REDUNDANCY ALLOCATIONS IN ELECTRONIC RELAY CIRCUITS

BY

KRISHNA BEHARI MISRA

A THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING,
UNIVERSITY OF ROORKEE,
ROORKEE.

APRIL, 1970.

CERTIFICATE

CERTIFIED that the thesis entitled "REDUNDANCY ALLOCATIONS IN ELECTRONIC RELAY CIRCUITS" which is being submitted by Mr. Krishna Behari Misra in fulfilment of the requirements for the degree of Doctor of Philosophy (Electrical Engineering) of the University of Roorkee, is a record of the student's own work carried out by him under my supervision and guidance. The matter embodied in this dissertation has not been submitted for the award of any other degree or diploma.

This is further to certify that he has worked for a period of three years and one month from February 1967 to March 1970 for preparing this thesis for the Doctor of Philosophy Degree, at the University.

T. S. Madhav Rao
Professor & Head
Department of Elect. Engg.
University of Roorkee

Roorkee, India:
April 4, 1970

ABSTRACT

The thesis gives a detailed study of the problem of redundancy allocations in electronic circuits associated with the protective relay circuits. The approach has been kept general so that application to various fields is unrestricted. Operational reliability is the main concern of any electronic circuit associated with such protective relay circuits. Unless the electronic components are made absolutely reliable by tried and tested methods of manufacturing processes, the choice rests on duplicating the components or, in general, what is called as redundancy applications.

The thesis begins with a detailed study of the redundancy circuits and their modelling as far as the reliability evaluation is concerned.

Various types of redundant circuits are analysed to complete the study. Different approaches are devised for reliability evaluation of such networks. In general, one may come across series and/or parallel or non series-parallel networks in practice. The non series-parallel networks usually present difficultywhen the problem is to evaluate the overall reliability of such networks. Flow-graph method has been developed wherein a method of inspection makes it all the more easy to calculate reliability of the redundant networks, quickly.

If the network is large and complex, the reliability evaluation poses a problem; therefore an algorithm is presented for straight and fast computation on a digital computer for any type of the redundant network. This has been possible by correlating the properties of redundant networks with those of di-graphs.

The thesis embodies optimisation techniques for maximisation of the system reliability subject to linear or non-linear constraints. Here again, various techniques have been applied, viz. gradient method, Kuhn-Tucker conditions of optimality, Dynamic programming, Variational method, Discrete maximum principle, Integer linear programming etc.

Several new approaches and modifications of the existing methods have been proposed and they are tested on problems from various sources.

One usually faces the problem of choosing proper values of Lagrangian multipliers when solving an optimisation problem with linear constraints. Attempts have been made to make proper selection of these and to solve such problems with ease. Dynamic programming formulation in 'summation' form has been developed and was found to be more convenient than usual 'product' formulation. An algorithm based on Lagrangian multipliers and general optimal condition is proposed in case of problems with linear constraints.

A Variational method for multiple linear constraints is also developed and has been tried on several problems. Discrete maximum principle has been used for problems with linear and non-linear constraints. Discrete optimisation technique is discussed in general perspective for reliability optimisation under several constraints. In the end a comparative assessment of the methods embodied in the thesis is made to provide the merits and demerits of each so as to allow one to make his own choice of the method under limitations and advantages exposed.

In brief, a detailed mathematical analysis has been presented for the problem of reliability evaluation and optimisation of the redundant networks under conditions specified which will help to pave the way for making circuits or systems more reliable.

ACKNOWLEDGEMENTS

The author expresses his profound gratitude to Dr. T. S. Madhav Rao, Professor & Head of the Electrical Engineering Department at the University of Roorkee, for his constant inspiration and valuable guidance given by him during the course of study and investigations reported herein. Thanks are also due to him for making various facilities available for carrying out the work.

The author is grateful to the staff of the Computer Centres at the Structural Engineering Research Centre, Roorkee, and the Indian Institute of Technology, Kanpur, for the facility of using their digital computers.

CONTENTS

Page

ABSTRACT i

ACKNOWLEDGEMENTS iv

INTRODUCTION 1

CHAPTER 1 - Analysis of Redundant Networks 6

1.1 - Introduction .. 6

1.2 - Definitions .. 6

1.3. - Redundant Networks .. 8

1.3.1 - Series-Parallel Configurations .. 9

1.3.2 - Non Series-Parallel Networks .. 10

1.3.3 - Dual Networks .. 10

1.3.4 - Development of Reliability Models 11

1.4. - Analysis of Redundant Networks .. 11

1.4.1 - Basic Property of an Element .. 12

1.5 - Historical Procedures .. 13

1.6 - Some Properties of Reliability Expressions for a Redundant Network .. 14

1.7 - Flow-graph Method .. 16

1.7.1 - A Speedy Method of Inspection .. 18

1.7.2 - Non Series-parallel Redundant Networks 20

1.8 - Networks with elements that can short or open 23

1.8.1 - Paths and Cuts .. 23

1.8.2 - Open and short circuit failures .. 25

1.9 - An Algorithm for Direct Reliability Evaluation using Di-graph Matrices .. 27

1.9.1 - Modelling of Networks .. 27

Page

1.9.2 - Combinations of parallel elements 27

1.9.3 - Series-parallel Networks .. 29

1.9.4 - Non Series-parallel Networks .. 34

1.9.5 - Applications .. 38

1.10 - A method of deriving Reliability expression 38

CHAPTER 2 - Optimisation of Reliability with Linear Constraints .. 40

2.1 - Statement of the Problem .. 40

2.2 - Domination .. 41

2.3 - Approximate Solution of Redundancy Allocation Problem 41

2.3.1 - Examples 42

(i) Single Cost Factor .. 42

(ii) Multiple Cost Factors .. 43

2.3.2 - Alternative Method .. 47

(i) Discrete steps variation .. 47

(ii) Random Numbers Approach .. 49

(iii) A Graphical Method .. 51

2.4 - Kettele's Algorithm .. 54

2.4.1 - Dominating Sequence .. 54

2.4.2 - Illustrative example .. 54

2.4.3 - Multiple cost constraints .. 58

2.4.4 - Multiple cost problem .. 59

2.5 - Bellman Dynamic Programming Approach 64

2.5.1 - Dynamic Programming Formulation .. 64

2.5.2 - Optimisation with two constraints and one control variable .. 67

2.5.3 - Lagrange's Multiple Technique .. 69

Page

2.5.4 - Optimum Redundancy Allocation .. 71

2.5.5 - Example 73

CHAPTER 3 - Variational Approach .. 81

3.1 - Condition for minimum cost .. 81

3.2 - Procedure for calculating optimum allocation 84

3.3 - Allocations with given cost constraint 87

3.4 - Allocations with Multiple constraints 90

3.4.1 - Example 92

3.5 - Optimisation using Maximum Principle 94

3.6 - Discrete Maximum Principle with Linear Constraints 97

CHAPTER 4 - Optimisation of System Reliability with Non-linear Constraints .. 103

4.1 - Proschan and Bray Extension .. 103

4.2 - Discrete Maximum Principle .. 107

CHAPTER 5 - Integer Programming Formulation 109

5.1 - Introduction .. 109

5.2 - Mizukami Formulation .. 109

5.3 - Tillman Formulation .. 111

5.4 - Future Approaches .. 116

5.4.1 - Lawler's Method .. 117

5.4.2 - Branch and Bound Method .. 119

CHAPTER 6 - Conclusions .. 120

APPENDICES:

A - Theorem 126

B - Program 128

C - Program 129

Page

D - Program 131

E - Kuhn-Tucker Theorem . .. 133

F - Program 136

G - Discrete Maximum Principle .. 138

H - Program 142

H-1 - Program 145

I - Subroutine Simplex . 147

List of Publications .. 149

References 150

About the Author 153

INTRODUCTION

It was mainly during World War II and the post-War years that the need for reliable electronic devices was unquestionably felt. Early efforts in this direction were aimed, principally toward determining the causes of unreliability.

Von Neumann, Shanon and Moore [1], were perhaps the investigators whose contribution in this field, gave impetus to the development of mathematical reliability theory. With the electronic devices and systems becoming increasingly complex and thus more susceptible to failures, new techniques for their reliability analysis, had to be developed. Much of the literature available on the subject, has come out in the past few years only.

More recently power system protective schemes have also undergone a remarkable change especially with advent of solid state devices. The shift has been from conventional relays to electronic relay schemes. It is needless to stress the importance of reliability of such schemes, whose failure may cause heavy financial loss and inconvenience.

Fundamentally, every electronic relay consists of several components such as tubes, transistors, resistors, condensers etc. The reliability of each of such components contributes to the overall reliability of the relay. It is therefore in this context that the thesis presents a generalised approach to the reliability analysis of such circuits.

Basically, there are two ways of achieving higher system reliability. The first is to develop highly reliable components for use in equipments and systems. The second is to design reliable systems from less reliable parts through use of redundancies. It is a fact that even if high reliability components and equipments are used, the overall system reliability decreases with their number becoming large. The aim of this thesis, therefore, been to explore the field and scope of the second alternative.

Reliability allocation is a process of assigning reliability requirements to individual units to attain the desired system reliability. Thus the object of redundancy allocations, is to maximise the system reliability with cer.ain constraints such as cost, weight, volume etc. imposed on its application.

Before the allocation problem may be discussed and analysed, it is often necessary to know special features of reliability functions which will be the objective function of the optimisation process. The fifst chapter of the thesis is, therefore, devoted to the study of reliability function and its evaluation by observing special properties thereof. A method of flow-graph has been developed and illustrated with several numerical examples of different classes. It has been found to be of great help in quickly determining the reliability function for all types of reliability network with different types of components.

Non series-parallel networks usually present difficulty in the reliability evaluation. The Factoring Theorem suggested by Moscowitz [2] was the only existing technique for analysing such networks. The thesis therefore presents alternative computational approaches for the reliability evaluation of these networks. A systematic study of redundant networks yielded that they can be treated with the help of di-graph modelling and an algorithm could be developed for use on digital computer for large systems.

