_i+v_j)$. If all $\Delta_{ij}>0$ then current solution is optimal and unique. If any $\Delta_{ij}\geq 0,$ the current solution is optimal and alternate solution exists. If any $\Delta_{ij}<0,$ than the current solution must be improved and go to the cell $(i,\ j)$.

Step 3: Select the unoccupied cell with the most negative Δ_{ij}. Ship the maximum amount to this cell and make the necessary adjustments to the other occupied cells.

Step 4: Return to Step 2

Step 5: This method is efficient if the starting solution is the near optimal solution. Therefore, a need to develop a method to find a good starting solution is desirable.

7.3.3 Network simplex method

This is an efficient method for solving the minimum cost network flow problem (MCNFP). Any transportation (assignment and transshipment included) problem can be formulated as a MCNFP by use of a dummy-arcs and or node.

Step 1: Determine the $(n-1)$ starting balanced feasible solution (bfs) which corresponds to a spanning tree. Indicate non-basic variables at their upper bound by coloured arcs.

Step 2: Compute the simplex multipliers, $(y_1, y_2, ..., y_n),)$ by solving $u_1 = 0$ and $c_{ij} = y_i - y_j)$ for all basic variables x_{ij}. For all non-basic variables, determine the row 0 coefficient from $\bar{c}_{ij} = y_i - y_j - c_{ij}$. The current *bfs* is optimal if $\bar{c}_{ij} \leq 0 \ \forall \ x_{ij} = L_{ij}$ and $\bar{c}_{ij} \geq 0 \ \forall \ x_{ij} = U_{ij}$. If the *bfs* is not optimal, choose the non-basic variable that most violates the optimality conditions as the entering basic variable.

Step 3: Identify the cycle that is created by adding the arc corresponding to the entering variable to the current spanning tree of the current *bfs*. Use conservation of flow to determine the new values of the variables in the cycle. The variable that exits the basis will be the variable that first hits its upper or lower bound as the value of the entering basic variable is changed.

Step 4: Find the new *bfs* by changing the flows of the arcs in the cycle found in Step 3 and return to Step 2.

Just like the transportation simplex method the efficiency of the network simplex method is determined by the starting solution. The more accurate the starting solution the fewer the steps to optimality.

7.3.4 The method of subtractions for an initial starting solution

7.3.4.1 Derivation

The derivation comes from the fact that subtracting or adding a constant to a row or column does not change the optimal solution. The transportation problem can be represented as given in (7.9).

$$\begin{aligned} \text{Minimize } Z = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}, \\ \sum_{j}^{m} x_{ij} = s_j, \\ \sum_{i}^{n} x_{ij} = d_i. \\ x_{ij} \geq 0. \end{aligned} \tag{7.9}$$

Let p_i be constant subtracted from row i and q_j be constant subtracted from column j. Thus, the constant element c_{ij} changes to $\bar{c}_{ij}$ as given in (7.10).

$$\bar{c}_{ij} = c_{ij} - p_i - q_j.$$

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} &= \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{ij} - p_i - p_j),\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} - \sum_i p_i\left(\sum_j x_{ij}\right) - \sum_j q_j\left(\sum_i x_{ij}\right),\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} - \left(\sum_i p_i\right)(d_j) - \left(\sum_j q_j\right)(s_i),\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} - \text{constant}. \end{aligned} \tag{7.10}$$

7.3.4.2 Cost matrix with at least a zero in every row and column

From Table 7.8 we can create at least a zero in every row and every column. This is done by selecting a row minimum and then subtracting it from all elements in that row. This is repeated for all rows and all columns to obtain Table 7.8.

Table 7.8: At least a zero in every row and column of the cost matrix.

						Source
	c'_{11}	c'_{12}	0	…	c'_{1n}	s_1
	0	c'_{22}	c'_{23}	…	c'_{2n}	s_2
	c'_{31}	0	c'_{33}	…	c'_{3n}	s_3
		…			…	…
	c'_{m1}	c'_{m2}	c'_{m3}	…	0	s_m
Demand	d_1	d_2	d_3	…	d_n	

Where $c'_{ij} = c_{ij} + k_{ij}, k_{ij} \geq 0$ and constant.

If allocation can be done to zero elements only then the current solution is optimal. If not possible to allocate to zero elements only then there is a need to create new zeros.

7.3.4.3 Allocation to zero elements

With the method of subtractions allocation is done to furthest zero. In this case the furthest zero is the zero that with the greatest difference from the next smallest number its row or column. If there is a tie, then the zero with larger numbers in its row or column is selected. Immediately after an allocation to the selected zero the satisfied row and or column is removed from the problem. The reduced cost matrix must have at least a zero in its row or column. If not at least a zero is created in the remaining rows and columns.

7.3.4.4 The method of subtractions

The method of subtractions is summarized as follows.

Step 1: Ensure that the given transportation problem is a balanced problem. This is done by creating a dummy row or column.

Step 2: Create at least a zero in every row and every column.

Step 3: Identify the furthest zero. If there is a tie, select and allocate that zero with larger numbers in its row or column.

Step 4: Remove the satisfied row and or column and return to Step 2 until all rows and columns are satisfied.

7.3.4.5 Numerical illustration 2

Use the method of subtractions to find a starting solution for the following transportation problem as shown in Table 7.9.

Table 7.9: Given transportation problem.

8	9	20	4	4	6	**20**
2	1	1	1	1	1	**30**
2	8	19	7	3	14	**40**
2	6	6	4	11	8	**50**
2	15	5	12	8	7	**60**
30	**40**	**50**	**20**	**20**	**40**	

Row minima are given in Table 7.10.

Table 7.10: Row minima.

8	9	20	4	4	6	**20**
2	1	1	1	1	1	**30**
2	8	19	7	3	14	**40**
2	6	6	4	11	8	**50**
2	15	5	12	8	7	**60**
30	**40**	**50**	**20**	**20**	**40**	

Subtracting row minimum, we have Table 7.11.

Table 7.11: Subtracting row minima.

4	5	16	0	0	2	**20**
1	0	0	0	0	0	**30**
0	6	17	5	1	12	**40**
0	4	4	2	9	6	**50**
0	13	3	10	6	5	**60**
30	**40**	**50**	**20**	**20**	**40**	

The furthest zero is in second row and second column (nearest number in column is 4 units away and 0 units away in row = 4+0 = 4). Allocating in the selected cell (2,2), making the necessary adjustments, and removing the satisfied row we have Table 7.12.

Table 7.12: Allocating in the selected cell and removing satisfied row.

4	5	16	0	0	2	**20**
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
0	6	17	5	1	12	**40**
0	4	4	2	9	6	**50**
0	13	3	10	6	5	**60**
30	**10**	**50**	**20**	**20**	**40**	

There are zeros in all rows but there are no zeros in the second, third and sixth columns. Creating at least a zero in these columns we have Table 7.13.

Table 7.13: Creating at least a zero in all columns.

4	1	13	0	0	0	**20**
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
0	2	14	5	1	10	**40**
0	0	1	2	9	4	**50**
0	9	0	10	6	3	**60**
30	**10**	**50**	**20**	**20**	**40**	

The furthest zero is in first row and last column (nearest number in column is 3 units away and 0 units away in row =3+0=3). Allocating in the selected cell, making the necessary adjustments, and removing the satisfied row we have Table 7.14.

Table 7.14: Allocating in the selected cell and removing satisfied row.

~~4~~	~~1~~	~~13~~	~~0~~	~~0~~	~~0[20]~~	~~**20**~~
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
0	2	14	5	1	10	**40**
0	0	1	2	9	4	**50**
0	9	0	10	6	3	**60**
30	**10**	**50**	**20**	**20**	**20**	

There are zeros in all rows but there are no zeros in the fourth, fifth and sixth columns. Creating at least a zero in these columns we have Table 7.15.

Table 7.15: Creating at least a zero in all columns.

~~4~~	~~1~~	~~13~~	~~0~~	~~0~~	~~0[20]~~	~~**20**~~
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
0	2	14	3	0	7	**40**
0	0	1	0	8	1	**50**
0	9	0	8	5	0	**60**
30	**10**	**50**	**20**	**20**	**20**	

The furthest zero is in third row and fifth column (nearest number in column is 5 units away and 0 units away in row =5+0=5). Allocating in the selected cell, making the necessary adjustments, and removing the satisfied column we have Table 7.16.

Table 7.16: Allocating in the selected cell and removing satisfied column.

~~4~~	~~1~~	~~13~~	~~0~~	~~0~~	~~0[20]~~	~~**20**~~
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
0	2	14	3	~~0[20]~~	7	**20**
0	0	1	0	~~8~~	1	**50**
0	9	0	8	~~5~~	0	**60**
30	**10**	**50**	**20**	~~**20**~~	**20**	

There are zeros in all rows and all columns, and we can select another cell and allocate. The furthest zero is in fourth row and fourth column (nearest number in column is 3 units away and 0 units away in row =3+0=3). Allocating in the selected cell, making the necessary adjustments, and removing the satisfied column we have Table 7.17.

Table 7.17: Allocating in the selected cell and removing satisfied column.

~~4~~	~~1~~	~~13~~	~~0~~	~~0~~	~~0[20]~~	~~**20**~~
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
0	2	14	~~3~~	~~0[20]~~	7	**20**
0	0	1	~~0[20]~~	~~8~~	1	**30**
0	9	0	~~8~~	~~5~~	0	**60**
30	**10**	**50**	~~**20**~~	~~**20**~~	**20**	

There are zeros in all rows and all columns, and we can select another cell and allocate. The furthest zero is in fourth row and second column (nearest number in column is 2 units away and 0 units away in row =2+0=2). Allocating in the selected cell, making the necessary adjustments, and removing the satisfied column we have Table 7.18.

Table 7.18: Allocating in the selected cell and removing satisfied column.

~~4~~	~~1~~	~~13~~	~~0~~	~~0~~	~~0[20]~~	~~**20**~~
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
0	2	14	3	0[20]	7	**20**
0	0[10]	1	0[20]	8	1	**20**
0	9	0	8	5	0	**60**
30	**10**	**50**	**20**	**20**	**20**	

There are zeros in all rows and all columns, and we can select another cell and allocate. The furthest zero is in third row and first column (nearest number in column is 0 units away and 7 units away in row =0+7=7). Allocating in the selected cell, making the necessary adjustments, and removing the satisfied column we have Table 7.19.

Table 7.19: Allocating in the selected cell and removing satisfied column.

~~4~~	~~1~~	~~13~~	~~0~~	~~0~~	~~0[20]~~	~~**20**~~
~~1~~	~~0[30]~~	~~0~~	~~0~~	~~0~~	~~0~~	~~**30**~~
~~0[20]~~	~~2~~	~~14~~	~~3~~	~~0[20]~~	~~7~~	~~**20**~~
0	0[10]	1	0[20]	8	1	**20**
0	9	0	8	5	0	**60**
10	**10**	**50**	**20**	**20**	**20**	

There are zeros in all rows and all columns, and we can select another cell and allocate. There are three furthest zeros (in the case of a tie). First zero is in fourth row and first column (nearest number in column is 0 units away and 1 unit away in row =0+1=1). The other two furthest zero are in third and fifth columns. In this case we select the one in fourth row and first column. This zero has more big numbers in its row (i.e., two 1s). Allocating in the selected cell, making the necessary adjustments, and removing the satisfied column we have Table 7.20.

Table 7.20: Allocating in the selected cell and removing satisfied column.

4	1	13	0	0	0[20]	**20**
1	0[30]	0	0	0	0	**30**
0[20]	2	14	3	0[20]	7	**20**
0[10]	0[10]	1	0[20]	8	1	**10**
0	9	0	8	5	0	**60**
10	**10**	**50**	**20**	**20**	**20**	

There are no zeros in the fourth row but there are zeros in the columns. Creating at least a zero in the fourth row we have Table 7.21.

Table 7.21: Creating at least a zero in all rows.

4	1	13	0	0	0[20]	**20**
1	0[30]	0	0	0	0	**30**
0[20]	2	14	3	0[20]	7	**20**
0[10]	0[10]	0	0[20]	8	0	**10**
0	9	0	8	5	0	**60**
10	**10**	**50**	**20**	**20**	**20**	

All zeros are furthest zeros (i.e. a tie). In this case the zero that can make the largest allocation is selected as given in Table 7.22 and Table 7.23.

Table 7.22: Allocating in the selected cell and removing satisfied column.

4	1	13	0	0	0[20]	**20**
1	0[30]	0	0	0	0	**30**
0[20]	2	14	3	0[20]	7	**20**
0[10]	0[10]	0	0[20]	8	0	**10**
0	9	0[50]	8	5	0	**10**
10	**10**	**50**	**20**	**20**	**20**	

Table 7.23: Allocating in the selected cell and removing satisfied rows and only column.

4	1	13	0	0	0[20]	**20**
1	0[30]	0	0	0	0	**30**
0[20]	2	14	3	0[20]	7	**20**
0[10]	0[10]	0	0[20]	8	0[10]	**10**
0	9	0[50]	8	5	0[10]	**10**
10	**10**	**50**	**20**	**20**	**20**	

There are zeros in all rows and in the only column and we can allocate as given in Table 7.24.

Table 7.24: Starting solution obtained.

8	9	20	4	4	6[20]	**20**
2	1[30]	1	1	1	1	**30**
2[20]	8	19	7	3[20]	14	**40**
2[10]	6[10]	6	4[20]	11	8[10]	**50**
2	15	5[50]	12	8	7[10]	**60**
30	**40**	**50**	**20**	**20**	**40**	

$$\text{Total cost} = \begin{array}{l} 20(2)+2(10)+1(30)+6(10)+5(50) \\ +4(20)+3(20)+6(20)+8(10)+7(10) \end{array} = 810.$$

The optimal solution for this transportation problem is **780** and solution approximated by the methods of subtractions is closer than any other available approximating methods. Starting solution using VAM is given in Table 7.25, using the Least Cost method in Table 7.26 and using the North West Corner method in Table 7.27.