Once it is established that the reliability of any system could be increased by recourse to redundancies one usually faces the problem 'how much to apply'. One can go on increasing the reliability of a component by putting several units in parallel infinitely but there are always some inherent constraints such as cost, weight etc., that prevent one from doing so. It is no good to design a system 'too costly' or 'too heavy' to compensate for the system reliability. Generally, there should be some compromise between these factors.

It is with this view that the problem of maxmisation of reliability, under the constraints imposed by economical considerations, has to be thought of. Usually a problem of maximisation of reliability subject to cost, weight or volume, is considered.

There were several attempts [10, 11, 12, 13] to aim at this problem. Moscowitz and Mclean [12] considered the problem of maximisation of reliability with only one constraint, i.e. cost. Moscowitz [12] in fact used a variational method to come to an optimum allocation. Gordon [13] also considered the problem of single constraint. Kettelle [10] provided a computational approach for maximising reliability subject to 'cost' constraint only. However, Proschan and Bray [15] extended the method of [10] to include more than one constraint, viz. cost, weight etc. This required an approximate estimate of the reliability. The above approach has been applied in the thesis, for non-linear constraint problems also. Bellman and Dreyfus [16] formulated the problem as a Dynamic programming problem. The bulk of computation however in this formulation was too heavy even for a problem with few stages only. Fan and Tillman [21] proposed a method using discrete maximum principle but a slightly different problem formulation was used. They infact optimised the profit accruing out of a system with high reliability. Tillman[23] again used the Discrete maximum principle for the case of non-linear constraint problem and very rece tly Tillman [27] proposed an Integer programming approach to the problem of maximising reliability subject to several non-linear separable constraints and with different modes of failure.

Mizukami [26] formulated the allocation problem again as integer linear programming problem by approximating the concave objective function as linear between two variable x_j-points and further formulating it as linear programming problem. Muzukami infact used Mixed-linear programming technique for the solution.

A survey paper by Lawler and Wood [30] provided a new approach to the problem of non-linear programming. Based on [30] initial work has already been taken up and Jacobson

CHAPTER 1

ANALYSIS OF REDUNDANT NETWORKS

1.1. Introduction

It is a well-known fact that if a high reliability of a system is to be ensured, either the constituent elements of the system should have high reliability or the elements could be duplicated so that if one fails another ensures the failure-free operation of the system. This applies to all systems whether they happen to be mechanical, electrical, communication or information channels. This logic finds its application in electronic circuits for protection schemes, military application, space programmes, where reliability is of prime importance for their faultless operation. For example, Fig. 1(a) gives the circuit of a relay using a vacuum tube - the probability that the relay will operate when a signal appears at the grid terminals of the tube, is the reliability of the vacuum tube. If there happens to be an open circuit in the filament circuit, the failure of the system occurs because of non-operation of relay. Now to ensure even more reliable operation if we duplicate the tube, the system will remain operative even if there happens to be a failure of one of the tubes. The reliability of system now is increased $(2 - p)$ times the original reliability of the tube where p, is reliability of a tube given that $0 < p < 1$. The system with two tubes will be called Redundant system.

1.2. Definitions

Redundancy can be defined as the existence of more than one means of accomplishing a task. All means should fail before the system failure occurs. Obviously chanches of failure of a system are less

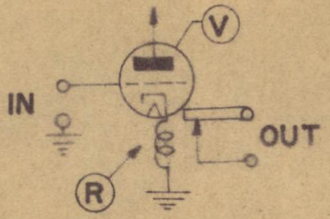
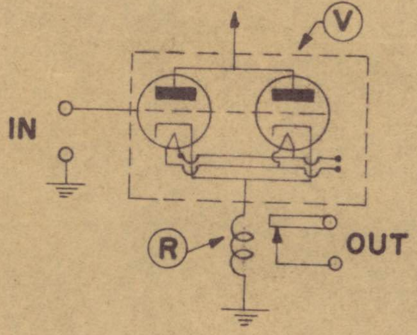


FIG. I (a) NON-REDUNDANT RELAY CIRCUIT USING ONE VACUUM TUBE.



V = VACUUM TUBE

R = RELAY

FIG. I (b) REDUNDANT RELAY CIRCUIT USING TWO VACUUM TUBES IN PARALLEL.

'cost' constraint only. However, Proschan and Bray [15] extended the method of [10] to include more than one constraint, viz. cost, weight etc. This required an approximate estimate of the reliability. The above approach has been applied in the thesis, for non-linear constraint problems also. Bellman and Dreyfus [16] formulated the problem as a Dynamic programming problem. The bulk of computation however in this formulation was too heavy even for a problem with few stages only. Fan and Tillman [21] proposed a method using discrete maximum principle but a slightly different problem formulation was used. They infact optimised the profit accruing out of a system with high reliability. Tillman[23] again used the Discrete maximum principle for the case of non-linear constraint problem and very rece tly Tillman [27] proposed an Integer programming approach to the problem of maximising reliability subject to several non-linear separable constraints and with different modes of failure.

Mizukami [26] formulated the allocation problem again as integer linear programming problem by approximating the concave objective function as linear between two variable x_j-points and further formulating it as linear programming problem. Muzukami infact used Mixed-linear programming technique for the solution.

A survey paper by Lawler and Wood [30] provided a new approach to the problem of non-linear programming. Based on [30] initial work has already been taken up and Jacobson

[31] has brilliantly worked out an algorithm using branch and bound method for minimising the cost of a system subject to maintaining a certain level of reliability. The author is also currently working on the same problem and hopes to bring out some fruitful results in future.

In short different investigators used different approaches to the problem of maximising the system reliability subject to specified constraints.

The present thesis aims at presenting few more aspects and computational approaches to the abore problem. The thesis also presents the comparative study of different approaches which is very much required by the system designer before any convenient solution to the problem is desired.

where there are redundancies or, in other words, the reliability of a system increases with the introduction of redundancy in a system.

Redundancies can be classified under three broad categories: Active redundancy, Standby redundancy and Voting redundancy.

In active redundancy all the redundant paths (units) are continuously energised while the system operates. If the redundant unit does not perform any function and comes into operation only when the primary unit fails, this type of redundancy is called standby redundancy. In such a redundancy system it is necessary to have some decision making device which will detect the failure of first unit and place the second unit into operation simultaneously. A standby unit may be partially or fully energised or completely inactive. In the third type of redundancy, three or more units operate in conjunction with a switch which selects the unit with agreeing outputs if they constitute a majority. This type of redundancy is commonly used in computer applications. The redundancies may be introduced at any level of a system, viz. component-part, component, unit (or equipment) system itself. This necessitates definitions of different terms used here.

Element or Component Part - This is a basic unit in any system, such as resistance, capacitance, diode, tube, transistor etc.

Component - Assembly of component parts forms a circuit, viz. oscillator, trigger cct, register etc.

Unit or Equipment - Next higher level of system assembly is an equipment or unit such as relays (static) etc.

System - A complete operating unit constituting of several equipments or units may be called as a system.

Redundancy may be introduced at any level in a system, i.e. component parts, a circuit, an equipment or a system itself may be duplicated. However, it is obvious that active redundancy in component parts such as resistors, capacitors are unsuitable because if one fails, out of, say, two parallel units then this changes the circuit constants. In such cases, standby redundancy may be resorted to if it becomes absolutely necessary. To make the analysis more general and depending on the level at which redundancy is introduced these terms may interchangeably be used. A block in reliability model will henceforth be called as an element and the whole assembly as a system.

1.3. Redundant Networks

After Shanon [1] suggested that a large number of less reliable relays could be connected in a lattice form to give more reliable operation, the attention of several investigators was drawn to the use of redundancies in several forms and to the evaluation of reliability of such networks.

Depending on the connections of different constituent elements in a system, three situations arise: the elements may be in series, parallel or in a non series-parallel form. Therefore in a broader perspective, all the networks can be divided into two categories:

a. Mixed redundancy or series-parallel configuration, in which the elements are connected to each other only in series and/or in parallel.

b. Non series-parallel configurations, which have not only series-parallel connections but also interconnecting elements such as in bridge networks. Because of these interconnecting elements it is not possible to call elements either being in series or parallel.

Non series-parallel circuits may be planar or non-planar which can be drawn only while crossing each other.