Table 7.25: VAM starting solution.

8	9	20	4[20]	4	6	**20**
2	1[30]	1	1	1	1	**30**
2	8[10]	19	7	3[20]	14[10]	**40**
2	6	6	4	11	8	**50**
2[30]	15	5[50]	12	8	7[30]	**60**
30	**40**	**50**	**20**	**20**	**40**	

$$\text{Total cost} = \begin{array}{l} 2(30)+1(30)+8(10)+5(50)+14[10] \\ +4(20)+3(20)+1(10)+7(30) \end{array} = 960.$$

Table 7.26: Least Cost method starting solution.

8	9	20	4[20]	4	6	**20**
2	1[30]	1	1	1	1	**30**
2[30]	8	19	7	3[10]	14	**40**
2	6[10]	6	4	11[10]	8[30]	**50**
2	15	5[50]	12	8	7[10]	**60**
30	**40**	**50**	**20**	**20**	**40**	

$$\text{Total cost} = \begin{array}{l} 2(30)+1(30)+6(10)+5(50)+4(20) \\ +3(20)+11(10)+8(30)+7(10) \end{array} = 930$$

Table 7.27: North West Corner method starting solution.

8[20]	9	20	4	4	6	**20**
2[10]	1[20]	1	1	1	1	**30**
2	8[20]	19[20]	7	3	14	**40**
2	6	6[30]	4[20]	11	8	**50**
2	15	5	12	8[20]	7[40]	**60**
30	**40**	**50**	**20**	**20**	**40**	

$$\text{Total cost} = \begin{matrix} 8(20)+2(10)+1(20)+8(20)+19(20) \\ +6(30)+4(20)+8(20)+7(10) \end{matrix} = 1140$$

In conclusion, one can state that the method of subtraction for transportation models performs much better for a starting solution. An experiment was created on 20 randomly generated problems and the method of subtraction performed better compared to VAG and Least Cost method on all those 20 problems. The method discussed above is simple and is made up of only subtraction operations. The only obvious weakness is number of iterations required to obtain this better starting solution.

7.4 The See-Saw algorithm for a general transportation problem

In this section, we develop the see-saw approach for the transportation problem. Since every transportation problem can be viewed as a special assignment problem, therefore the method for the assignment problem can also be used for solving the general transportation problem. Slight modifications are required, which are presented in this section. It may be noted that unlike the assignment problem where there is exactly one unit of allocation exists in every column and row, the transportation problem may have more than one allocation in a column or row. The See-Saw moves in this case are done from more than one cells in one column to more than one cells in the other column. As stated in earlier, See-Saw moves are very simple and can be calculated at the same time by parallel processors.

7.4.1 A General transportation model

A transportation model in general can be expressed as given in Table 7.28

Table 7.28: Transportation model in general.

					Supply
c_{11}	c_{12}	c_{13}	$\dots$	c_{1n}	s_1
c_{21}	c_{22}	c_{23}	$\dots$	c_{2n}	s_2
c_{31}	c_{32}	c_{33}	$\dots$	c_{3n}	s_3
	$\dots$		$\dots$	$\dots$	
c_{m1}	c_{m2}	c_{m3}	$\dots$	c_{mn}	s_m

Where c_{ij} is the cost of transporting a unit from i to j. We are assuming this is a balanced transportation problem, i.e. number of total supply is equal to the total demand (D_T) i.e. $\sum_i s_i = \sum_i d_j = D_T$. The objective is to minimize the total transportation cost.

In mathematical form the transportation problem can be represented as given in (7.11).

$$\begin{aligned} &\text{Minimize } Z_{Tra} = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}, \\ &\sum_{j}^{m} x_{ij} = s_i, ,i = 1, 2, \dots, n. \\ &\sum_{i}^{n} x_{ji} = d_j, j = 1, 2, \dots, n. \end{aligned} \tag{7.11}$$

$x_{ij} \geq 0$. and integer.

Like the assignment model the transportation model is also degenerate.

7.4.2 The assignment-transportation model relationship

A general transportation problem can be easily converted into an assignment problem by splitting all rows so that supply for each is exactly one as given in Table 7.29. Similarly, for the columns are repeated so that demand for each column is also one. The transportation problem as an assignment problem is shown in Table 7.30.

Splitting the columns so that each column has exactly a demand of one we have Table 7.30. Note that Table 7.30 is the transportation problem in an assignment form.

Table 7.30 shows that any balanced transportation problem can be expressed as an assignment problem.

Table 7.29: Splitting the rows of transportation problem.

				Supply	
c_{11}	c_{12}	$\cdots$	c_{1n}	1	
c_{11}	c_{12}	$\cdots$	c_{1n}	1	s_1 times
				$\cdots$	
c_{11}	c_{12}	$\cdots$	c_{1n}	1	
c_{21}	c_{22}	$\cdots$	c_{2n}	1	
c_{21}	c_{22}	$\cdots$	c_{2n}	1	s_2 times
				$\cdots$	
c_{21}	c_{22}	$\cdots$	c_{2n}	1	
$\cdots$		$\cdots$	$\cdots$		
c_{m1}	c_{m2}	$\cdots$	c_{mn}	1	
c_{m1}	c_{m2}	$\cdots$	c_{mn}	1	s_m times
				$\cdots$	
c_{m1}	c_{m2}	$\cdots$	c_{mn}	1	
d_1	d_2	$\cdots$	d_n	D_T	

A starting solution is required for the See-Saw procedure and the more accurate the starting solution is, a fewer See-Saw moves will be necessary to reach the optimal solution. In theory, we have three approaches, which are:

a) Method of subtractions,
b) Least cost method and
c) Vogel's approximation method.

As stated earlier, method of subtraction, seems to the best choice.

7.4.3 See-Saw rule for the transportation model

The See-Saw rule for the general transportation problem is like the one for the assignment problem. The only difference is that more than one allocation is possible in every row and column unlike in the assignment problem where there is exactly one allocation.

In transportation See-Saw moves, one column is pivoted and then paired with the other movement is done in one column such that when one goes up in the pivoted

Table 7.30: Transportation problem in an assignment form.

← d_1 times →				← d_2 times →					← d_n times →				Supply	
c_{11}	c_{11}	$\cdots$	c_{11}	c_{12}	c_{12}	$\cdots$	c_{12}	$\cdots$	c_{1n}	c_{1n}	$\cdots$	c_{1n}	1	
c_{11}	c_{11}	$\cdots$	c_{11}	c_{12}	c_{12}	$\cdots$	c_{12}	$\cdots$	c_{1n}	c_{1n}	$\cdots$	c_{1n}	1	s_1 times
									$\cdots$					
c_{11}	c_{11}	$\cdots$	c_{11}	c_{12}	c_{12}	$\cdots$	c_{12}	$\cdots$	c_{1n}	c_{1n}	$\cdots$	c_{1n}	1	
c_{21}	c_{21}	$\cdots$	c_{21}	c_{22}	c_{22}	$\cdots$	c_{22}	$\cdots$	c_{2n}	c_{2n}	$\cdots$	c_{2n}	1	
c_{21}	c_{21}	$\cdots$	c_{21}	c_{22}	c_{22}	$\cdots$	c_{22}	$\cdots$	c_{2n}	c_{2n}	$\cdots$	c_{2n}	1	s_2 times
									$\cdots$					
c_{21}	c_{21}	$\cdots$	c_{21}	c_{22}	c_{22}	$\cdots$	c_{22}	$\cdots$	c_{2n}	c_{2n}	$\cdots$	c_{2n}	1	
				$\cdots$		$\cdots$	$\cdots$							
c_{m1}	c_{m1}	$\cdots$	c_{m1}	c_{m2}	c_{m2}	$\cdots$	c_{m2}	$\cdots$	c_{mn}	c_{mn}	$\cdots$	c_{mn}	1	
c_{m1}	c_{m1}	$\cdots$	c_{m1}	c_{m2}	c_{m2}	$\cdots$	c_{m2}	$\cdots$	c_{mn}	c_{mn}	$\cdots$	c_{mn}	1	s_m times
									$\cdots$					
c_{m1}	c_{m1}	$\cdots$	c_{m1}	c_{m2}	c_{m2}	$\cdots$	c_{m2}	$\cdots$	c_{mn}	c_{mn}	$\cdots$	c_{mn}	1	
1	1		1	1	1		1		1	1		1	(Demand)	

column then the other is forced to go down in the paired column and vice versa. A column may have more than one allocation and all these basic values must be subjected to See-Saw moves. Similarly, the paired column may have more than one basic allocation and all these must be subjected to See-Saw moves, and most profitable move selected.

7.4.4 Initial position before the See-Saw move

The initial or starting solution for the See-Saw rule has more than one allocation in the columns as given in Table 7.31.

The current combined cost (ccc_{ij}) for the two paired columns is given in (7.12).

$$ccc_{ij} = c_{1i}\gamma_{1i} + c_{2i}\gamma_{2i} + \ldots + c_{(n-1)i}\gamma_{(n-1)i} + c_{2j}\gamma_{2j} + \ldots + c_{(n-1)j}\gamma_{(n-1)j} + c_{nj}\gamma_{nj.} \tag{7.12}$$

Table 7.31: Initial position.

$\downarrow$ $c_{1i}[\gamma_{1i}]$	…	$\uparrow$ c_{1j}
$c_{2i}[\gamma_{2i}]$	…	$c_{2j}[\gamma_{2j}]$
…	…	…
$c_{(n-1)i}[\gamma_{(n-1)i}]$	…	$c_{(n-1)j}[\gamma_{(n-1)j}]$
c_{ni}	…	$c_{nj}[\gamma_{nj}]$

The new cost after the See-saw move is calculated from the allocation given in Table 7.32.

Table 7.32: New position after the See-Saw move.

c_{1i}	…	$c_{1j}[\gamma_{1i}]$
$c_{2i}[\gamma_{2i}+\gamma_{1i}]$	…	$c_{2j}[\gamma_{2j}-\gamma_{1i}]$
…	…	…
$c_{(n-1)i}[\gamma_{(n-1)i}]$	…	$c_{(n-1)j}[\gamma_{(n-1)j}]$
c_{ni}	…	$c_{nj}[\gamma_{nj}]$

The new combined cost (ncc_{ij}) is given in (7.13).

$$ncc_{ij} = c_{2i}(\gamma_{1i}+\gamma_{2i}) + \ldots + c_{(n-1)i}\gamma_{(n-1)i} + c_{2j}(\gamma_{2j}-\gamma_{1i}) + \ldots + c_{(n-1)j}\gamma_{(n-1)j} + c_{nj}\gamma_{nj}. \tag{7.13}$$

See-Saw movement is possible and profitable if the new combined cost (ncc_{ij}) is less than or equal to current combined cost (ccc_{ij}) as given in (7.14).

$$ccc_{ij} \leq ncc_{ij}. \tag{7.14}$$

Pairing of columns

Just like the See-Saw move for the assignment model pairing of columns is done from the left to the right i.e., select first column and pair it with all the (n-1) on the right and select the best see-saw move. The only version for assignment model is that all allocations in the same column are used for the see-saw moves.

7.4.5 See-Saw algorithm for the general transportation model

The See-Saw algorithm for general transportation problem is summarized as follows.

Step 1: Determine a stating solution for the transportation problem

Step 2: Pair the columns starting from the left. Use allocations in the column for the See-Saw moves and select the move that gives $ccc_{ij} \leq ncc_{ij}$. Repeat procedure for all columns from left to right until there is no more See-Saw move possible and go Step 3.

Step 3: Current solution is optimal.

7.4.6 Numerical illustration of transportation model

Use the See-Saw rule to solve the transportation problem as given in Table 7.33.

Table 7.33: Given example.

					Supply
	11	3	21	12	150
	13	8	10	21	250
	5	15	17	19	100
Demand	50	150	150	150	

Starting solution by using the method of subtractions

Row minima is given in Table 7.34 and after subtraction, we have Table 7.35. Similarly column give Tables 7.36 and 7.37.

Table 7.34: Row minima.

					Supply
	11	**3**	21	12	150
	13	**8**	10	21	250
	5	15	17	19	100
Demand	50	150	150	150	

Table 7.35: Subtracting row minima.

					Supply
	8	**0**	19	9	150
	5	**0**	2	13	250
	0	10	12	14	100
Demand	50	150	150	150	

Table 7.36: Column minima.

					Supply
	8	**0**	19	**9**	150
	5	**0**	**2**	13	250
	0	10	12	14	100
Demand	50	150	150	150	

Table 7.37: Subtracting column minima.

					Supply
	8	**0**	17	**0**	150
	5	**0**	**0**	4	250
	0	10	10	5	100
Demand	50	150	150	150	

From Table 7.37, we obtain differences of the two minimum as shown in Table 7.38.

Table 7.38: Determining row (α_j) & column (β_i) differences.

					Supply	β_i
	8	**0**	17	**0**	150	8
	5	**0**	**0**	4	250	0
	0	10	10	5	100	5
Demand	50	150	150	150		
α_j	5	0	10	4		

The furthest zero has a total distance of $(\beta_3 + \alpha_2) = 10 + 0 = 10$. Allocating this zero we have Table 7.39.

Table 7.39: Subtracting column minima.