It can be shown very easily that a circuit of 4 elements of

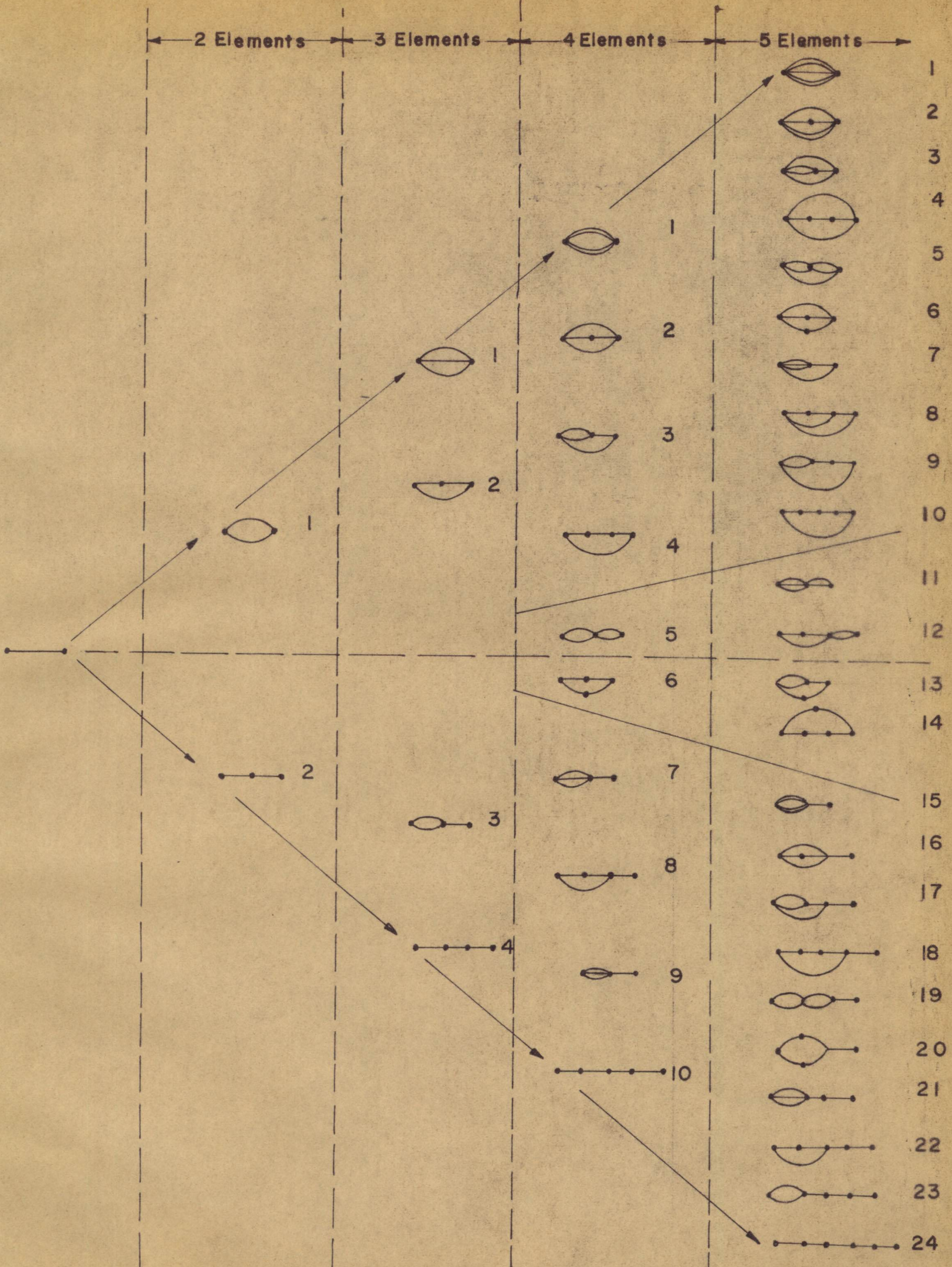


FIG. 2 Schematic Development Of Series-Parallel Configrations

two parallel paths with two elements in series can be made still more reliable by the introduction of an interconnecting link to make it a bridge circuit.

1.3.1. Series-parallel configurations

A schematic development of series-parallel configurations of like elements is given in Fig. 2. As is clear from the Fig. 2, the different possible configurations for four elements can be derived from those of three elements realising the fact that the new element could be placed in the following manner:

a. In parallel with the whole unit of three elements.

b. In series with the whole unit of three elements.

c. Introduced in branch path of the unit of three elements, in either parallel or in series with an individual element.

It is obvious that (a) and (b) just double the possibilities by the introduction of a new element; however (c) gives a definite number of possibilities only. It is also clear that the independent configurations contributed by (c) for a particular number of elements can be found from the configurations falling under the same group (c) of the preceding number (i.e. one short) of elements. The number of possible configurations upto seven elements are listed in Table 1.

Table 1 - Possible Series-parallel Configurations

Elements	2	3	4	5	6	7
Total number of configurations	2	4	10	24	66	180

If the configurations listed in Table 1 are classified on the basis of number of nodes they have, then Table 2 is obtained. It is evident from Table 2 that the maximum number of configurations lie in mean number of node's column and are almost equal to the total number has its dual drawn in Fig. 6. The procedure o1 drawing dual network is to take two terminals outside the network whose dual is to be found and then putting a node in each loop of the original network, lines can be drawn through all the elements joining the two proper nodes. The method is displayed in Fig. 5(b).

1.3.4. Development of reliability models

Before the reliability of a system consisting of several functional units is evaluated a representative model of the system is developed depending on how different constituent units interact as regards their functions to make a system operative. When this block diagram is developed it will fall in any of the above configurations discussed earlier. Thus knowing the reliability parameter of the units or in more common language the elements, the overall reliability parameter of the system can be obtained by the methods to be described later.

1.4. Analysis of Redundant Networks

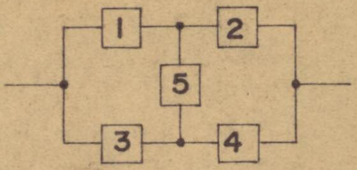
Some of the results desired from an analysis of redundant networks are:

a. The overall reliability for various kinds of redundancy, given the appropriate parameters of the elements of the network.

b. For particular subsystem should it contain several replicas in a redundant formulation or should a more reliable element be used by itself? One can make tradeoffs between reliability and various resources for this purpose. A typical cost vs reliability curve is shown in Fig. 7.

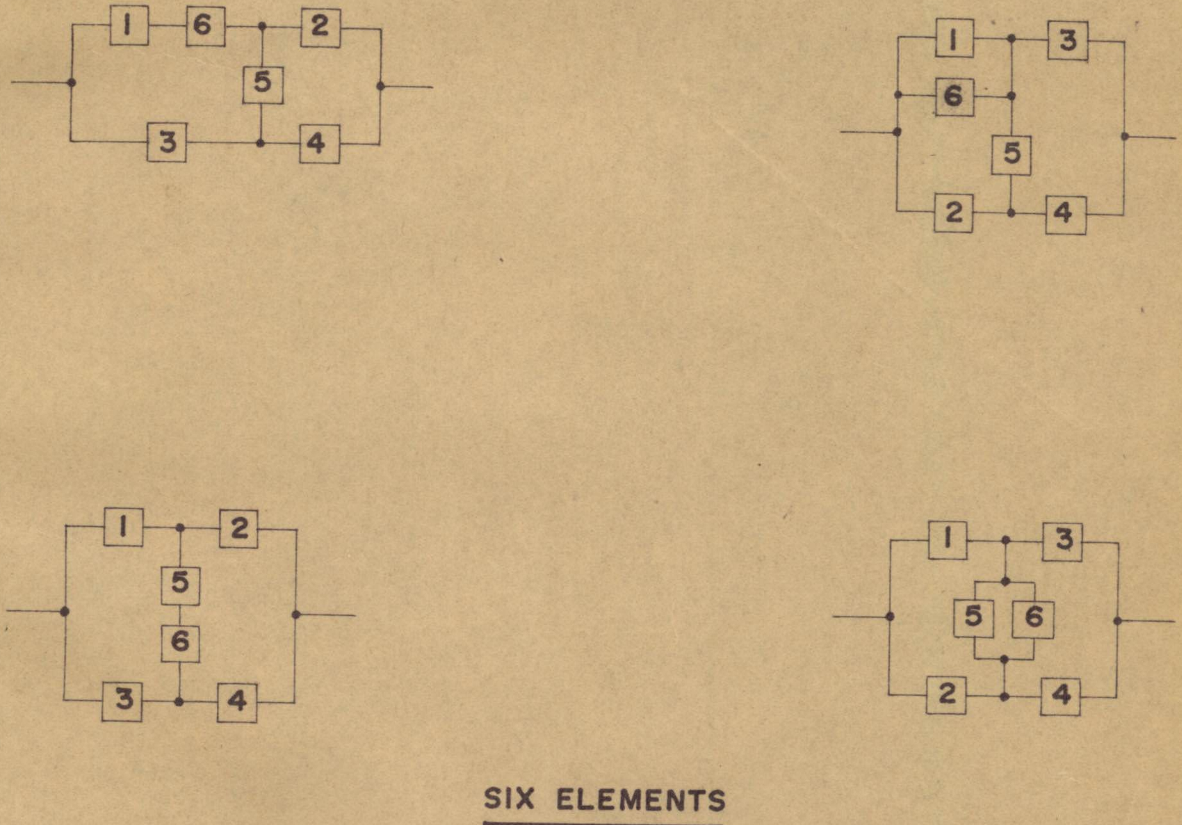
c. If the reliability of a system must be improved, on which subsystem should the effort be allocated?

d. The proper tradeoffs of reliability versus volume, weight, cost or other factors.



FIVE ELEMENTS

FIG.3 A BRIDGE NETWORK OF FIVE ELEMENTS.



SIX ELEMENTS

FIG. 4. BRIDGE NETWORKS FOR SIX ELEMENTS.

of configurations for the preceding number of elements case.

Table 2 - Distribution of Configurations on the basis of No. of Nodes and Elements

	No. of Configurations					
	No. of Elements					
No. of Nodes	2	3	4	5	6	7
2	1	1	1	1	1	1
3	1	2	4	6	9	12
4	-	1	4	10	23	44
5	-	-	1	6	23	66
6	-	-	-	1	9	44
7	-	-	-	-	1	12
8	-	-	-	-	-	1
Total	2	4	10	24	66	180

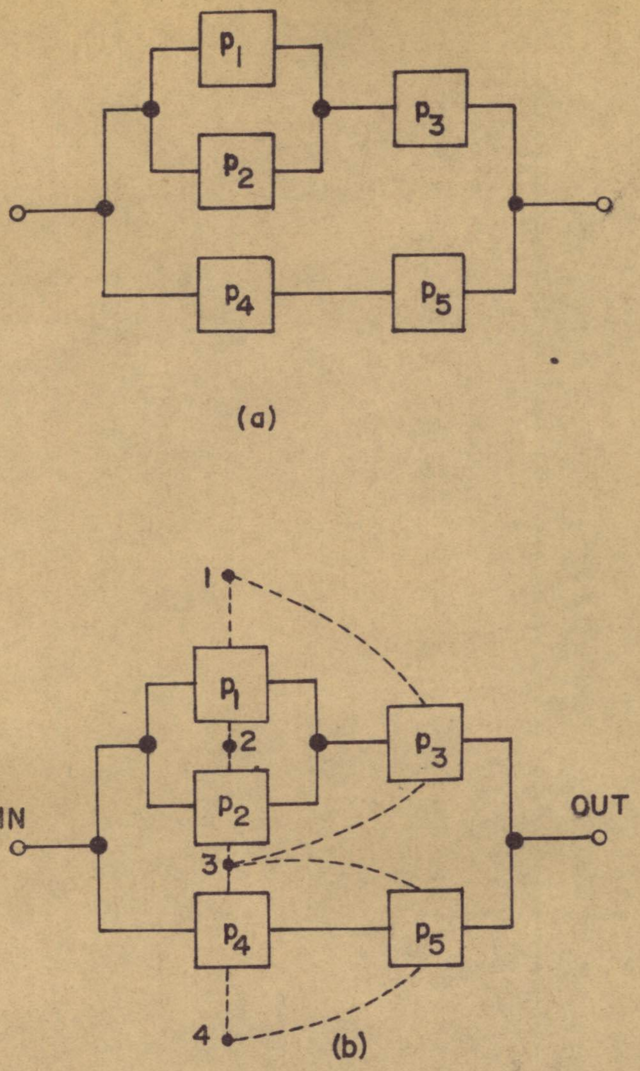
1.3.2. Non series-parallel networks

Fig. 3 shows a bridge circuit which is the first non series-parallel circuit that can be drawn with minimum of 5 elements.

It may be made clear that only independent configurations have been considered. However, in all these cases any particular element can take up all other positions of the elements. This would not change the approach of analysis of a particular configuration. The next non series-parallel configurations which can be drawn for 6 elements, are shown in Fig. 4. Further development is easier for a case of 7 elements and so on.

1.3.3. Dual networks

In fact all the network configurations shown in Fig. 2 can be grouped in two sections. The networks shown above the centre line have their image networks as their duals. For example, in case of 4 elements the configuration 4 has its dual as 7, and 2 has its dual as 9, etc. The method of drawing dual network is shown in Figs. 5 and 6. Fig.5(a)



(a)

(b)

FIG.5 Method Of Drawing Dual Configration.

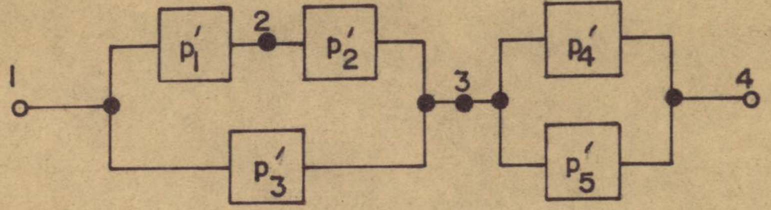


FIG.6 Dual Configration.

has its dual drawn in Fig. 6. The procedure oı drawing dual network is to take two terminals outside the network whose dual is to be found and then putting a node in each loop of the original network, lines can be drawn through all the elements joining the two proper nodes. The method is displayed in Fig. 5(b).

1.3.4. Development of reliability models

Before the reliability of a system consisting of several functional units is evaluated a representative model of the system is developed depending on how different constituent units interact as regards their functions to make a system operative. When this block diagram is developed it will fall in any of the above configurations discussed earlier. Thus knowing the reliability parameter of the units or in more common language the elements, the overall reliability parameter of the system can be obtained by the methods to be described later.

1.4. Analysis of Redundant Networks

Some of the results desired from an analysis of redundant networks are:

a. The overall reliability for various kinds of redundancy, given the appropriate parameters of the elements of the network.

b. For particular subsystem should it contain several replicas in a redundant formulation or should a more reliable element be used by itself? One can make tradeoffs between reliability and various resources for this purpose. A typical cost vs reliability curve is shown in Fig. 7.

c. If the reliability of a system must be improved, on which subsystem should the effort be allocated?

d. The proper tradeoffs of reliability versus volume, weight, cost or other factors.

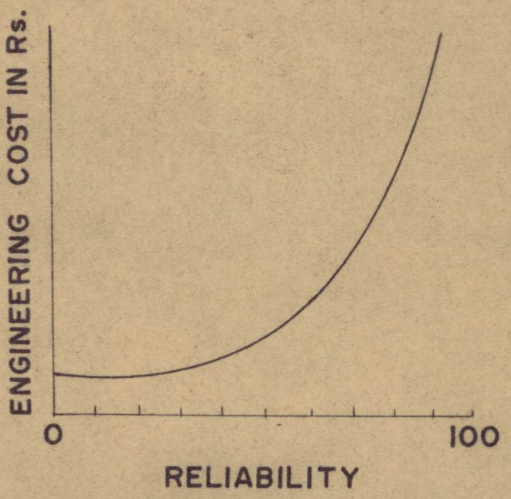


FIG. 7. A TYPICAL CURVE OF ENGINEERING COST Vs RELIABILITY.

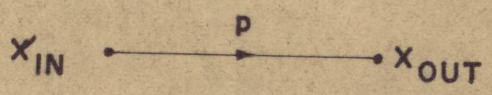


FIG. 8. AN ELEMENT.

In this chapter, the main concern is with (a) above. The solution to that problem is necessary for any of the subsequent results. The following assumptions are made:

a. All elements are always operating (no standby or switched redundancy).

b. The states of all elements are statistically independent. This means that the failure of one element does not affect the probability of failure of other elements.

c. Time is not explicitly an independent variable.

d. Each element may be represented as a two-terminal device.

e. The state of each element and of the network is either good (operating) or bad (failed).

1 4.1. Basic property of an element

An element in a reliability model of a system may be given a statistical parameter p such that it represents the probability of that element to survive under the specified condition of environment. Describing the same parameter in other words [2] in a physical sense, if X_{in} is the number of alike items with probability of survival p then X_{out} is the number of items expected to remain in operating condition after a certain time t. Therefore an element or a block may be represented by a two-terminal link with parameter as p having a linear relationship as -

$$X_{out} = p \; X_{in} \qquad \ldots(1)$$

Here the author differs with usual convention as described in [2], in that the element as represented by (1) must also be given a direction from 'IN' terminal to 'OUT' terminal (as shown in Fig. 8), so as to make it possible to extend topological methods for the analysis of redundant networks.

However, it may be made clear that in case of interconnecting links such as we come across in non series-pa allel configuration such an oriented graph would not be possible for these interconnecting links but as will be seen later such an eventuality can be byepassed by defining more than one oriented graph for the same network.

1.5.Historical Procedures

The problem of finding the overall reliability parameter knowing the reliability parameters of constituent elements becomes complicated and time-consuming when the system is large and complex. Each element can have either of the two states, i.e. either it is operating or has failed. Same applies to the system also i.e. either it will be operating or has failed. Therefore, the overall performance of the system is binary function of the element performance. Consideration of all combinational states of different elements multiplies the number of states for each element in the network. If the state 1 denotes the operative state of an element and state 0 represents the failure of that element then the number of states for three elements would be 8 and for a case of seven elements it will be 128 or in short 2^n for n elements. Further for a case of 20 elements it will be more than one million as was pointed out in [2] also. The system performance will be the summation of all the events leading to successful operation of the system.

Moskowitz [2] suggested breaking up of the large complex system into smaller units of series and parallel networks of the system and used dot and cross operators for systematic evaluation of the network function. No doubt, the system performance function can be easily written down using dot and cross operations defined as below:

Dot operation; $x.y = xy$

Cross operation; $x \times y = x + y - xy$

but actual evaluation is even tedious, for it involves many multiplications.

For bridge circuits, [2] suggested the use of factoring theorem. Factoring theorem states that if F $(p_1, p_2, p_3 \ldots p_n)$ is the reliability function of the network of n elements including an interconnecting link k whose reliability is pk, the overall function can be written as -

$$F\ (p_1, p_2, p_3 \ldots p_n) = pk\ \Big[F\ (p_1, p_2 \ldots p_n)\Big]_{pk\,=\,1} + \overline{pk}\ \Big[F\ (p_1, p_2 \ldots p_n)\Big]_{pk\,=\,0}$$

where pk is reliability of the element k and $\overline{pk} = (1 - pk)$. Again here if there are many such interconnecting links then for each link the number of series-parallel configuration of the same size as the number of elements in the original network would be doubled.

1.6 Some Properties of Reliability Expressions for a Redundant Network

Before discussing the topological method, author has developed, some of the properties of reliability polynomials will be given.

If each element has a probability of survival p, then the expression for the reliability of the network will be a polynomial in the various p's. Some of the properties of these reliability polynomials are -

a. The highest degree for any term is the number of elements in the network.

b. The sum of all the coefficients of the polynomial is unity

c. The coefficient for the term of highest degree is unity in case of series-parallel networks. For non series-parallel networks, it is the number of variations in the orientation of the graph of the interconnecting links as will be discussed later.

d. The sign of the highest degree coefficient will be positive if there is an even number of loops (zero is an even number). For