					Supply
	8	**0**	17	**0**	150
	5	**0**	**0[150]**	4	100
	0	10	10	5	100
Demand	50	150	150	150	

The third column is now satisfied. Repeating the procedure in method of subtractions we have Table 7.40.

Table 7.40: Determining new row (α_j) & column (β_i) differences.

					Supply	β_i
	8	**0**	17	**0**	150	0
	5	**0**	**0**	4	250	4
	0	10	10	5	100	5
Demand	50	150	150	150		
α_j	5	0		4		

The furthest zero has a total distance of $(\beta_1 + \alpha_3) = 5 + 5 = 10$. Allocating this zero we have Table 7.41, Table 7.42 and Table 7.43.

Table 7.41: Allocating the furthest zero and identifying new row & column minima.

					Supply
	8	**0**	17	**0**	150
	5	**0**	**0[150]**	4	100
	0[50]	10	10	5	50
Demand	50	150	150	150	

Table 7.42: Subtracting row minima.

					Supply
	8	**0**	17	**0**	150
	5	**0**	**0[150]**	4	100
	0[50]	5	10	**0**	50
Demand	50	150	150	150	

Table 7.43: Determining new row (α_j) & column (β_i) differences.

					Supply	β_i
	8	**0**	17	**0**	150	0
	5	**0**	**0**	4	250	4
	0[50]	5	10	**0**	100	5
Demand	50	150	150	150		
α_j		0		0		

The furthest zero has a total distance of $(\beta_4 + \alpha_3) = 0 + 4 = 4$. Allocating this zero we have Table 7.44, and Table 7.45.

Table 7.44: Allocating the furthest zero.

					Supply
	8	**0**	17	**0**	150
	5	**0**	**0[150]**	4	100
	0[50]	5	10	0[50]	50
Demand	50	150	150	100	

Table 7.45: Determining new row (α_j) & column (β_i) differences.

					Supply	β_i
	8	**0**	17	**0**	150	0
	5	**0**	**0**	4	250	4
	0[50]	5	10	**0[50]**	50	5
Demand	50	150	150	100		
α_j		0		0		

The furthest zero has a total distance of $(\beta_4 + \alpha_1) = 0 + 4 = 4$. Allocating this zero we have Table 7.46.

Table 7.46: Allocating the furthest zero.

					Supply
	8	**0**	17	**0[100]**	50
	5	**0**	**0[150]**	4	100
	0[50]	5	10	0[50]	50
Demand	50	150	150	100	

Automatically the second column is allocated as given in Table 7.47.

Table 7.47: Allocating in the second column.

					Supply
	8	**0[50]**	17	**0[100]**	50
	5	**0[100**	**0[150]**	4	100
	0[50]	5	10	0[50]	50
Demand	50	150	150	100	

The total cost is given as 250+100+800+2100+1200+950=5400.

Using the See-Saw algorithm to move to optimality

Iterative steps of the algorithm are shown in successive tables, as given by Table 7.48 to Table 7.58 below.

Table 7.48: Starting solution.

					Supply
	8	**0[50]**	17	**0[100]**	150
	5	**0[100]**	**0[150]**	4	250
	0[50]	5	10	**0[50]**	100
Demand	50	150	150	100	

---Pivoting Column 1--

Table 7.49: See-Saw Move 1.

					Supply
	11	**3[50]**	21	**12[100]**	150
	13	**8[100]**	**10[150]**	21	250
	5[50]	15	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 1:$5(50) + 3(50) + 8(100) - \{11(50) + 15(50) + 8(100)\} = -900 < 0$. The See-Saw move is not profitable.

Table 7.50: See-Saw Move 2.

					Supply
	11	**3[50]**	21	**12[100]**	150
	↑ 13	**8[100]**	**10[150]**	21	250
	5[50]	↓ 15	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 2:$5(50)+8(100)-\{13(50)+15(50)+8(50)\}=-750<0$. The See-Saw move is not profitable.

Table 7.51: See-Saw Move 3.

					Supply
	11	**3[50]**	21	**12[100]**	150
	↑ 13	**8[100]**	**10[150]**	21	250
	5[50]	15	↓ 17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 3:$5(50)+10(150)-\{13(50)+100(100)+50(17)\}=-750<0$. The See-Saw move is not profitable.

Table 7.52: See-Saw Move 4.

					Supply
	↑ 11	**3[50]**	21	**12[100]**	150
	13	**8[100]**	**10[150]**	21	250
	5[50]	15	17	↓ **19[50]**	100
Demand	50	150	150	100	

See-Saw Move 4:$5(50)+12(100)+19(50)-\{11(50)+12(50)+19(100)\}=-650<0$. The See-Saw move is not profitable

---Pivoting Column 2---

Table 7.53: See-Saw Move 5.

					Supply
	11	**3[50]**	↑ 21	**12[100]**	150
	13	↓ **8[100]**	**10[150]**	21	250
	5[50]	15	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 5:$8(100)+3(50)+10(150)-\{8(150)+21(50)+10(100)\}=-800<0$. The See-Saw move is not profitable.

Table 7.54: See-Saw Move 6.

					Supply
	11	**3[50]** ↑	21	**12[100]**	150
	13	**8[100]**	**10[150]**	21 ↓	250
	5[50]	15	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 6:3(50)+8(100)+12(100)+19(50)−{3(150)+21(100)+19(50)=−400<0. The See-Saw move is not profitable.

Table 7.55: See-Saw Move 7.

					Supply
	11	**3[50]**	21	**12[100]**	150
	13	**8[100]**	**10[150]**	21 ↑	250
	5[50]	15 ↓	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 7:3(50)+8(100)+12(100)+19(50)−{3(50)+8(50)+15(50)+12(100)+21(50)}= −450<0. The See-Saw move is not profitable.

Table 7.56: See-Saw Move 8.

					Supply
	11	**3[50]**	21	↑**12[100]**	150
	13	**8[100]**	**10[150]**	21	250
	5[50]	↓ 15	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 8:3(50) + 8(100) + 12(100) + 19(50) − {8(100) + 15(50) + 12(150)} = − 250 < 0. The See-Saw move is not profitable.

--Pivoting Column 3--

Table 7.57: See-Saw Move 9.

					Supply
	11	**3[50]**	↑ 21	**12[100]**	150
	13	**8[100]**	**10[150]**	↓ 21	250
	5[50]	15	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 9:10(150) + 12(100) + 19(50) − {21(100) + 10(50) + 21(100) + 19(50)} = − 2000 < 0. The See-Saw move is not profitable.

Table 7.58: See-Saw Move 10.

					Supply
	11	**3[50]**	21	**12[100]**	150
	13	**8[100]**	**10[150]** ↓	↑ 21	250
	5[50]	15	17	**19[50]**	100
Demand	50	150	150	100	

See-Saw Move 10:$10(150)+12(100)+19(50)-\{10(100)+21(50)+12(100)+50(17)+19(50)\}=-1400<0$. The See-Saw move is not profitable.

The current solution is optimal since there is no See - Saw move possible.

The See-Saw algorithm use a current solution to examine the possibility of moving into an unoccupied cell.

Possible See-Saw move: $ccc_{ij} \leq ncc_{ij}$.

Impossible See-Saw move: $ccc_{ij} > ncc_{ij}$.

The See-Saw algorithm examines <u>all</u> unoccupied cells.

Algorithm terminates if See-Saw moves are impossible to all unoccupied cell.

Impossibility to move implies current solution cannot be improved and is optimal.

In conclusion, the See-Saw is a heuristic rule and moves are simple and can be done independently. The See-Saw moves for all the paired columns can be calculated independently by use of massively parallel processing. This is not possible for the available transportation simplex, Hungarian and the network simplex methods. The transportation model has application in the military, see Kline et al. (2019) and scheduling problems, see Lan et al. (2019).

7.5 Determination of k^{th} ($k \geq 2$) *ranked optimal solution*

Determination of the best or an optimal solution for a given mathematical model is relatively easy as the existing methods have in them some inbuilt optimality conditions, and when these conditions are satisfied at a particular point of search; that point is declared as the optimal solution and the search process ends. However, determination of the $k^{th}best, k \geq 2$ is mathematically a difficult problem as ranked optimality or the $k^{th}best, k \geq 2$ is linked to previous $(k-1)$ ranked optimal solutions. For example, the 2^{nd} best solution may be defined as the best solution among all possible solutions, excluding the best one. In general, the $(k+1)^{th}$ best solution will be the best solution after excluding 1, 2, . . ., k^{th} best solutions and this seemingly simple variation makes the problem difficult and computationally demanding. In context of an assignment problem, many attempts have been made by various researchers. Here we just mention the attempts by Murty (1968) and Kumar et al. (2020). In this section, we discuss, a new approach and apply this new approach to a couple of numerical illustrations from Murty (1968) and Kumar et al. (2020) and compare with the results obtained by them.

7.5.1 Murthy's (1968) approach

The main idea behind Murthy's approach can be explained by considering a *nXn* assignment problem and assume the optimal string of assignment has been determined by the Hungarian method of assignment. Murty has denoted this optimal string by: $a(1) = \{(i_1, j_1), \cdots, (i_r, j_r), \cdots, (i_n, j_n)\}$. It means job j_r is assigned to the person i_r, $r = 1, 2, \ldots (n-1), n$. Murthy solves (n-1) assignment problems, as follows:

$$M_1 = \left\{\left(\overline{i_1, j_1}\right)\right\}$$

$$M_2 = \left\{(i_1, j_1), \left(\overline{i_2, j_2}\right)\right\}$$

$$\cdots$$

$$M_r = \left\{(i_1, j_1), \ldots, (i_{r-1}, j_{r-1}), \left(\overline{i_r, j_r}\right)\right\}, r = 1, 2, \ldots (n-1) \tag{7.15}$$

In other words, from the given problem, one can create (n-1) more assignment problems, where the problem M_1 is the given problem with the element $(i_1,\ j_1) = \infty$, and the problem is solved again by the Hungarian method of assignment. Similarly, the problem is solved again with an assignment in row 1 as was in the optimal string, i.e., the location $(i_1,\ j_1)$ and the next element from the optimal string $(i_2,\ j_2) = \infty$. This process is repeated on each $M_r,\ r = 1, 2, \ldots, (n-1)$.

This approach is implementing the idea that the 2nd best will differ from the best with respect to at least one assignment, which is achieved by setting that cost of assignment equal to ∞

Thus Murty (1968) continues to use the Hungarian method of assignment, and minimum from these (n-1) problems gives the required 2nd best.

Let us reconsider the illustration used by Murthy (1968), which is shown in Table 7.59.

The optimal assignment for the problem in Table 7.59 is given by the following assignment combinations:

$$a(1) = \{(1,9), (2,7), (3,3), (4,8), (5,6), (6,4), (7,10), (8,1), (9,5), (10,2)\}, \text{with total cost} = 0 \tag{7.16}$$

Since the 2nd best will differ from the best solution in at least one assignment, therefore, for a *nXn* assignment model, Murty (1968) solves (n-1) conditional assignment problems. For the above illustration, he solves 9 conditional assignment problems and obtained optimal cost as follows:

Table 7.59: Assignment data for the 10X10 problem.

R\C	1	2	3	4	5	6	7	8	9	10
1	7	51	52	87	38	60	74	66	0	20
2	50	12	0	64	8	53	0	46	76	42
3	27	77	0	18	22	48	44	13	0	57
4	62	0	3	8	5	6	14	0	26	39
5	0	97	0	5	13	0	41	31	62	48
6	79	68	0	0	15	12	17	47	35	43
7	76	99	48	27	34	0	0	0	28	0
8	0	20	9	27	46	15	84	19	3	24
9	56	10	45	39	0	93	67	79	19	38
10	27	0	39	53	46	24	69	46	23	1

$$M_1 = \{\overline{(1,9)]}\}, M_1 = 10$$

$$M_2 = \left\{(1,9), \overline{(2,7)}\right\}, Cost\ M_2 = 14$$

$$M_3 = \left\{(1,9), (2,7), \left(\overline{3.3}\right)\right\}, Cost\ M_3 = 14$$

$$M_4 = (1,9), (2,7), (3,3), \left(\overline{4.8}\right), Cost\ M_4 = \ 1$$

$$M_5 = (1,9), (2,7), (3,3), (4,8), \left(\overline{5.6}\right), Cost\ M_5 = 12$$

$$M_6 = \{(1,9), (2,7), (3,3), (4,8), (5,6), \left(\overline{6.4}\right), Cost\ M_6 = 53$$

$$M_7 = \{(1,9), (2,7), (3,3), (4,8), (5,6), (6,4), \left(\overline{7.10}\right), Cost\ M_7 = 45$$

$$M_8 = \{(1,9), (2,7), (3,3), (4,8), (5,6), (6,4), (7,10), \left(\overline{8.1}\right), Cost\ M_8 = 47$$

$$M_9 = \{(1,9), (2,7), (3,3), (4,8), (5,6), (6,4), (7,10), (8,1), \left(\overline{9.5}\right), CostM_9 = 56 \qquad (7.17)$$

From (7.17), he concludes that the 2nd best solution is given by M_4 with total cost equal to 1. The assignment for the 2nd best solution were obtained as:

$$a(2) = \{(1,9), (2,7), (3,3), (4,2), (5,6), (6,4), (7,8), (8,1), (9,5), (10,10)\} \qquad (7.18)$$

7.5.2 Minimal cost assignment approach for the ranked solution

The minimum cost approach is an alternative approach, following are the steps to find the ranked solutions.

Step 1: Consider a *nXn* assignment problem. Find the optimal solution by the Hungarian method of assignment with total cost equal to zero for the modified cost in the usual Hungarian approach.

Step 2: Locate the minimum nonnegative coat cell in the modified cost matrix. Let this cell be denoted by $(i,\ j)$.