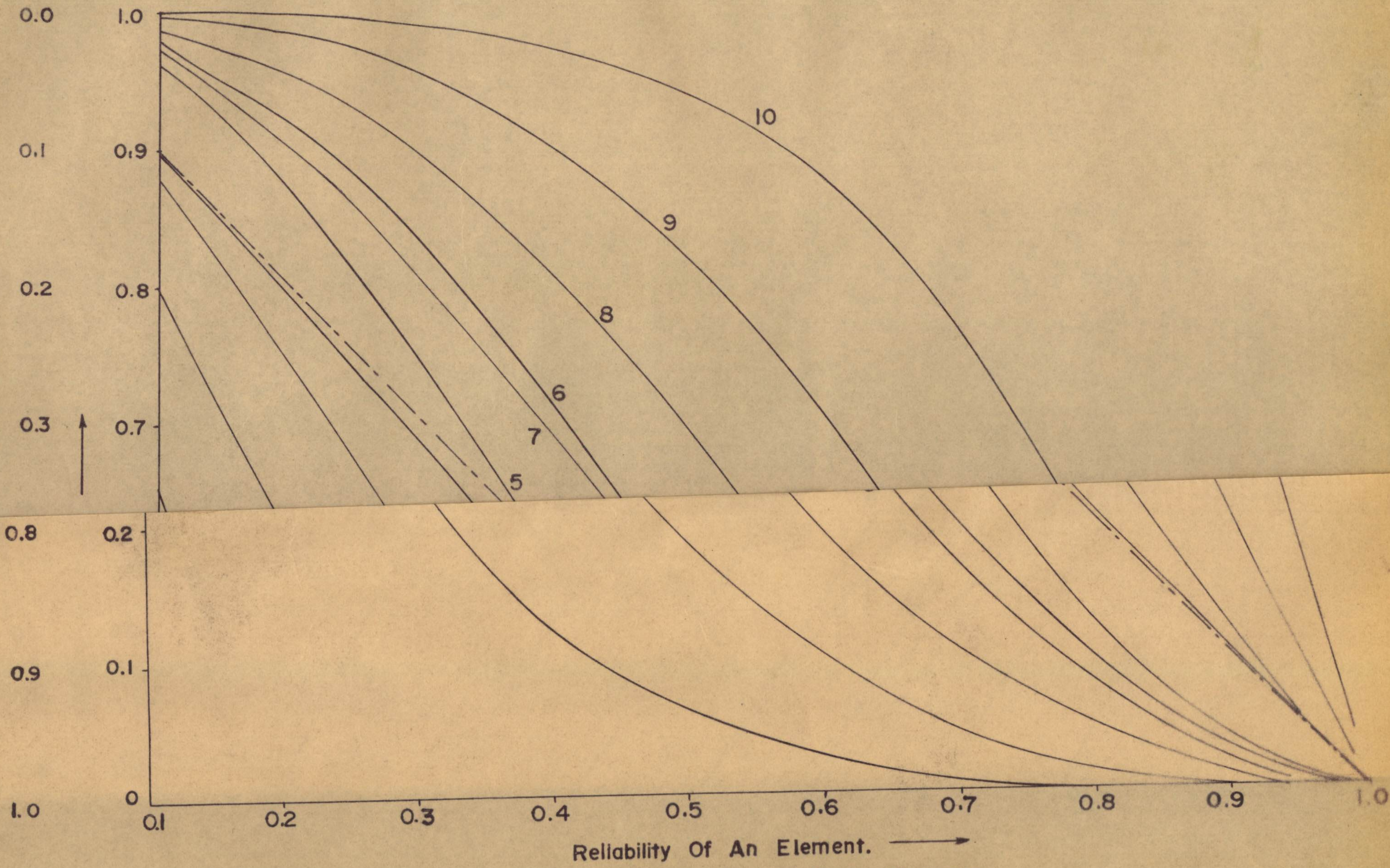


FIG.-9. Relaibility Curves Of Different Configurations Of Four Elements.

an odd number of loops the sign of the highest degree coefficient will be negative. (Actually in the strictest sense we cannot have any loops in the oriented graph of a network in reliability modelling as will be seen later; the complete oriented graph turns out to be a cascaded graph.)

e. The sum of the number of nodes and loops in a network will be equal to N + 1, (N is the number of elements in the network).

f. The sum of the number of nodes in a network and in its dual network will be N + 3, (N is the number of elements in both networks).

g. Let the parameter of each element be the same, p. Then the reliability polynomial when plotted against p will be S-shaped if there is no single element in series or parallel overall. This means that for some range of p, the network will be more reliable than a single element and for some other range of p, the network will be less reliable than a single element. This is clear from Fig. 9 drawn for the case of 4 elements. Chained line in Fig. 9 shows the curve when there were only one element. The curves of the polynomial corresponding to configurations 1, 2, 3, 4 and 7, 8, 9, 10 of Fig. 2 are either below this line or above this line respectively and decreasing monotonically but configurations 5 and 6 exhibit a migratory tendency or S-shapedness i.e. for certain range of probability of success of an element the network may be better than a single element in reliability and for another range of element reliability the network may be worse than a single element in reliability. If we trace back then we realise that these networks were obtained by introducing an element in the branch in place of putting the element either in overall series or parallel while going from

3 elements network to 4 elements netwoi... For 5· elements networks, S-shaped curves will be for configurations 11, 12, 13 and 14 only (Refer Fig. 2). Another interesting thing about these curves is that crossing point with chained line can be obtained at any point by choosing a proper network. For example, in case of 6 elements case, these configurations will be 18 and for 7 elements they will be 48 in number and they can provide any range of crossing points.

1./7. Flow-graph Method

In Section 1.4.1 the property of an element was given and it was pointed out that an element must be oriented for flowgraph analysis. Therefore if an element is to have transmittance p_{ij} when the element is connected between nodes i and j, the signal must be 'in' at the terminal i and 'out' at the terminal j, to recognise it as an oriented graph. Since in the analysis of redundant circuits we are mainly concerned with the evaluation of transmittance between two (or otherwise specified) terminals, all elements must be oriented such that they seem to carry a signal from the IN (source) terminal to the OUT (sink) terminal. A source node will have only outgoing branches and the sink node, only incoming branches. This convention should be followed while orienting the branches of a graph.

With such an assumption for series-parallel redundant networks, the resulting oriented graph turns out to be a cascade flow graph since any cascade sequence of coefficients always cascades into a new variable. There will be no feedback loops.

Since the variables at each node have the same dimension, the application of topological methods becomes easier. The ordinary multiplication and addition rules of linear flow-graphs can be applied. When two elements are in series with probability of success p_1 and p_2, and the failures are statistically independent then the total transmittance (reliability) is p_1p_2. In general for m elements in series the total transmittance, Tr, will be -

$$Tr = \prod_{i=1}^{m} p_i \qquad (2)$$

Also when two elements are in parallel the total transmittance will be $p_1 + p_2 - p_1p_2$, or in general for n elements

$$Tr = 1 - \prod_{i=1}^{n} (1-p_i) \qquad (3)$$

One can use a Boolean sum of events, to give the formula

$$Tr = Pr\left\{\bigcup_{i=1}^{n} E_i\right\} \qquad (4)$$

where E_i is the event i^{th} element is good. Therefore the solution of redundant networks can be found straightforwardly by finding all possible forward paths in an oriented graph of the network and then summing them for the transmittance between the IN and OUT terminals according to (4) using the Boolean algebra rules. Remembering the basic Boolean rules the expansion of the terms into algebraic sums could be done. For example, considering the configuration of Fig. 10, all possible forward paths in oriented graph will be

$$E_6,\ E_7,\ E_4 \cap E_5,\ E_1 \cap E_2\ E_3 \cap E_4 \qquad (5)$$

Then

$$Tr = Pr\left\{E_6 \cup E_7 \cup (E_4 \cap E_5) \cup (E_1 \cap E_2 \cap E_3 \cap E_4)\right\} \qquad (6)$$

This is the transmittance between terminals IN and OUT of the network, i.e. the reliability. Equation (6) could be expanded by the usual laws of probabilities of statistically independent events (remember that the $[E_i]$ are independent). The number o- terms becomes very large, for 7 forward paths the total number of terms will be 127 - of course many of them would combine.

1.7.1. A speedy method of analysis by inspection

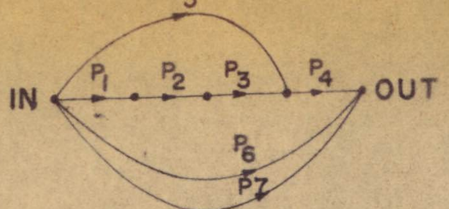
For the series-parallel case of 7 elements of Fig. 10, the method described below gives all 15 terms directly without any mathematics involved, by inspection and following certain rules. This is the easiest approach - the analysis and speedy solution of the problem is without any tedious manipulations.

a. Find out, one by one, all the possible forward paths available. The maximum number of elements in any forward path will not be more than one short of the number of nodes assuming that (i) there is at least one path which contains all the nodes or (ii) all nodes are interconnected as may be the case in non series-parallel networks. Find their sum.

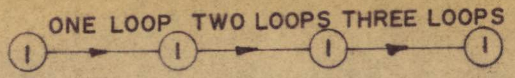
b. Find all oriented graphs touching IN and OUT terminals containing one loop only. Assign negative sign to the sum of product of the probabilities of success of those elements which constitute a particular graph. For example, in Fig. 2, the 6th graph has only one loop (actually in the language of flow graph, this cannot be called a closed loop) and consists of elements 4, 5, 7. Therefore this gives rise to a term $p_4p_5p_7$ with negative sign.

c. Next, we find all oriented graphs again touching IN and OUT terminals having two loops and sum their products of probabilities of success; attach a positive sign.

d. Steps 2 and 3 are repeated for all loops until the graphs that contain the maximum number of loops have been considered. An odd number of loops gets a minus sign, the even numbers get a

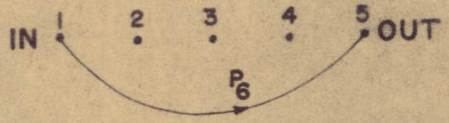


SIGN RULE

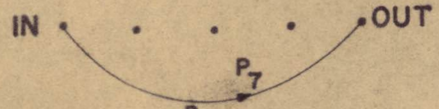


IF THE ELEMENTS HAVE EQUAL PROBABILITY OF SUCCESS.

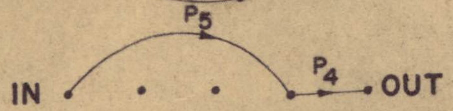
IF THE ELEMENTS HAVE DIFFERENT PROBABILITIES OF SUCCESS.

1. 

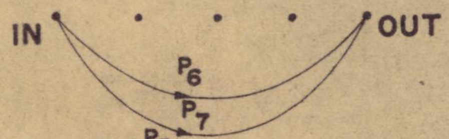
p p_6

2. 

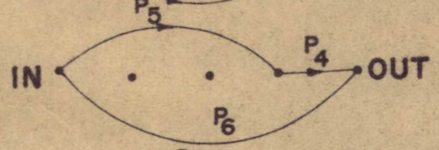
$+p$ $+p_7$

3. 

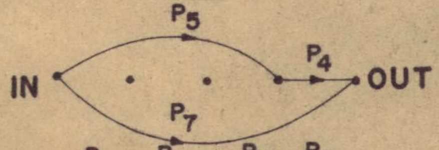
$+p^2$ $+p_4 p_5$

4. 

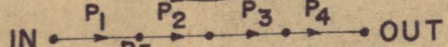
$-p^2$ $-p_6 p_7$

5. 

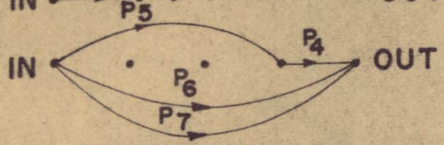
$-p^3$ $-p_4 p_5 p_6$

6. 

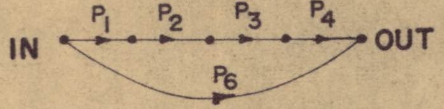
$-p^3$ $-p_4 p_5 p_7$

7. 

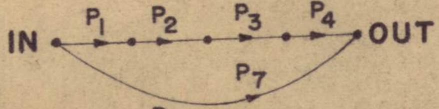
$+p^4$ $+p_1 p_2 p_3 p_4$

8. 

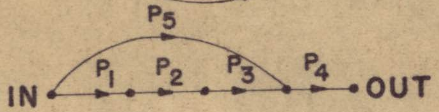
$+p^4$ $+p_4 p_5 p_6 p_7$

9. 

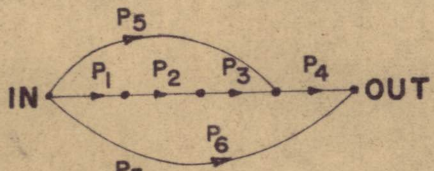
$-p^5$ $-p_1 p_2 p_3 p_4 p_6$

10. 

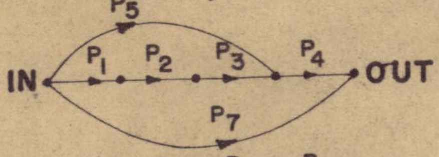
$-p^5$ $-p_1 p_2 p_3 p_4 p_7$

11. 

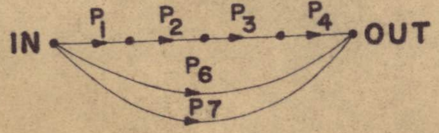
$-p^5$ $-p_1 p_2 p_3 p_4 p_5$

12. 

$+p^6$ $+p_1 p_2 p_3 p_4 p_5 p_6$

13. 

$+p^6$ $+p_1 p_2 p_3 p_4 p_5 p_7$

14. 

$+p^6$ $+p_1 p_2 p_3 p_4 p_6 p_7$

15. WHEN ALL ARE PRESENT $-p^7$ $-p_1 p_2 p_3 p_4 p_5 p_6 p_7$

FIG. 10. TOPOLOGICAL METHOD.

plus sign. The maximum number of loops in any network will be one plus the number of elements minus the number of nodes. The above procedure is so simple that one can write the complete reliability polynomial or transmittance without difficulty or mistakes. All 15 steps for the problem of Fig. 10 are shown thereon.

The procedure can be programmed and successfully performed with a computer for a large complex network if the sole purpose is to evaluate transmittance. Although with a computer any of the methods may be used with ease, the method just described is recommended because it only requires the information as regards the connection of different elements to particular nodes, i.e. connection matrix. No other information or manipulation is necessary. Therefore the method described has an edge over other methods. Before applying the above procedure the network can first be reduced by combining parallel elements across any two particular nodes. The author used the above method and found it successful. The flow chart of the computer algorithm is shown in Fig. 11 (NN is the number of nodes).

In the first part of the program, the element reliabilities and the nodes to which the elements are connected are stored in a table. Next the elements of the connection matrix of the order NN × NN are made zero, and a reduced matrix (NN × NN) is prepared from the stored table with the help of a Subroutine Reduce which combines all the parallel elements across any two nodes. For example, the connection matrix developed for the configuration of Fig. 10 will be of the form:

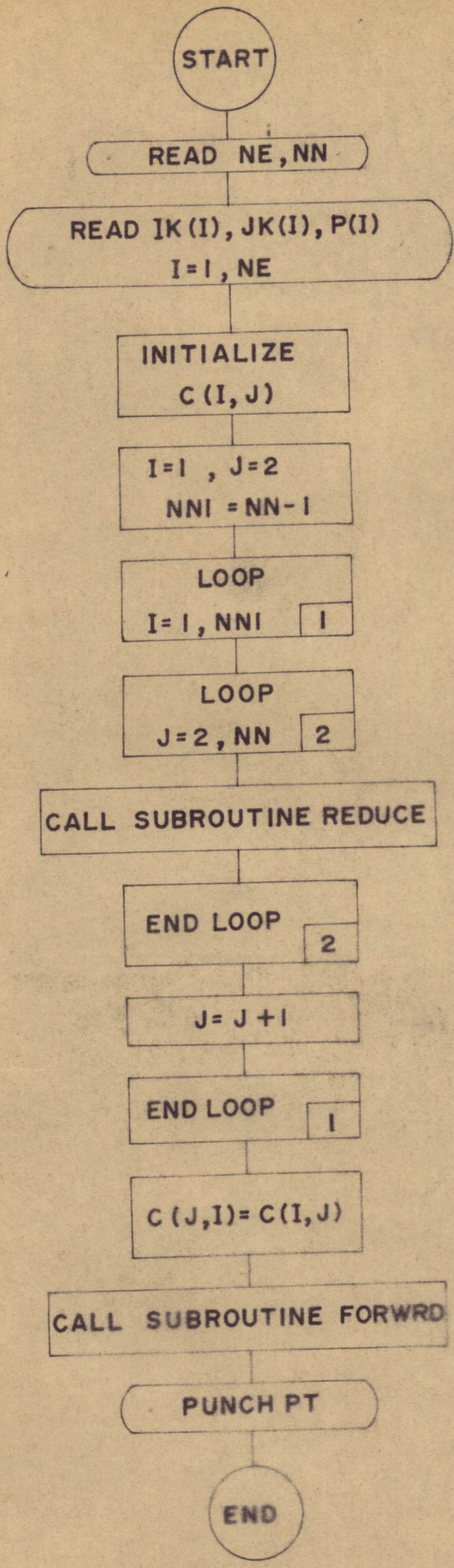


FIG.11. FLOW CHART FOR TOPOLOGICAL METHOD.

$$C\,[5,\ 5] \equiv \begin{bmatrix} 0 & c_{12} & 0 & c_{14} & c_{15} \\ c_{21} & 0 & c_{23} & 0 & 0 \\ 0 & c_{32} & 0 & c_{34} & 0 \\ c_{41} & 0 & c_{43} & 0 & c_{45} \\ c_{51} & 0 & 0 & c_{54} & 0 \end{bmatrix} \qquad (7)$$

where $c_{12} \equiv p_1$, $c_{14} \equiv p_5$, $c_{15} \equiv (p_6 + p_7 - p_6p_7)$ as obtained from Subroutine Reduce. Once the connection matrix is developed, Subroutine Forwrd finds all possible paths from nodes 1 to NN. The upper diagonal elements of matrix C take care of the formation of a forward path, first with one loop, next with two loops and so on depending on the number of terms in any particular row in the lower diagonal. The procedure followed is exactly as described above: All the products of probabilities during these walks are added with proper sign to give the transmittance between nodes 1 and NN.

This method necessitates that the numbering of the nodes be in ascending order - a condition for the network to have a cascaded digraph. All elements should be oriented from lower node number to the higher node number. This is not difficult to achieve in practice.

1.7.2. Non series-parallel redundant networks

Non series-parallel networks differ from others, in that there are interconnecting elements which are bilateral in nature, viz. they are oriented in both directions. It was observed earlier in series-parallel networks that by properly orienting the graph, it turned out to be a cascade flow graph. The IN node of each element has its serial number less than the OUT node and all graphs are oriented from lower node number to higher node number. But in non series-parallel networks it may not be so, due to the interconnecting elements. This problem can be solved by what may be called superposition. Again, writing down the transmittance will be easier than any other method. Except for the interconnecting elements all other elements have a fixed orientation.

Take the example of the bridge network of Fig. 12. Element 5 is an interconnecting element and cannot be given any fixed orientation. Now since element 5 may be oriented in either direction, two separate networks with all other elements having their orientation the same, except that of 5, are developed as shown in Figs. 14 (a) and (b). The solutions of these two networks by graph theory are found separately.

The di-graph14a has forward paths 12,34 and 154; similarly, the graph 14b has forward paths 12, 34 and 352. The paths of 6 are:

$$\begin{aligned} \text{Path}_1 &= (E_1 \cap E_2) \cup (E_3 \; E_4) \cup (E_1 \cap E_4 \cap E_5), \\ \text{Path}_2 &= (E_1 \cap E_2) \cup (E_3 \cap E_4) \cup (E_2 \cap E_3 \cap E_5) \end{aligned} \tag{8}$$

The total transmittance of the network 12 is

$$\begin{aligned} \text{Tr} &= \Pr\,[\text{Path}_1 \cup \text{Path}_2] \\ &= \Pr\left[(E_1 \cap E_2) \cup (E_3 \; E_4) \cup (E_1 \cap E_4 \cap E_5) \cup (E_2 \cap E_3 \cap E_5)\right] \end{aligned} \tag{9}$$

One must take precaution while applying the method of inspection and tracing out the paths, that no path having an element oriented backwards can be taken, since that violates the properties of cascade flow graphs. All elements directly connected to source and sink must be properly oriented.

While orienting an interconnecting element one must not direct it so that a closed loop is formed because the graph would then not be a cascade flow graph.