Step 3: Make an allocation in the cell $(i,\ j)$, delete the row *i and column j*. Find the optimal allocation for the remaining $(n-1)X(n-1)$ assignment problem.

Step 4: If the total cost for this $(n-1)X(n-1)$ is equal to zero, then we have obtained the ranked solution, else list the result and return to Step 2.

Let us reconsider the numerical illustration by Murty and find the 2^{nd} best solution by this approach. The best solution has been identified in (7.16) with total cost zero.

As per the step 2, we identify a minimum non-zero cost cell from Table 7.59. This element is 1 situated at the cell (10,10). Thus, we make a conditional allocation in the cell (10, 10), and therefore delete the row 10 and, also delete the column 10. The new matrix will be as given in Table 7.60.

Table 7.60: Modified Table after deleting row 10 and column 10.

R\C	1	2	3	4	5	6	7	8	9
1	7	51	52	87	38	60	74	66	0
2	50	12	0	64	8	53	0	46	76
3	27	77	0	18	22	48	44	13	0
4	62	0	3	8	5	6	14	0	26
5	0	97	0	5	13	0	41	31	62
6	79	68	0	0	15	12	17	47	35
7	76	99	48	27	34	0	0	0	28
8	0	20	9	27	46	15	84	19	3
9	56	10	45	39	0	93	67	79	19

It may be noted that the cell (10,10) together with the allocations shown in red in Table 7.60 constitute a feasible solution to the given assignment model and it is the 2^{nd} best solution as it is the best after excluding the optimal solution shown by the allocations in (7.16).

Illustration 2 taken from Kumar et al.(2020)

Consider the assignment problem given by Table 7.61. Optimal cost is 9 and the optimal string is {(1,2), (2,4), (3,1), (4,3)}.

Table 7.61: Assignment cost matrix considered by Kumar et al. (2020).

$$C_{ij} = \begin{vmatrix} 5 & 2 & 9 & 6 \\ 1 & 10 & 8 & 2 \\ 2 & 3 & 8 & 9 \\ 5 & 7 & 3 & 1 \end{vmatrix} \equiv \begin{vmatrix} 3 & 0 & 4 & 3 \\ 0 & 9 & 4 & 0 \\ 0 & 1 & 3 & 6 \\ 5 & 7 & 0 & 0 \end{vmatrix}$$

Once again, we select a non-zero smallest element which is 1 in the cell (3,2). Deleting row 3 and column 2, we get a reduced cost matrix as shown in Table 7.62.

Table 7.62: Assignment matrix after the conditional allocation.

$$C_{ij} \equiv \begin{vmatrix} 3 & \cancel{0} & 4 & 3 \\ 0 & \cancel{9} & 4 & 0 \\ \cancel{0} & \cancel{1} & \cancel{3} & \cancel{6} \\ 5 & \cancel{7} & 0 & 0 \end{vmatrix} \equiv \begin{vmatrix} 3 & 4 & 3 \\ 0 & 4 & 0 \\ 5 & 0 & 0 \end{vmatrix}$$

Feasible solution at 0 cost locations is possible only after subtracting 3 from the elements in row 1 and possible feasible solution for given problem will be as shown in the string {(1,1), (2,4), (3,2), and (4,3)}. Its total cost will be given by (9+1+3) =13. This solution is not accepted as the second-best solution as increase in total cost is 4 units, whereas the reduced cost matrix (See Table 7.61) has elements less than 4. Therefore, we must check with respect to the elements which are less than 4. These elements in addition to the element in the cell (3,2) are 3 in cells (1,1), (1,4) and (3,3). In fact, if we make an allocation in the cell (3,3), we do end up getting a feasible solution given by {(1,2), (2,1), (3,3) and (4,4) with total cost 12, which is now the 2nd best solution. There are more than one third best solutions with total cost equal to 13.

7.6 Concluding remarks

In this chapter we have introduced a new approach for solving an assignment and transportation problem, which is suitable for parallel computing. Earlier approaches are not suited to parallel computing ideas. We have emphasized the importance of a starting solution for a transportation model and presented a method that gives a better starting solution compared to existing approaches. Finally, we have also identified some methods for determination of the $k^{th} best, k \geq 2$ for an assignment problem. A need exists for developing methods for the $k^{th} best, k \geq 2$ for other optimization models.

References

Aboli H. Patil, Parikshit N. Mahalle. (2020). Trends and Challenges in Measuring Performance of Reviewer Paper Assignment, Procedia Computer Science, Volume 171, 709–718.

Aramuthakannan S, Kandasamy PR. (2013). Revised distribution method of finding optimal solution for transportation problems. IOSR Journal of Mathematics (IOSR-JM). 4(5):39–42.

Charnes Cooper. (1954). The stepping-stone method for explaining linear programming. Calculation in transportation problems. Management Science. 1954;1(1):49–69.

Date, Ketan, Nagi, Rakesh. (2016). GPU-accelerated Hungarian algorithms for the Linear Assignment Problem. Parallel Computing. 57, 52–72.

Edmonds, Jack, Karp, Richard. (1972). Theoretical Improvement in Algorithmic Efficiency for Network Flow Problems. Journal of the ACM. 19. 248–264.

Jenő Egerváry, (1931). [=Eugene], Matrixok kombinatorius tulajdons´agair´ol, [in Hungarian: On combinatorial properties of matrices] Matematikai ´es Fizikai Lapok **38**, 16–28.

Hillier, F.S., Lieberman, G.J. (2015), Introduction to Operations Research, ISBN 13:79812 59545962.

Kline, A., Ahner, D., Hill, R. (2019). The Weapon-Target Assignment Problem. Computers and Operations Research **105**, 226–236.

Kőnig, D. 1931. Graphok ´es matrixok, [in Hungarian: Graphs and matrices], Matematikai ´es Fizikai Lapok **38**, 116–119.

Kuhn, H.W. (1955). The Hungarian Method for the assignment problem, Naval Research Logistic Quarterly **2**, 83–97.

Kumar, S., Ncube, O., and Munapo, E. (2003). Tsoro and Hungarian approaches: A hybrid algorithm for an assignment problem. Asia-Pacific Journal of Operational Research. 20. Pp 41–49.

Kumar, S., Al-Hasani, A., Al-Rabeeah, M. and Ebehard, A. (2020). A random search method for finding ‘ k≥2’ number of ranked optimal solution to an assignment problem, Journal of Physics: Conference series 1490 (2020)012063, doi:10.1088/1742-6596/1490/1/012063, pp 1–13.

Lan S., Fan, W., Liu, T., Yang, S. (2019). A hybrid SCA-VNS meta-heuristic based on Iterated Hungarian algorithm for physicians and medical staff scheduling problem in outpatient department of large hospitals with multiple branches. Applied Soft Computing 85.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. Journal of the Society for Industrial and Applied Mathematics **5**, 32–38.

Murty, K.G. (1968). An algorithm for ranking all the assignments in order of increasing cost, Operations Research, 16.3, pp 682–687.

Niv A., Maccaig M., Sergeev, S. (2020). Optimal assignments with supervisions. Linear Algebra and its Applications **595**. 72–100.

Oncan, T., Suvak, Z., Akyuz, M.H., Altinel, I.K., (2019). Assignment problem with conflicts Computers and Operations Research 111, 214–229.

Pandian, P., Natarajan, G. (2010). A new method for finding an optimal solution for transportation problems. International J. of Math. Sci. and Engg. Appls 4. 59–65.

Quddoos A., Javaid S., Khalid M.M. (2012). A new method for finding an optimal solution for transportation problems. International Journal on Computer Science & Engineering. 4(7).

Quddoos A., Rabbani, Q. (2019). Modified Hungarian method for unbalanced assignment problem with multiple jobs. Applied Mathematics and Computation **361**. 493–498.

Soomro A.S., Tularam, G.A., Bhayo, G.M. (2014). A comparative study of initial basic feasible solution methods for transportation problem. Mathematical theory and Modelling. 2224–5804.

Soomro, A.S., Jamali, S., Shaikh, M.M. (2017). On non-optimality of direct exponential approach method for solution of transportation problems, Sindh Univ. Res. Jour. (Sci. Ser.). 49(1):183–188.

Taha, H.A. (2017). Operations Research: An Introduction, Pearson Educators, 10th Edition.

Tomizawa, N. (1971). On some techniques useful for solution of transportation network problems. Networks, 1, 173–194.

Winston, W.L. (2004). Operations Research Applications and Algorithms, Duxbury Press, 4th Edition

Zhang, Ren-Qian, Wang, M. and Pan, X. (2019). New model of the storage location assignment problem considering demand correlation pattern. Computers and Industrial Engineering 129, 210–219.

Chapter 8
The travelling salesman problem: Sub-tour elimination approaches and algorithms

Abstract: This chapter presents a few new approaches to the travelling salesman problem by generating sub-tour elimination cuts and adding these to a binary LP formulation of the TSP. The binary LP is converted as a convex quadratic problem which is solved efficiently by interior point algorithms. The other approach is to deal with the TSP network and convert that into transshipment sub-problems. These separate transshipment problems are then combined to come up with a master formulation for the TSP problem. The proposed formulation has the advantage that in most cases the relaxed LP gives an optimal integer solution.

Keywords: Traveling salesman problem, Sub-tour elimination cuts, Formulation of the TSP as a binary LP, Convex quadratic problem, Interior point algorithm, **L**inear integer model, Totally unimodular and transshipment

8.1 Introduction

A travelling salesman problem (TSP) can be defined as a problem of visiting *n* centers in such a way that:
- each center is visited only once,
- after visiting all centers, finally return to the original center (i.e. the starting center) and
- the total distance covered is the shortest of all the possible routes available.

The TSP model has so many applications and it is for this reason, it has remained as an active research area. Some of the areas of practical applications are vehicle routing, crystallography, circuit board drilling, order collection in warehouses etc.

The traveling salesman problem (*TSP*) was believed to be a difficult problem in combinatorial optimization until recently, see Munapo (2020). Researchers were unaware of any consistent and efficient general-purpose algorithm for this *NP* hard problem. The TSP has been extensively discussed by Finke et al. (1983), Gavish and Graves (1978), Lenstra and Rinnoonkan (1975), Miller (1960), Schrijver (1998), Vajda (1961), and Wong (1980). Several variants of the *TSP* have originated from various real-life problems, see Applegate et al. (2007), and Gutine and Punnen (2006). Large scale TSP has been studied by Bland and Shallcross (1989). Special class of TSP have been studied by Claus (1984), Fox et al. (1980), Grostschel et al. (1991), Van Dal (1992) and Plante et al. (1987). Munapo et al. (2016) developed a minimum spanning tree with node

https://doi.org/10.1515/9783110703023-008

index ≤2. Kumar et al. obtained a TSP using the special minimum spanning tree, see Kumar et al. (2016, 2017, 2020).

In this chapter the *TSP* is studied to come up with a linear binary integer formulation. The binary linear integer formulation has a special feature that it can be converted into a convex quadratic problem which can then be solved efficiently by interior point algorithms. This makes the *TSP* model easier to solve than what was widely believed to be. This chapter has been organized in 6 sections. The TSP is a binary linear integer program, which has been converted into a convex quadratic program and solved in polynomial time. This discussion has been covered in sections 8.2, 8.3 and 8.4. The TSP network has been reconsidered as a transshipment model in sections 8.5 and 8.6. Finally, the chapter is concluded in section 8.7.

8.2 Binary formulation of the TSP

The TSP model is represented diagrammatically as shown in Figure 8.1 below.

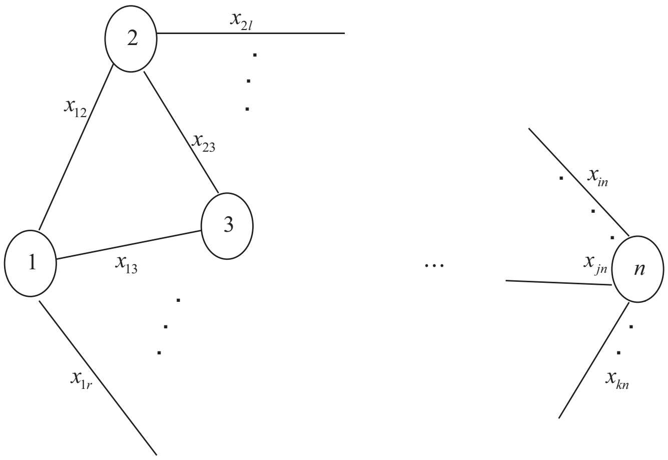


Figure 8.1: The TSP model, where nodes *i,j,k,l* and *r* are some of the nodes in the *TSP* network diagram.

From the nature of the TSP problem to every node exactly two arcs must be basic, one to come-in and the other to go-out. Suppose there are ω arcs emanating from some node *i* as shown in Figure 8.2.

It is simple that the in - out rule for the TSP will be shown by (8.1):

$$x_{i1} + x_{i2} + \dots + x_{i\omega} = 2 \tag{8.1}$$

It is also clear that if we assume that at node *1*, *r* number of links are emanating, at node 2, *number* of links are emanating and let from the node *k, the* number of links

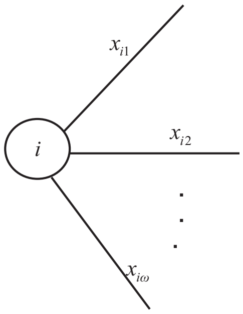


Figure 8.2: A typical sub-section of the TSP network.

emanating from node i be denoted by n. Nodes 1, 2, . . ., k will give rise to constraints as given in (8.2)

$$\begin{aligned} &x_{12} + x_{13} + \ldots + x_{1r} = 2; \\ &x_{21} + x_{23} + \ldots + x_{2i} = 2; \\ &\qquad\ldots \\ &x_{k1} + x_{kp} + x_{kn} = 2 \end{aligned} \tag{8.2}$$

This well-known formulation on its own can have feasible basic solutions corresponding to a collection of sub-tours. It can produce a collection of two or more node disjoint simple cycle covering the set of n nodes rather than a Hamiltonian cycle as required by the *TSP* problem. Detailed information about *TSP* sub-tours can be found in introductory books such as Papadimitriou and Steiglitz (1998).