For the problem of Fig. 15 (dropping the letter E from the event notation and implying intersection by the grouping) the paths for oriented graphs of Fig. 16(a), (b) and (c) are

$$\begin{aligned} a &= 147 \cup 123 \cup 67 \cup 1257 \\ b &= 147 \cup 123 \cup 67 \cup 356 \cup 1453 \qquad (10a) \\ c &= 123 \cup 67 \cup 6423 \cup 356 \end{aligned}$$

The total paths are

$$a \cup b \cup c = 147 \cup 123 \cup 67 \cup 356 \cup 1257 \cup 1345 \cup 2346 \qquad (10b)$$

The transmittance is the probability of this combined event. The probability can be calculated as mentioned above (probabilities of terms taken by ones, threes, fives, and sevens are positive; the others are negative).

$$\begin{aligned} Tr = {} & 67 + 123 + 147 + 356 + 1257 - 1467 - 3567 + 2346 \\ & + 1345 - 12345 - 12346 - 12347 - 12356 - 12357 \\ & - 12367 - 12457 - 12567 - 34567 - 23456 - 13456 \\ & - 13457 + 2(123456) + 2(123457) + 2(123567) + 2(123467) \\ & + 124567 + 134567 + 234567 - 3(1234567), \qquad (11) \end{aligned}$$

where now the p's have been dropped and the numbers 1-7 stand for the probabilities of individual events (e.g. $67 \equiv p_6p_7$). If the elements have equal probabilities of success, p, (11) will be

$$Tr = p^2 + 3p^3 + p^4 - 12p^5 + 11p^6 - 3p^7, \qquad (12)$$

satisfying the condition that Σ coefficients = 1 as indicated earlier.

1.8.Networks with elements that can short or open

In the preceding sections, we have considered situations in which the failure of an individual element or a path failure had no effect on the operation of the remaining elements or paths. In a situation where an individual element can fail in either of the two ways, viz. open circuit or short circuit, the analysis will be slightly different. An example of an element that can short or open is a diode. The failure in either way affects the operation of the surviving elements.

Since a single element fails by open or short circuit but not by both, open and short circuit failures are mutually exclusive events. Denoting q_o and q_s as the probabilities of open and short respectively, the total probability failure q is

$$q = q_o + q_s \tag{13}$$

subject to the condition

$$o \leqslant q \leqslant 1, \; o \leqslant q_o \leqslant 1 \text{ and } o \leqslant q_s \leqslant 1$$

There have been only a few references [4, 5] where series-parallel configurations of such networks have been considered. In any redundant network of the above combinations, the analysis would be too tedious to argue out based on the analysis that has been described in the above references. However, an easy method based on flow graph approach is very convenient for any network consisting of the elements that can fail either by open or short circuit.

1.8.1.Paths and cuts

In any of the two terminal networks considered earlier, the overall reliability has been computed by finding all possible paths from source node to sink node and then adding up the events using

Boolean algebra rule and the probabilities associated with them. For successful operation of the system , successful operation of each element forming a path is necessary.

With each parth A_j, j = 1, 2, . . . r, say, a binary function may be written as

$$\alpha_j(x) = \prod_{i \in A_j} x_i \tag{13}$$

which will take the value of 1 if all elements in the path function successfully. It is also obvious from (13) that all elements of such a j^{th} path act in series. Assuming a performance probability distribution of the elements such that

$$p_i \equiv P\left[x_i = 1\right] \equiv E\left[x_i\right]$$

where p_i is reliability of i^{th} element and x_i is the binary random number denoting the state of the element i, the probability for successful operation of a path would be given by

$$P\left[\alpha_j(x) = 1\right]$$

and the reliability of the system could be written as

$$R = P\left[\varphi(x) = 1\right] \tag{14}$$

where $\varphi(x) = 1 - \prod_{j=1}^{r}\left[1 - \alpha_j(x)\right]$ which gives the probability of successful operation of a system. Similarly, there are elements in any network if failed, would render a system as failed. Such elements are called cuts. Thus any cut B_k, k = 1, 2, . . .s, say, again a binary function could be written as

$$\beta_k(x) = 1 - \prod_{i \in B_k}(1 - x_i) \tag{15}$$

which takes the value 0 if all elements in k^{th} cut fail and 1 other-

otherwise.

1.8.2. Open and short circuit failures

Now the properties of paths and cuts could be used for the analysis of short circuit and open circuit failures of a system.

A path of a system, elements of which could short can only fail if all the elements constituting a path short. Similarly, a cut of a system, elements whereof could open would only fail if all the elements constituting a cut open.

Keeping above points in view one can redefine paths and cuts such that probability of short circuit failure associated with a path j,

$$q_{sj} = P\left[\alpha_j (x_s) = 1\right] \tag{16}$$

Obviously, the total probability of a system failing due to short circuit will be given by

$$q_s = 1 - \prod_{j=1}^{r} (1 - P\left[\alpha_j (x_s) = 1\right] \tag{17}$$

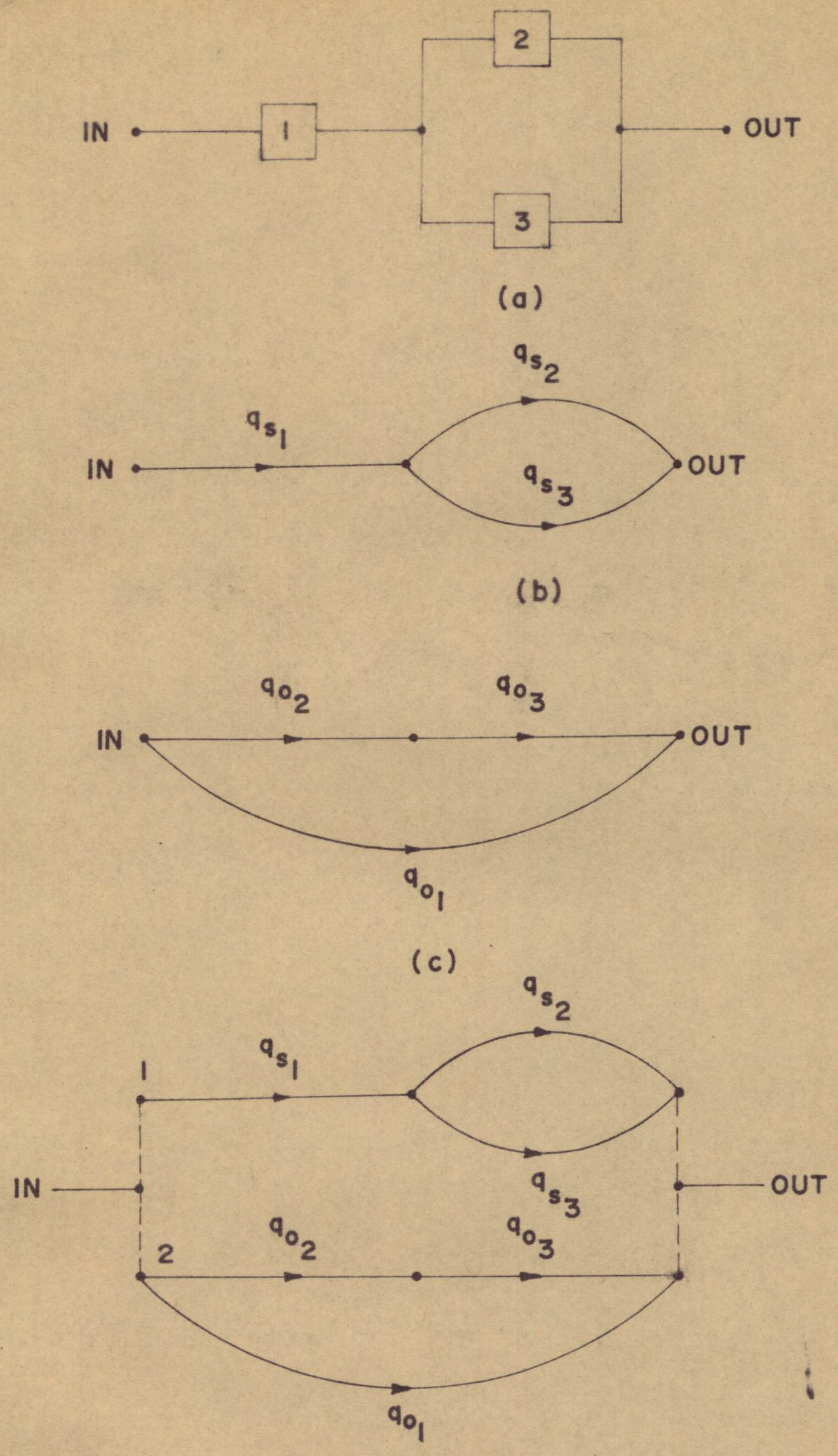
Similarly, the total probability of system failing due to open circuit, constituent elements of which could open, can be written, through concept of cuts, as

$$q_o = \prod_{k=1}^{s} P\left[\beta_k (x_o) = 0\right] \tag{18}$$

Applying the methods discussed earlier an example of Fig. 17a consisting of three elements which can either open or short.

The flow diagram for paths for the consideration of short circuit failures would be as shown in Fig. 17b. The total probability of failure of the system due to short circuit will be given by

$$q_s = 1 - \left[(1-q_{s1}q_{s2})(1-q_{s1}q_{s3})\right] \tag{19}$$



(d)

FIG. 17. SYSTEM WITH OPEN AND SHORT CIRCUIT FAILURES.

Using the method of inspection described earlier q_s can also be written as

$$q_s = q_{s1}\ q_{s2} + q_{s1}\ q_{s3} - q_{s1}\ q_{s2} q_{s3} \tag{20}$$

Again the flow-diagram for cuts for the consideration of open circuit is given in Fig.17c. It may be noted here that while applying topological method it is easier to write all possible paths quickly, therefore one can write down the cuts of a system by finding the paths of a dual network of the original network.

Thus the probability of failure due to open circuits of the elements can be written as

$$q_o = q_{o1} + q_{o2}\ q_{o3} - q_{o1}\ q_{o2}\ q_{o3} \tag{21}$$

As the open circuit and short circuit are two mutually exclusive events, the probability of failure of the system due to these will be algebraic sum of the probabilities associated with these two events, i.e.

$$q = q_s + q_o$$

Graphically, the situation is as shown in Fig. 17d. The branch 1, of the di-graph of Fig. 17d considers the short circuit failures and branch 2, the open circuit failures. The topological method can be applied directly to find q from Fig. 17d once it is drawn for system of 17a.

Finally, the reliability of the system of which elements can short or open then can be written as

$$R = 1 - q$$

To distinguish between mutually exclusive events and otherwise in a di-graph we may use dotted lines for the former and firm lines for the latter. Such a situation is shown in di-graph of Fig. 17d. This approach will be found to be very convenient in case of complex networks.