Given the nature of the TSP, it may be noted that for a n node network, where $n > 2$, a minimum of n arcs are required to connect all the nodes and forming a tour. In other words, the number of basic variables in the optimal solution is n, i.e., the sum of selected arcs should be equal to n.

8.2.1 Sub-tour elimination constraints

Any two sub-tours (ST_1 and ST_2) can be connected by addition of the sub tour elimination inequality, $x_{is} + x_{jt} + \ldots + x_{ku} \geq 2$, as shown in Figure 8.3.

Here, note that:
- $x_{is}, x_{jt}, \ldots, x_{ku}$ are the non-basic variables connecting nodes in the two sub-tours,
- ℓ is a sub tour elimination line,
- $\lozenge$ represents a node in the sub-tour.

The inequality, $x_{is} + x_{jt} + \ldots + x_{ku} \geq 2$, is valid because a minimum of two arcs are required to connect any two sub-tours. There can be two or more sub-tours in a *TSP*.

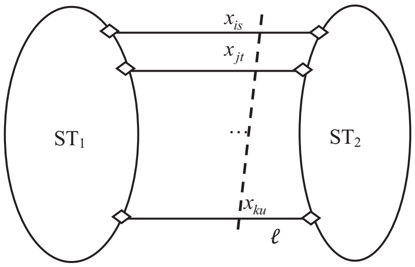


Figure 8.3: Sub-tour elimination constraint.

A sub-tour can form if the number of nodes (η) is at least three and if the number of arcs (α) connecting the identified nodes is also at least three, i.e., $\eta, \alpha \geq 3$.

Justification for the above statement is simple. The smallest sub-tour with respect to the formulation presented above can only be formed by at least three nodes and this is possible when there are three or more arcs to connect the identified nodes.

8.2.2 Some conceptual ideas and typical structure of the TSP model

Some terms are necessary for developing the further discussion.

(1) Boundary arcs

In a *TSP* network diagram, boundary arcs are defined as the outer most arcs of the network diagram. In Figure 8.1 for example, arcs (1-2), (2-*l*) or (*k*-*n*) are examples of boundary arcs.

(2) Construction of sub-tour elimination lines

Given a *TSP* diagram, all the sub-tour elimination lines can easily be identified, and the necessary sub-tour elimination constraints generated. A sub-tour elimination line in this case can be defined as a line moving from one boundary arc to another and passing through arcs only in between. All the arcs that are crossed (including the two boundary arcs) are then used to generate the sub-tour elimination constraint.

(3) The objective function of a TSP

Since the objective is to minimize the distance travelled then the objective function will be:

$$\text{Minimize } Z = c_{12}x_{12} + c_{13}x_{13} + \ldots + c_{1r}x_{1r} + \ldots + c_{kn}x_{kn}. \tag{8.3}$$

Where c_{ij} is the cost associated with moving from node the *i* to the node *j*.

(4) Non-negativity conditions on the variables

The basic nature of the variables in a TSP is that they are binary. Thus, the variables $x_{12}, x_{13}, \ldots, x_{1r}, \ldots, x_{kn}$ are binary. This integral restriction can be relaxed to develop a LP model that is easier to solve. Thus, we change the binary to continuous variables as given by (8.4)

$$\begin{gathered} 0 \le x_{12} \le 1; \\ 0 \le x_{13} \le 1; \\ \ldots \\ 0 \le x_{1r} \le 1; \\ \ldots \\ 0 \le x_{kn} \le 1. \end{gathered} \tag{8.4}$$

In other words, these variables can also be considered as upper-bounded variables.

(5) The relaxed *LP* is presented by (8.5) below

Minimize

$$\begin{gathered} Z = c_{12}x_{12} + c_{13}x_{13} + \ldots + c_{1r}x_{1r} + \ldots + c_{kn}x_{kn}; \\ x_{12} + x_{13} + \ldots + x_{1r} = 2; \\ x_{12} + x_{23} + \ldots + x_{2l} = 2; \\ \ldots \\ x_{in} + \ldots + x_{jn} + \ldots + x_{kn} = 2; \\ x_{12} + x_{13} + \ldots + x_{1r} + \ldots + x_{kn} = n; \\ x_{is} + x_{jt} + \ldots + x_{ku} \ge 2; \\ \ldots \\ x_{fo} + x_{gp} + \ldots + x_{hq} \ge 2; \\ 0 \le x_{ij} \le 1 \ \forall ij. \end{gathered} \tag{8.5}$$

The model (8.5) has five structured components, which are (1) the objective function, (2) the constraints from each node, (3) the total number of basic variables in the TSP tour, (4) sub-tour elimination constraints, and (5) the non-negative upper bound conditions. The model (8.5) is not unimodular and there is a need to convert it into a convex quadratic problem.

In the linear integer form, there is no algorithm that can solve the model (8.5) directly in polynomial time. A way out was discussed by Munapo (2016), where he transformed the linear integer model into a convex quadratic form and then applied interior point algorithms, see Gondizio (2012) to obtain an optimal integer solution.

8.2.3 Changing model (8.5) from linear integer to quadratic convex program (QP)

Let

$$S = (\, s_1 \quad s_2 \quad \dots \quad s_n \,).$$

Such that $X^T + S^T = I^T$ and $S^T \geq 0$.

Then the convex quadratic objective function becomes

$$f(\bar{X}) = \ell_1(c_1x_1^2 + c_2x_2^2 + \dots + c_nx_n^2) + (s_1^2 + s_2^2 + \dots + s_n^2) + \ell_2(x_1s_1 + x_2s_2 + \dots + x_ns_n). \tag{8.6}$$

In matrix form it simplifies to

$$f(\bar{X}) = \ell_1 CXX^T + SS^T + \ell_2 XS^T. \tag{8.7}$$

The LP model (8.5) becomes a convex quadratic problem given by (8.8).

Minimize

$$\begin{gathered} f(\bar{X}) = \ell_1 CXX^T + SS^T + \ell_2 XS^T, \\ AX^T \geq B^T, \\ X^T + S^T = I^T. \end{gathered} \tag{8.8}$$

Where ℓ_1 and ℓ_2 are very large numbers in terms of their sizes compared to any of the coefficients in the objective function. For this to work

$$\begin{aligned} &\ell_1 < < \ell_2 \\ &\ell_1 = 1000(c_1 + c_2 + \dots + c_n) \\ &\ell_2 = 1000000(c_1 + c_2 + \dots + c_n) \end{aligned} \tag{8.9}$$

Note that the weights of 1 000 and 1 000 000 depend on the sizes of the coefficients of the problem. For some small binary linear problems with small coefficients, weights of 10 and 1000 can be used effectively.

Significance of the term $\ell_2(x_1s_1 + x_2s_2 + \dots + x_ns_n)$. in the objective function in the model (8.8) acts as an 'Enforcer' towards the integer solution. Note that (8.8) is a minimization quadratic objective function, the objective function will be minimal when:

$$\ell_2(x_1s_1 + x_2s_2 + \dots + x_ns_n) = 0 \tag{8.10}$$

i.e. $\quad x_1s_1 + x_2s_2 + \dots + x_ns_n = 0$

i.e. $\quad x_1s_1 = x_2s_2 = \dots = x_ns_n = 0$

This is possible when either $x_j = 0$ or $s_j = 0$. The expression in (8.10) is called an enforcer since it forces the variables to assume only binary values. It may further be noted that from (8.5) the linear term and (8.6), the quadratic terms are equivalent.

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n = c_1x_1^2 + c_2x_2^2 + \ldots + c_nx_n^2 \tag{8.11}$$

The two quantities are equal if $x_j = 0$ or $x_j = 1$. Similarly,

$$s_1 + s_2 + \ldots + s_n = s_1^2 + s_2^2 + \ldots + s_n^2 \tag{8.12}$$

if $s_j = 0$ or $s_j = 1$.

8.2.4 Convexity of $f(\bar{X})$

Since $f(\bar{X}) = \ell_1(c_1x_1^2 + c_2x_2^2 + \ldots + c_nx_n^2) + (s_1^2 + s_2^2 + \ldots + s_n^2) + \ell_2(x_1s_1 + x_2s_2 + \ldots + x_ns_n)$, then this function $f(\bar{X}) = f(x_1, x_2, \ldots, x_n, s_1, s_2, \ldots, s_n)$ is convex if and only if it has second-order partial derivatives for each point $\bar{X} = (x_1, x_2, \ldots, x_n, s_1, s_2, \ldots, s_n) \in S$ and for each $\bar{X}' \in S$, all principal minors of the Hessian matrix are non-negative.

Proof
In this case

$$f(\bar{X}) = \ell_1(c_1x_1^2 + c_2x_2^2 + \ldots + c_nx_n^2) + (s_1^2 + s_2^2 + \ldots + s_n^2) + \ell_2(x_1s_1 + x_2s_2 + \ldots + x_ns_n)$$

This has continuous second order partial derivatives and the $(2n$ X $2n)$ Hessian matrix is given by

$$H(x_1, x_2, \ldots, x_n, s_1, s_2, \ldots, s_n) = \begin{bmatrix} 2\ell_1c_1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ 0 & 2\ell_1c_2 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ & & \ldots & & & & & \\ 0 & 0 & \ldots & 2\ell_1c_n & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 2 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & 0 & 2 & \ldots & 0 \\ & & \ldots & & & & & \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 2 \end{bmatrix} \tag{8.13}$$

Since all principal minors of $H(x_1, x_2, \ldots, x_n, s_1, s_2, \ldots, s_n)$ are nonnegative then $f(x_1, x_2, \ldots, x_n, s_1, s_2, \ldots, s_n)$ is convex. See Winston (2004) [14] for more on convex functions.

Note that $\bar{X}H\bar{X}^T \geq 0, \forall \bar{X}^T \geq 0$. Thus the matrix H is symmetric and positive definite.

Theorem 8.1: A binary solution that minimizes $f(\bar{X})$ also minimizes

$$c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

Proof

From $(\bar{X}) = \ell_1(c_1x_1^2 + c_2x_2^2 + \ldots + c_nx_n^2) + (s_1^2 + s_2^2 + \ldots + s_n^2) + \ell_2(x_1s_1 + x_2s_2 + \ldots + x_ns_n)$, ℓ_2 is very large and $\ell_1 < < \ell_2$ then $\ell_2(x_1s_1 + x_2s_2 + \ldots + x_ns_n) = 0$. Similarly ℓ_1 is very large and $\ell_1(c_1x_1^2 + c_2x_2^2 + \ldots + c_nx_n^2) > > (s_1^2 + s_2^2 + \ldots + s_n^2)$.

Therefore, minimizing $\{c_1x_1^2 + c_2x_2^2 + \ldots + c_nx_n^2.\}$ is the same as minimizing $\{ \mathrm{c_1x_1 + c_2x_2 + \ldots + c_nx_n}.\}$ because the variables in this case assume only binary variables.

8.2.5 Complexity of convex quadratic programming

Reason for converting a binary linear problem into a convex quadratic programming model is to use the available interior point algorithms which have polynomial complexity for convex QPs. If the binary LPs can be converted into a convex quadratic problem, then P=NP, see Fortnow (2009, 2013) and Shapiro (1979) for more on complexity and for more on convex quadratic form see Freund (2002), Jenson and Bard (2012), and Munapo and Kumar (2015).

8.2.6 Other considerations

1. Consider that the feasible solution does not exist

In the case when the given problem has no feasible solution, the convex quadratic program will have a solution, but it will not be an integer solution.

2. Consider the given problem is a mixed binary linear integer model

For such models, the enforcer $\ell_2(x_1s_1 + x_2s_2 + \ldots + x_ns_n)$ is made up of only those variables that are restricted to binary values.

Munapo (2016) presented a very interesting and important property of binary variables. The special property is given in (8.14).

$$x_j^2 + s_j^2 = 1. \tag{8.14}$$

Where s_j is a slack-variable. When $x_j = 1$ then $s_j = 0$ and vice versa. The special feature given in (8.14) will force variables to assume binary values and is incorporated into a quadratic function.

8.3 Construction of sub-tour elimination cuts

There are so many ways of generating sub-tour elimination cuts. In this chapter we present the *x*-*y* movement. In this approach we move in the *x*-direction drawing lines that separates groups of three nodes and then repeating the same technique in the *y*-direction. Suppose the TSP network diagram is given in Figure 8.4. Three nodes are used because we need a maximum of three nodes to form a sub-tour shown in Figure 8.5 and Figure 8.6.

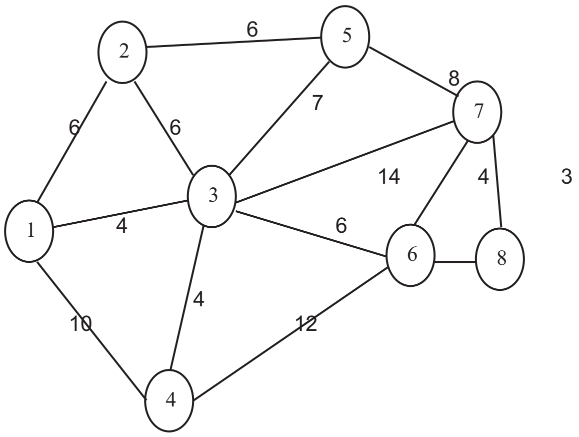


Figure 8.4: Given TSP network diagram.

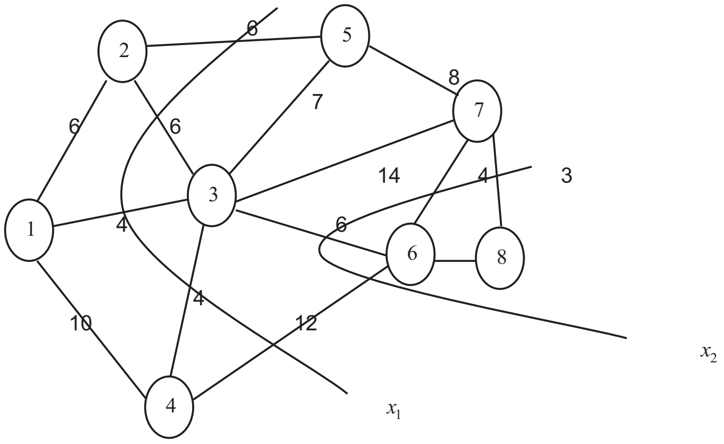


Figure 8.5: $\vec{x}$- movement.

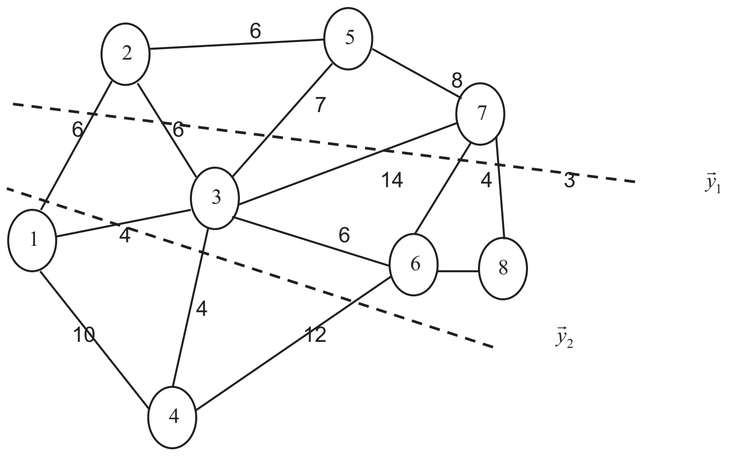


Figure 8.6: $\vec{y}$- movement.

8.4 Proposed algorithm

The algorithm will be comprised of the following steps.

Step 1: From the TSP network diagram formulate the binary LP of the TSP.

Step 2: From the TSP network diagram generate the sub-tour elimination constraints and add these to the formulated binary LP of the TSP to come up with the combined binary LP.

Step 3: Convert combined binary LP into convex quadratic problem.

Step 4: Solve the convex quadratic problem using interior point algorithm.

8.4.1 Numerical illustration

Use the proposed algorithm to solve the TSP network diagram given in Figure 8.4. The combined binary LP is formulated as (8.15).

Minimize

$$z = 6x_{12} + 4x_{13} + 10x_{14} + \ldots + 4x_{78};$$

Subject to

$$\left.\begin{aligned} x_{12} + x_{13} + x_{14} &= 2; \\ x_{12} + x_{23} + x_{25} &= 2; \\ &\ldots \\ x_{68} + x_{78} &= 2; \\ x_{12} + x_{13} + x_{14} + \ldots + x_{78} &= 8; \end{aligned}\right\} \text{Standard constraints} \qquad (8.15a)$$

$$\left.\begin{array}{r} x_{13}+x_{23}+x_{25}+x_{34}+x_{46}\geq 2;\\ x_{36}+x_{46}+x_{67}+x_{78}\geq 2;\end{array}\right\}\ \vec{x}\text{ direction} \tag{8.15b}$$

$$\left.\begin{array}{r} x_{12}+x_{23}+x_{35}+x_{37}+x_{67}+x_{78}\geq 2;\\ x_{12}+x_{13}+x_{34}+x_{45}\geq 2;\end{array}\right\}\ \vec{y}\text{ direction} \tag{8.15c}$$

Where $x_{12}, x_{13}, . . ., x_{78}$ are binary.

In the convex quadratic form the BLP will becomes (8.16).

Minimize

$$Z=\ell_1\{6x_{12}+4x_{13}+10x_{14}+\ .\ .\ .+4x_{78}\}+\{s_{12}+s_{13}+s_{14}+\ .\ .\ .+s_{78}\}+\\ \ell_2\{x_{12}s_{12}+x_{13}s_{13}+x_{14}s_{14}+\ .\ .\ .+x_{78}s_{78}\};$$

Subject to

$$\left.\begin{array}{r} x_{12}+x_{13}+x_{14}=2;\\ x_{12}+x_{23}+x_{25}=2;\\ \ldots\\ x_{68}+x_{78}=2;\\ x_{12}+x_{13}+x_{14}+\ .\ .\ .+x_{78}=8;\end{array}\right\}\ \text{Standard constraints} \tag{8.16a}$$

$$\left.\begin{array}{r} x_{13}+x_{23}+x_{25}+x_{34}+x_{46}\geq 2;\\ x_{36}+x_{46}+x_{67}+x_{78}\geq 2;\end{array}\right\}\ \vec{x}\text{ direction} \tag{8.16b}$$

$$\left.\begin{array}{r} x_{12}+x_{23}+x_{35}+x_{37}+x_{67}+x_{78}\geq 2;\\ x_{12}+x_{13}+x_{34}+x_{45}\geq 2;\end{array}\right\}\ \vec{y}\text{ direction} \tag{8.16c}$$

$$\begin{array}{c} x_{12}+s_{12}=1;\\ x_{13}+s_{13}=1;\\ x_{14}+s_{14}=1;\\ \ldots\\ x_{78}+s_{78}=1.\end{array}$$

Where $s_{12}, s_{13}, . . ., s_{78}$. are binary $\ell_1=(92)(1000)$ and $\ell_2=(92)(1000000)$.

Note that a large value of $\ell_2=(92)(1000000)$ will force $\{x_{12}s_{12}+x_{13}s_{13}+x_{14}s_{14}+\ .\ .\ .+x_{78}s_{78}\}=0$. This is only possible if variables are binary which is what we want. Solving (8.16) using interior point algorithms we have Figure 8.7a and b.

Optimal solution: $Z=45, x_{12}=x_{13}=x_{34}=x_{25}=x_{46}=x_{57}=x_{68}=x_{78}=1$ (8.7a)

Alternate solution: $Z=45, x_{12}=x_{14}=x_{25}=x_{43}=x_{36}=x_{57}=x_{68}=x_{78}=1$ (8.7b)

Where $x_{ij}=0$, $\forall$ other variables ***ij***.

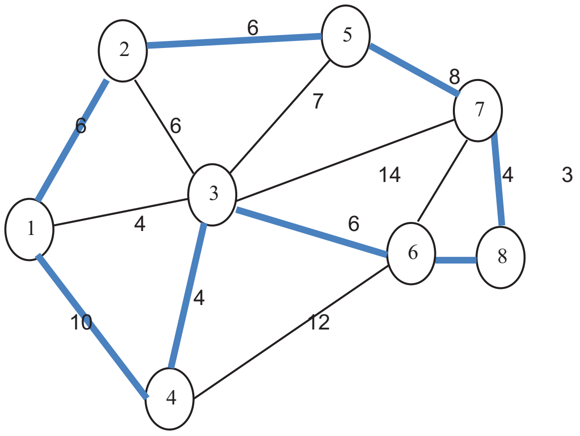


Figure 8.7a: Optimal tour in color.

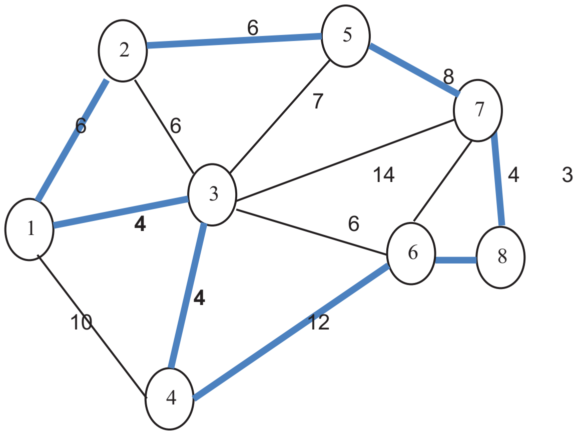


Figure 8.7b: Optimal tour-alternate solution.

8.4.2 Conclusion

The algorithm proposed in this section shows that the *TSP* can be solved efficiently in polynomial time. The main challenge of the proposed approach is the number of sub-tour elimination cuts that increase with an increase in the number variables. However, the interior point algorithms have the strength of being able to handle large number of variables. As an area of further research, there is a need to minimize the number sub-tour elimination constraints.

8.5 The transshipment approach to the travelling salesman problem

First let us reconsider a conventional formulation of a TSP.

8.5.1 Conventional formulation

In this section, we first present only the conventional TSP formulation. Readers are referred to Orman and Williams (2006) for detailed information on the other classical formulations of TSP. The challenging task of the conventional TSP formulation is the exponential increase in number of constraints as the number of centers (n) increases. The conventional formulation is presented in (8.17) and was proposed by Dantzig et al. (1957). For the network diagram, see Figure 8.1

Minimize

$$c_{12}x_{12} + c_{13}x_{13} + \ldots + c_{kn}x_{kn}$$

$$\left.\begin{aligned} \sum_j x_{ij} = 1 \\ \sum_i x_{ij} = 1 \end{aligned}\right\} \tag{8.17.1}$$

$$\sum_{i,j \in S} x_{ij} \le |S| - 1,\ |S| \ge 2, S \subseteq N\backslash\{1\}, S \ne \{1\} \tag{8.17.2}$$

$$i \ne j.$$

Note that (8.17.1) are called assignment constraints and the (8.17.2) are called subtour elimination constraints.

8.5.2 Some important properties of a totally unimodular matrix

A Matrix A is totally unimodular (TU) if the determinant of every square sub-matrix of A (also called minor of A) has value -1, 0 or 1, for more details see De Werra (1981). The properties of totally unimodular matrices, which are relevant for this section are:

Property 8.1: If (0,1,-1) matrix A has no more than two nonzero entries in each column and if $\sum a_{ij} = 0$ for column j that contains two nonzero elements then A is totally unimodular (TU).

Property 8.2: A matrix obtained by multiplying a row/column of A by -1 is TU.

The proofs for these two important properties are given in Nemhauser and Wolsey (1999).

8.5.3 Breaking a TSP into transshipment sub-problems

Given any two adjacent nodes on the boundary of a TSP model, all the possible ways of connecting the two nodes directly or via other nodes can be viewed as a transshipment problem.

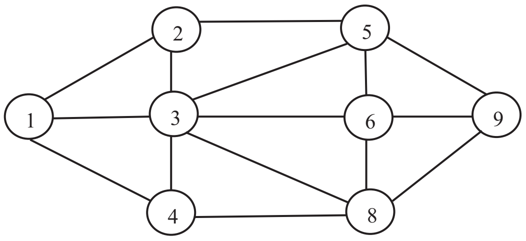


Figure 8.8: Breaking a TSP into transshipment.

Suppose the two adjacent boundary centers are 2 and 5 as shown in Figure 8.8 and that we are starting from base 2. There are only three feasible ways of moving from center 2 to center 5. These ways are:

2-5	Direct movement (way 1)
2-3-5	Indirect movement (way 2)
2-3-6-5	Indirect movement (way 3)

The other possible but not feasible ways are:
2-3-8-6-5
2-3-4-8-6-5

These two ways are not feasible in the sense that, a center can only be visited once. We cannot include center 8 in the movement from 2 as it will be used on the way back to base. This is called *back space* in this chapter. The possible movements from 2 to 5 can be described as a transshipment problem where center 2 is the supply point, center 5 is the demand point and 3 and 6 being the transshipment centers. The total supply is 1 and total demand is 1. This transshipment sub-problem is shown in Figure 8.9.

This give rise to the transshipment constraints for this sub-problem are:

$$\begin{aligned} x_{23} + x_{25} &= 1 \\ x_{25} + x_{35} + x_{56} &= 1 \\ x_{23} &= x_{35} + x_{36} \\ x_{36} &= x_{56} \end{aligned} \tag{8.18}$$

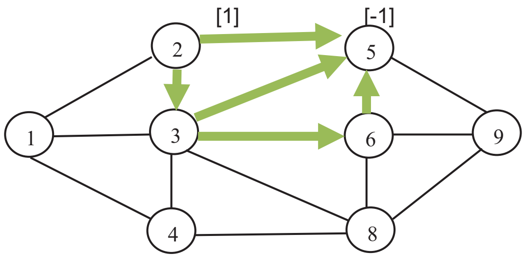


Figure 8.9: Transshipment sub-problem.

The coefficient matrix for this transshipment sub-problem can be shown to be totally unimodular (TU).

$$\begin{array}{ccccc} x_{23} & x_{25} & x_{35} & x_{36} & x_{56} \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{array}$$

Multiply row one by −1

$$\begin{array}{ccccc} -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{array}$$

The coefficient matrix is unimodular because:
- Every column has a maximum of two nonzero elements.
- Sum of every column is 0.

For more on transshipment models, see Balakrishman (1995).

8.5.4 General case – transshipment sub-problem

The transshipment constraints for this general case sub-problem are:

$$x_{ij} + x_{il} + \ldots + x_{ik} = 1$$

$$x_{ij} + x_{jt} + \ldots + x_{js} = 1$$

$$\ldots$$

$$x_{il} = x_{lp} + \ldots + x_{lq}$$

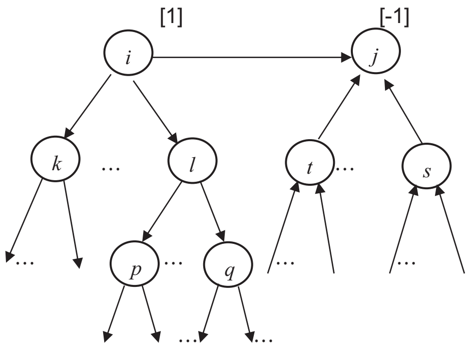


Figure 8.10: General case – Transshipment sub-problem.

8.5.5 Standard constraints

According to the definition of a TSP, a center must be visited once. This implies that one must enter a center through one arc and leave through a different arc. This is presented in Figure 8.11 and gives a set of constraints. This class of constraints is not necessarily new as they have been used before without success. On their own do not give a feasible optimal solution. Some of the optimal solutions may have sub-tours.

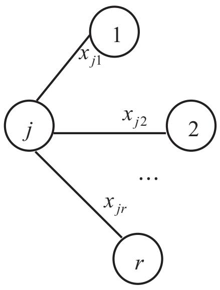


Figure 8.11: Standard constraint.

Suppose *r* is the number of arcs originating from center *j*.

$$x_{j1} + x_{j2} + \ldots + x_{jr} = 2 \tag{8.19}$$

Where $r \in N$. Standard constraints on their own give optimal integer solutions. The only problem is that the optimal integer solution is sometimes not feasible for the TSP, as it may contain sub-tours.

Proof of integrality

Given a TSP network of the form shown in Figure 8.1, the number of standard constraints can be found to be *n*. If *n* standard constraints are used on their own, then there are *n* basic variables at optimality. This implies that one of the optimal solutions is integer.

8.5.6 Infeasibility

For the formulated model to be feasible the transshipment constraints for an arc must not have the same variables as those that are contained in transshipment constraints of other arcs. In other words, the transshipment sub-problems cannot share variables. This challenge is alleviated by introducing extra variables. Suppose variable x_i is common in sets of constraints from *j* arcs. For the model to be feasible this variable is represented as *j* different variables i.e.

$$x_i^1, x_i^2, \ldots, x_i^j \tag{8.20}$$

8.6 The transshipment TSP linear integer model

For any TSP network model of the form given in Figure 8.1, a transshipment TSP linear integer model can be formulated as:
Minimize

$$c_{12}x_{12} + c_{13}x_{13} + \ldots + c_{kn}x_{kn}$$

Such that

$$\left.\begin{array}{l} \left.\begin{array}{r} \ldots \\ x_{j1} + x_{j2} + \ldots + x_{jr} = 2 \\ \ldots \end{array}\right\} \text{Standard constraints} \\ x_{ij} + x_{il} + \ldots + x_{ik} = 1 \\ \left.\begin{array}{r} x_{ij} + x_{jt} + \ldots + x_{js} = 1 \\ \ldots \end{array}\right\} \text{Transshipment constraints for arc (i, j)} \end{array}\right\} \tag{8.21}$$

$$x_{il} = x_{lp} + \ldots + x_{lq}$$

$$\ldots$$

Where x_{ij} are binary variables $\forall ij$ and $i \neq j$.

The formulated model can be shown to be infeasible if the transshipment sub-problems contain common variables.

8.6.1 Numerical illustration

The transshipment linear integer model for Figure 8.12 is presented in model (8.22).

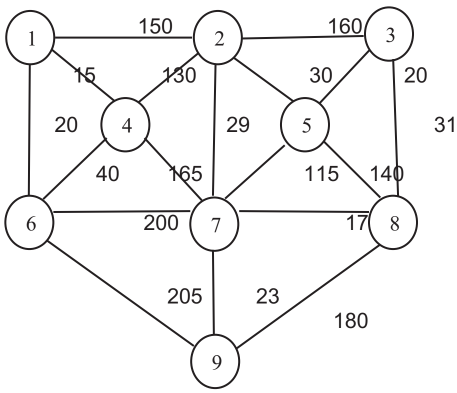


Figure 8.12: Numerical Illustration.

8.6.2 The formulated transshipment TSP linear integer model

$$\text{Objective function: Minimize} \left. \begin{array}{l} 150x_{12}+15x_{14}+20x_{16}+160x_{23}+130x_{24}+30x_{25}+ \\ 29x_{27}+20x_{35}+31x_{38}+40x_{46}+165x_{47}+115x_{57}+ \\ 140x_{58}+200x_{67}+205x_{69}+170x_{78}+23x_{79}+180x_{89} \end{array} \right\} \tag{8.22.1}$$

Standard constraints:

$$\begin{array}{l} x_{12}+x_{14}+x_{16}=2 \quad (\text{Centre } 1) \\ x_{12}+x_{23}+x_{24}+x_{25}+x_{27}=2 \quad (\text{Centre } 2) \\ x_{23}+x_{35}+x_{38}=2 \quad (\text{Centre } 3) \\ x_{14}+x_{24}+x_{46}+x_{47}=2 \quad (\text{Centre } 4) \\ x_{25}+x_{35}+x_{57}+x_{58}=2 \quad (\text{Centre } 5) \\ x_{16}+x_{46}+x_{67}+x_{69}=2 \quad (\text{Centre } 6) \\ x_{27}+x_{47}+x_{57}+x_{67}+x_{78}+x_{79}=2 \quad (\text{Centre } 7) \\ x_{38}+x_{58}+x_{78}+x_{89}=2 \quad (\text{Centre } 8) \\ x_{69}+x_{79}+x_{89}=2 \quad (\text{Centre } 9) \end{array} \tag{8.22.2}$$

Transshipment sub-problem constraints are shown in equation (8.22.3) and shown in Figure 8.13. Similarly other transshipment constraints are shown in Figures 8.14 to Figure 8.18:

$$\begin{aligned}
&x_{12}+x_{14}=1 \quad (\text{Centre } 1)\\
&x_{12}+x_{24}+x_{25}+x_{27}=1 \ (\text{Centre } 2)\\
&x_{14}=x_{24}+x_{47} \ \text{Transshipment 1 from arc } (1,2)\\
&x_{47}=x_{27}+x_{57}\\
&x_{57}=x_{25}
\end{aligned} \tag{8.22.3}$$

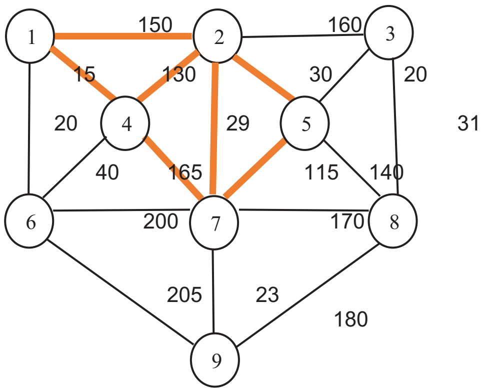


Figure 8.13: Illustration.

$$\begin{aligned}
&x_{23}+x_{24}+x_{25}+x_{27}=1 \ (\text{Centre } 2)\\
&x_{23}+x_{35}=1 \ (\text{Centre } 3)\\
&x_{24}=x_{47} \ \text{Transshipment 2 from arc } (2,3)\\
&x_{27}+x_{47}=x_{57}\\
&x_{25}+x_{57}=x_{35}
\end{aligned} \tag{8.22.4}$$

$$\begin{aligned}
&x_{35}+x_{38}=1 \ (\text{Centre } 3)\\
&x_{38}+x_{58}+x_{78}=1 \ (\text{Centre } 8)\\
&x_{35}=x_{57}+x_{58} \ \text{Transshipment 3 from arc } (3,8)\\
&x_{57}=x_{78}
\end{aligned} \tag{8.22.5}$$

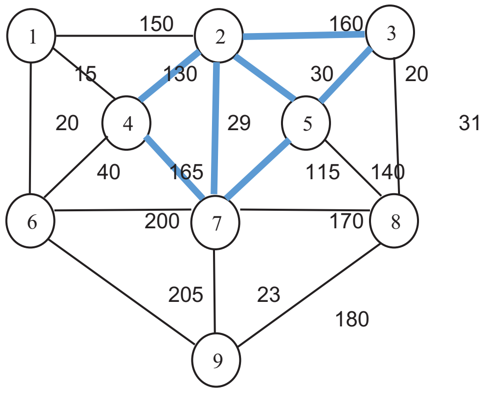


Figure 8.14: Illustration for the link (2-3).

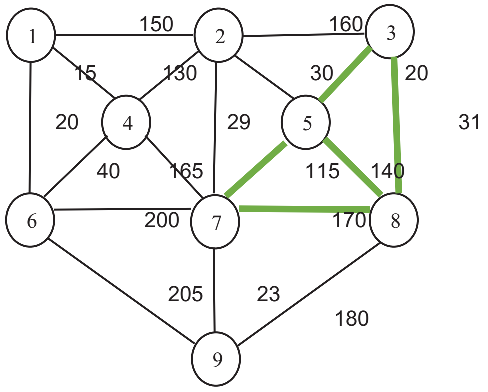


Figure 8.15: Illustration for the link (3-8).

$$\begin{aligned}
&x_{58}+x_{78}+x_{89}=1\\
&x_{79}+x_{89}=1\\
&x_{58}=x_{57}\ \text{Transshipment 4 from arc } (8,9)\\
&x_{57}+x_{78}=x_{79}
\end{aligned}\tag{8.22.6}$$

$$\begin{aligned}
&x_{69}+x_{79}=1\ (\text{Centre } 9)\\
&x_{46}+x_{67}+x_{69}=1\ (\text{Centre } 6)\\
&x_{79}=x_{47}+x_{67}\ \text{Transshipment 5 from arc } (9,6)\\
&x_{47}=x_{46}
\end{aligned}\tag{8.22.7}$$

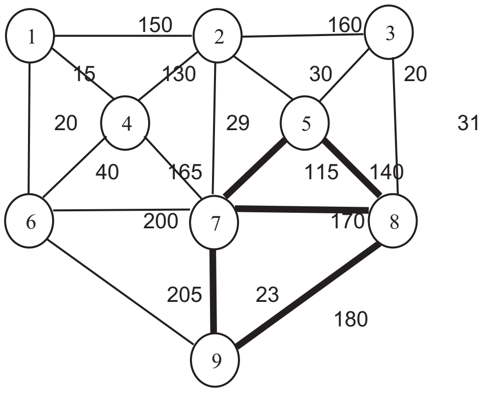


Figure 8.16: Illustration link (8-9).

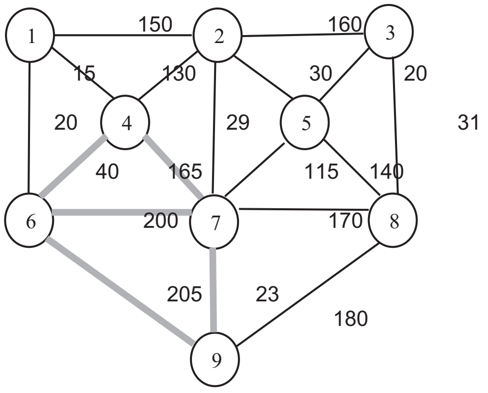


Figure 8.17: Illustration link (9-6).

$$\begin{aligned}
&x_{16} + x_{46} + x_{67} = 1 \ (\text{Centre } 6)\\
&x_{14} + x_{16} = 1 \ (\text{Centre } 1)\\
&x_{47} = x_{67} \text{ Transshipment 6 from arc } (6,1)\\
&x_{46} + x_{47} = x_{14}
\end{aligned} \tag{8.22.8}$$

Binary variables

x_{ij} is a binary variable $\forall ij$.

This integer model is infeasible. The infeasibility is alleviated by considering variables that are common to two or more transshipment sub-problems and replace them by new variables.

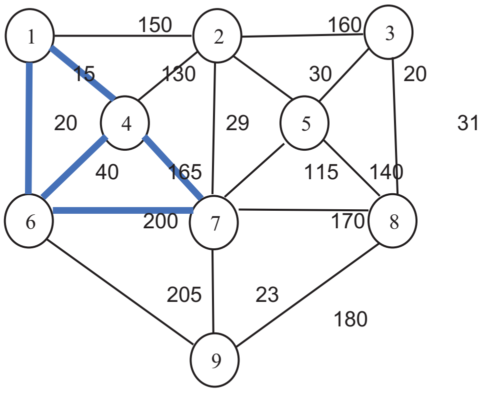


Figure 8.18: Illustration link (6-1).

New variables

- x_{14} is common to transshipments 1 and 6, thus introduce x^1_{14} and x^6_{14}.
- x_{24} is common to transshipments 1 and 2, thus introduce x^1_{24} and x^2_{24}.
- x_{25} is common to transshipments 1 and 2, thus introduce x^1_{25} and x^2_{25}.
- x_{27} is common to transshipments 1 and 2, thus introduce x^1_{27} and x^2_{27}.
- x_{35} is common to transshipments 2 and 3, thus introduce x^2_{35} and x^3_{35}.
- x_{46} is common to transshipments 5 and 6, thus introduce x^5_{46} and x^6_{46}.
- x_{47} is common to transshipments 1,2,5 and 6, thus introduce $x^1_{47}, x^2_{47}, x^5_{47}$ and x^6_{47}.
- x_{57} is common to transshipments 1,2,3 and 4, thus introduce $x^1_{57}, x^2_{57}, x^3_{57}$ and x^4_{57}.
- x_{58} is common to transshipments 3 and 4, thus introduce x^3_{58} and x^4_{58}.
- x_{67} is common to transshipments 5 and 6, thus introduce x^5_{67} and x^6_{67}.
- x_{78} is common to transshipments 3 and 4, thus introduce x^3_{78} and x^4_{78}.
- x_{79} is common to transshipments 4 and 5, thus introduce x^4_{79} and x^5_{79}.

The feasible linear integer model becomes:

$$\text{Minimize}\quad \left.\begin{aligned}&150x_{12}+15x^1_{14}+15x^6_{14}+20x_{16}+160x_{23}+130x^1_{24}+130x^2_{24}+30x^1_{25}\\&+30x^2_{25}+29x^1_{27}+29x^2_{27}+20x^2_{35}+20x^3_{35}+31x_{38}+40x^5_{46}\\&+40x^6_{46}+165x^1_{47}+165x^2_{47}+165x^5_{47}+165x^6_{47}+115x^1_{57}+115x^2_{57}\\&+115x^3_{57}+115x^4_{57}+140x^3_{58}+140x^4_{58}+200x^5_{67}+200x^6_{67}\\&+205x_{69}+200x^3_{78}+200x^4_{78}+30x^4_{79}+23x^5_{79}+180x_{89}\end{aligned}\right\}\quad (8.23.1)$$

Standard constraints

$$\begin{aligned}
&x_{12}+x_{14}^1+x_{14}^6+x_{16}=2 \quad (\text{Centre }1)\\
&x_{12}+x_{23}+x_{24}^1+x_{24}^2+x_{25}^1+x_{25}^2+x_{27}^1+x_{27}^2=2 \quad (\text{Centre }2)\\
&x_{23}+x_{35}^2+x_{35}^3+x_{38}=2 \quad (\text{Centre }3)\\
&x_{14}^1+x_{14}^6+x_{24}^1+x_{24}^2+x_{46}^5+x_{46}^6+x_{47}^1+x_{47}^2+x_{47}^5+x_{47}^6=2 \quad (\text{Centre }4)\\
&x_{25}^1+x_{25}^2+x_{35}^2+x_{35}^3+x_{57}^1+x_{57}^2+x_{57}^3+x_{57}^4+x_{58}^3+x_{58}^4=2 \quad (\text{Centre }5)\\
&x_{16}+x_{46}^5+x_{46}^6+x_{67}^5+x_{67}^6+x_{69}=2 \quad (\text{Centre }6)\\
&x_{27}^1+x_{27}^2+x_{47}^1+x_{47}^2+x_{47}^5+x_{47}^6+x_{57}^1+x_{57}^2+\\
&x_{57}^3+x_{57}^4+x_{67}^5+x_{67}^6+x_{78}^3+x_{78}^4+x_{79}^4+x_{79}^5=2 \quad (\text{Centre }7)\\
&x_{38}+x_{58}^3+x_{58}^4+x_{78}^3+x_{78}^4+x_{89}=2 \quad (\text{Centre }8)\\
&x_{69}+x_{79}^4+x_{79}^5+x_{89}=2 \quad (\text{Centre }9)
\end{aligned} \tag{8.23.2}$$

Transshipment sub-problem constraints

$$\left.\begin{aligned}
x_{12}+x_{14}^1&=1\\
x_{12}+x_{24}^1+x_{25}^1+x_{27}^1&=1\\
x_{14}^1&=x_{24}^1+x_{47}^1\\
x_{47}^1&=x_{27}^1+x_{57}^1\\
x_{57}^1&=x_{25}^1
\end{aligned}\right\}\text{Transshipment 1} \tag{8.23.3}$$

$$\left.\begin{aligned}
x_{23}+x_{24}^2+x_{25}^2+x_{27}^2&=1\\
x_{23}+x_{35}^2&=1\\
x_{24}^2&=x_{47}^2\\
x_{27}^2+x_{47}^2&=x_{57}^2\\
x_{25}^2+x_{57}^2&=x_{35}^2
\end{aligned}\right\}\text{Transshipment 2} \tag{8.23.4}$$

$$\left.\begin{aligned}
x_{35}^3+x_{38}&=1\\
x_{38}+x_{58}^3+x_{78}^3&=1\\
x_{35}^3&=x_{57}^3+x_{58}^3\\
x_{57}^3&=x_{78}^3
\end{aligned}\right\}\text{Transshipment 3} \tag{8.23.5}$$

$$\left.\begin{aligned}
x_{58}^4+x_{78}^4+x_{89}&=1\\
x_{79}^4+x_{89}&=1\\
x_{58}^4&=x_{57}^4\\
x_{57}^4+x_{78}^4&=x_{79}^4
\end{aligned}\right\}\text{Transshipment 4} \tag{8.23.6}$$

$$\left.\begin{array}{c} x_{69}+x_{79}^5=1 \\ x_{46}^5+x_{67}^5+x_{69}=1 \\ x_{79}^5=x_{47}^5+x_{67}^5 \\ x_{47}^5=x_{46}^5 \end{array}\right\}\text{Transshipment 5} \tag{8.23.7}$$

$$\left.\begin{array}{c} x_{16}+x_{46}^6+x_{67}^6=1 \\ x_{14}^6+x_{16}=1 \\ x_{47}^6=x_{67}^6 \\ x_{46}^6+x_{47}^6=x_{14}^6 \end{array}\right\}\text{Transshipment 6} \tag{8.23.8}$$

Solution
The optimal solution to the relaxed model is obtained as

$$\left.\begin{array}{c} x_{14}^1=x_{16}=x_{24}^1=x_{25}^2=x_{35}^2=x_{38}=x_{69}+x_{78}^4=x_{79}^4=1 \\ x_{12}=x_{23}=x_{14}^6=x_{24}^2=x_{25}^1=x_{27}^1=x_{27}^2=x_{35}^3=x_{46}^5=0 \\ x_{46}^6=x_{47}^1=x_{47}^2=x_{47}^5=x_{47}^6=x_{57}^1=x_{57}^2=x_{57}^3=x_{57}^4=0 \\ x_{58}^3=x_{58}^4=x_{67}^5=x_{67}^6=x_{78}^3=x_{79}^3=x_{89}=0 \end{array}\right\} \tag{8.23.9}$$

Going back to the original variables this is the same as

$$\left.\begin{array}{c} x_{14}=x_{16}=x_{24}=x_{25}=x_{35}=x_{38}=x_{69}+x_{78}=x_{79}=1 \\ x_{12}=x_{23}=x_{27}=x_{46}=x_{47}=x_{57}=x_{58}+x_{67}=x_{89}=0 \end{array}\right\} \tag{8.23.10}$$

The optimal cost is = 644. The optimal solution is shown in Figure 8.19.

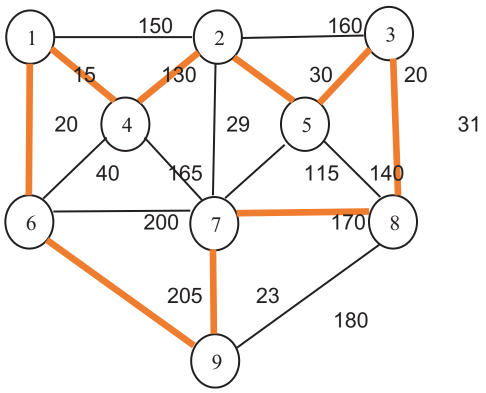


Figure 8.19: The optimal solution to the illustrative example.

8.7 Conclusions

The TSP model has so many applications in real life. So much work in terms of research has been done without a breakthrough on this model. We are not aware of any efficient and consistent approach to this problem. We hope that the approaches discussed in this chapter will open new avenues for other researchers. There is no need for those branch-and-bound related approaches where there is a fear computational explosion. Also the approaches presented here can solve the TSP in polynomial time.

References

Applegate, D.L., R.E. Bixby, V. Chvatal, W.L. Cook. 2007. The Traveling Salesman Problem: A Computational Study. Princeton University Press.

Balakrishman, VK. Network Optimization, Chapman and Hall Mathematics, London, 1995.

Bland, R.E. and Shallcross, D.E. 1989. Large traveling salesman problem arising from experiments in X-ray crystallography, a preliminary report on computation, Operations Research Letters 8(3), 125–128.

Claus, A. 1984. A new formulation for the travelling salesman problem, SIAM J. Alg. Disc. Math. 5, 21–25.

Dantzig, G.B., Fulkerson, D.R. and Johnson, S.M. 1954. Solutions of a large-scale travelling salesman Problem, Ops. Res. 2 pp 393–410.

De Werra D., 1981. On some characterizations of totally unimodular matrices, Mathematical Programming, 20 pp. 14–21.

G. Finke, Claus, A., and Gunn, E. 1983. A two-commodity network flow approach to the travelling salesman problem, Combinatorics, Graph Theory and Computing, Proc.14th South Eastern Conf., Atlantic University, Florida.

Freund, R.M. 2002. Solution Methods for Quadratic Optimization: Lecture notes, Massachusetts Institute of Technology. 2002.

Fortnow, F. 2013. The Golden Ticket: P, NP, and the Search for the Impossible. Princeton University Press. Princeton, NJ.

Fortnow, F. 2009.The status of the P versus NP problem. Communications of the ACM 52(9), 78–86.

Fox, K.R., Gavish, B., and Graves, S.C. 1980. An n-constraint formulation of the (time-dependent) travelling salesman problem, Ops. Res. 28, 1018–1021.

Gavish, B., Graves,S.C. 1978. The traveling salesman problem and related problems, Working Paper OR-078-78, Operations Research Center, MIT, Cambridge MA.

Gondzio, J. 2012. Interior Point Methods 25 Years Later. *European* Journal of Operational Research 218, 587–601.

Grötschel, M. Jünger, G. Reinelt, 1991. Optimal Control of Plotting and Drilling Machines: A Case Study, Mathematical Methods of Operations Research 35(1), 61–84.

Gutin G., A.P. Punnen (eds.). 2006. The Traveling Salesman Problem and its Variations. Springer.

Jensen, P.A., and Bard, J.F. 2012. Operations Research Models and Methods. John Wiley and Sons, Inc.

Kumar, S., Munapo, E., Lesaoana, M., and Nyamugure, P. (2016), Is the travelling salesman problem actually NP hard? Chapter 3 in Engineering and Technology: Recent Innovations and

Research, Editor A. Matani, International Research Publication House, ISBN 978-93-86138-06-4, pp 37–58.

Kumar, S., Munapo, E., Lesaoana, M., and Nyamugure, P. 2017. A minimum spanning tree-based heuristic for the travelling salesman tour, Opsearch, DOI 10.1007/s1259-017-0318-5, pp1–15.

Kumar, S., Munapo, E., Sigauke, C., and Al-Rabeeah, M. 2020. The minimum spanning tree with node index ≤2 is equivalent to the minimum travelling salesman tour, Chapter 8 in Mathematics in Engineering Sciences: Novel Theories, Technologies, and Applications Edited by Mange Ram, CRC Press, ISBN 13:978-1-138-57767-1.

Lenstra, L.K., Rinnooy Kan, A.G.H., 1975. Some simple applications of the traveling salesman Problem, Operational Research Quarterly 26, pp. 717–33.

Miller, C.E., Tucker, A.W., and Zemlin, R.A. 1960. Integer programming formulation of travelling salesman problems, J. ACM 3, pp326–329.

Munapo, E. (2016) Solving the Binary Linear Programming Model in Polynomial Time, American Journal of Operations Research, 6, 1–7. http://dx.doi.org/10.4236/ajor.2016.61001

Munapo, E. and Kumar, S. (2015) A New Heuristic for the Convex Quadratic Programming Problem. American Journal of Operations Research, 5, 373–383. http://dx.doi.org/10.4236/ajor.2015.55031

Munapo, E., Kumar, S., Lesaoana, M., and Nyamugure, P. 2016. A minimum spanning tree with node index ≤2, ASOR Bulletin, Vol 34, Issue 1, pp 1–14.

Munapo, E. 2020. Development of a dummy guided formulation and exact solution method for TSP, Eastern-European Journal of Enterprise Technologies, pp. 12–19.

Nemhauser, G, and Wolsey, L. 1999. Integer and Combinatorial Optimization, John Wiley

Orman, A.J., Williams, H.P. 2006. A survey of different integer programming formulations of the travelling salesman problem. In: Kontoghiorghes E. & Gatu C. (eds). Optimization, Econometric and Financial Analysis Advances in Computational Management Science, Springer: Berlin, Heidelberg, pp. 91–104.

Papadimitriou, C.H., K. Steiglitz. 1998. Combinatorial Optimization: Algorithms and Complexity. Dover Publications.

Plante, R.D., Lowe, T.J., and Chandrasekaran, R. 1987. The Product Matrix Traveling Salesman Problem: An Application and Solution Heuristics, Operations Research 35, 772–783.

Shapiro J.F. 1979. Mathematical Programming: Structures and Algorithms. John Wiley and Sons.

Schrijver, A. 1998. Theory of Linear and Integer Programming. Wiley, John Wiley & Sons, ISBN 978-0-471-98232–6.

Vajda, S. 1961. Mathematical Programming, Addison-Wesley, London.

Van Dal, R. 1992. Special Cases of the Traveling Salesman Problem. Wolters-Noordhoff, Groningen.

Winston, W.L. 2004. Operations Research: Applications and Algorithms. Duxbury Press 4th Edition.

Wong, R.T. 1980. Integer programming formulations of the travelling salesman problem, Proc. IEEE Conf. on Circuits and Computers, pp. 149–152.

Index

https://doi.org/10.1515/9783110703023-009