1.9. An algorithm for Direct Reliability Evaluation using Di-graph Matrices

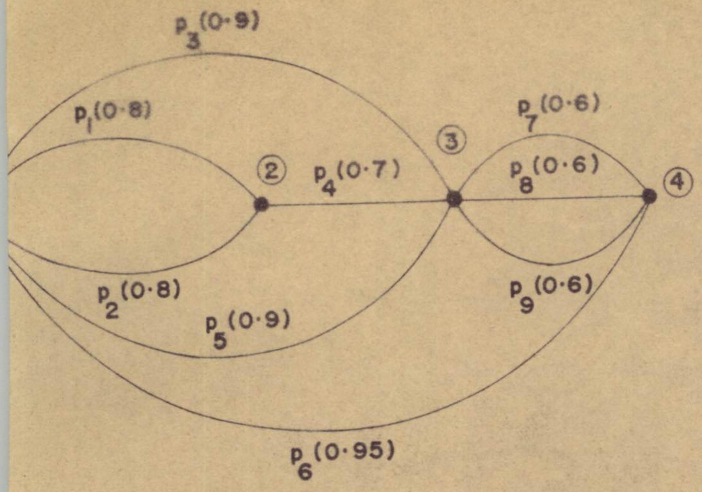
An algorithm is presented in the following sections for direct evaluation of reliability of series parallel and non-series parallel networks using di-graph matrices. This will be especially suitable on digital computers for larger and complex networks. The algorithm is quite fast and programming is fairly simple.

1.9.1. Modelling of networks

As discussed in section 1.4.1, the modelling of a redundant network can be done to represent it as a di-graph with 'IN' (source) and 'OUT' terminals. A source-node will have out-going branches and the sink-node, incoming branches only. Any element between i - j terminals is given a transmittance p_{ij} which is reliability of that element. For example, di-graph for a series parallel network of Fig. 18a is shown in Fig. 18b. For non-series parallel network, the same technique is observed except that an interconnecting element is replaced by two links with equal transmittances P_{ij} between node i and j, one oriented from i to j, the other from j to i. Such a di-graph for a simple non-series parallel network of Fig. 19a is shown in Fig. 19b. Here an obvious assumption will be made that both the oriented links, i.e. from i to j, and j to i, cannot exist together simultaneously and that probability associated with such an event is zero.

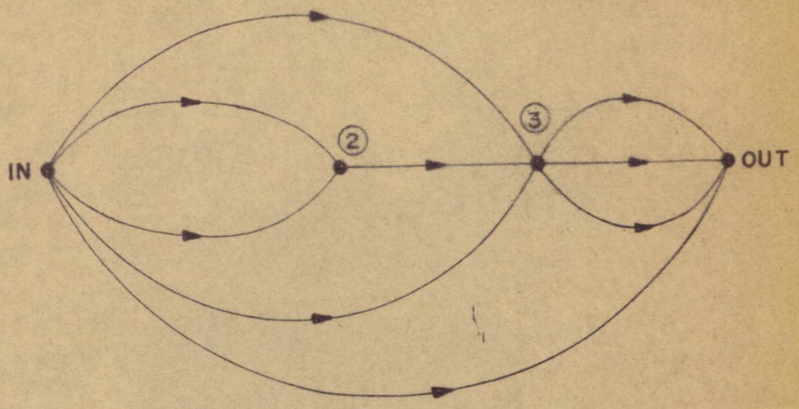
1.9.2 Combination of parallel elements

Before proceeding to evaluate the overall reliability of



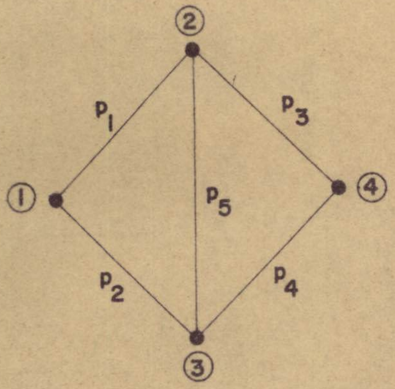
A SERIES PARALLEL NETWORK

FIG.18(a)



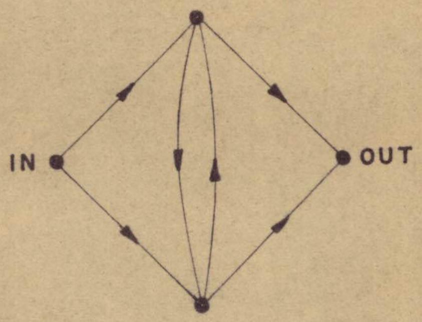
A DI-GRAPH OF A SERIES PARALLEL NETWORK.

FIG.18(b)



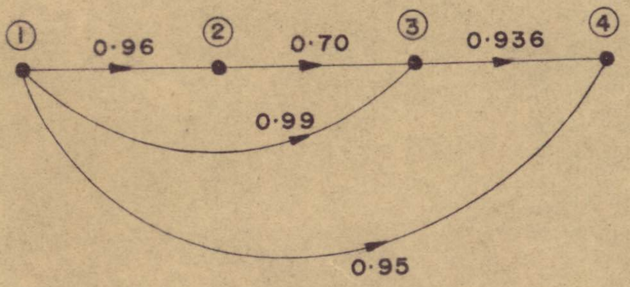
A NON-SERIES PARALLEL NETWORK

FIG. 19(a)



DIRECTED GRAPH FOR THE BRIDGE NETWORK OF FIG.19(a)

FIG.19(b)



REDUCED NETWORK.

FIG.20

redundant networks it is usually advantageous to combine all parallel elements between nodes i and j using boolean algebra rules and replace them by an equivalent link having reliability as c_{ij} connecting nodes i and j. If there are n parallel elements

$$c_{ij} = \Pr \left\{ \bigcup_{k=1}^{n} E_k \right\} \tag{22}$$

where E_k is the event that the k^{th} element is good.

Alternatively,

$$c_{ij} = 1 - \prod_{k=1}^{n} (1 - p_{ij}) \tag{23}$$

As a matter of fact, this can be done as soon as the data about the system or network is 'read' in the computer. The data about the network can be fed in a tabular form as given below:

```
(IK(I), JK(I), P(I), I = 1, NE)
```

where NE is the total number of elements in the network, IK and JK are the nodes having i^{th} element with reliability as P(I). The computer then scans the table 'read' and the elements with common nodes are combined together and stores an equivalent reliability link between nodes IK and JK while removing the nodes and elements from the table that have been combined to quicken the scanning. This process is repeated for all possible combinations of the nodes of the network till, finally, a weighted connection matrix [C] is obtained with the property that for any non-zero entry in [C] there exists one and only one branch between any two nodes. Initially all elements of [C] are initialised to zero and therefore only non-zero entries are transferred to [C]. A portion of main program (in FORTRAN) and the subroutine which tests for the parallel

```
C  C  K.B.MISRA.  MAIN PROGRAM
      DIMENSION IK(10),JK(10),P(25),C(10,10)
      COMMON  NN,NE,P,IK,JK
      READ100,NN,NE
100   FORMAT(2I3)
      READ200,(IK(I),JK(I),P(I),I=1,NE)
200   FORMAT(5(2I2,F10.6))
      DO1I1=1,NN
      DO1J1=1,NN
1     C(I1,J1)=0.
      JJ=2
      NN1=NN-1
      DO2I1=1,NN1
      DO3J1=JJ,NN
      CALL TESTPR(I1,J1,PO)
3     C(I1,J1)=C(I1,J1)+PO
      JJ=JJ+1
2     CONTINUE

      SUBROUTINE TESTPR(I1,J1,PO)
      DIMENSION IK(10),JK(10),P(25)
      COMMON NN,NE,P,IK,JK,PO
      PO=0.
      DO1 I=1,NE
      IF(IK(I)-I1)1,2,1
2     IF(JK(I)-J1)1,3,1
3     PI2=P(I)
      IK(I)=0
      JK(I)=0
      QO=1.-PO
      PO=PO+QO*PI2
1     CONTINUE
      RETURN
      END
```

FIG.23 A PROGRAM FOR NETWORK REDUCTION

branches and combines them, are given in Fig. 23.

1.9.3. Series parallel networks

Once the parallel branches have been grouped together and a weighted adjacency matrix [C] is developed, the equivalent network will be having less number of branches and will be equal to the non-zero entries of [C].

For example the matrix [C] for the network of Fig. 18 will be

$$[C] \equiv \begin{bmatrix} 0 & c_{12} & c_{13} & c_{14} \\ 0 & 0 & c_{23} & 0 \\ 0 & 0 & 0 & c_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad (24)$$

where $c_{12} \equiv p_1+p_2-p_1p_2$; $c_{13} \equiv p_3+p_4-p_3p_4$;

$c_{34} \equiv p_7+p_8+p_9-p_7p_8-p_7p_9-p_8p_9+p_7p_8p_9$ etc.

This matrix will automatically be developed by the computer by scanning the table (fed-in as data), again and again for each term as described in section 1.9.2.

The reduced network corresponding to (24) will be as shown in Fig. 20, with the values of the corresponding probabilities indicated.

To make further progress in the process of evaluation of total transmittance between terminals 1 and 4 we will make use of certain properties of a di-graph. What we actually desire finally, is an equivalent edgeconnecting nodes 1 and 4. This can be achieved if we can somehow eliminate the intermediate nodes; for Fig. 20 these will be, nodes 2 and 3. In series parallel networks the elements can either be in series or in parallel. Fig. 20 obtained after reduction (combining the parallel elements only) does not contain any two or more edges across a pair of nodes and as a matter of fact it should not if all parallel edges have been combined. The only possibility that exists is: there is at least one such node to which only two edges are connected, one is incident to and the other will be incident from the node.

Node 2 in Fig. 20 satisfies this condition. This type of node can be called as series-node and will be the first to be eliminated from the reduced di-graph. The equivalent edge between nodes 1 and 3 corresponding to the two edges 1-2 and 2-3 will have a probability value associated as obtained by multiplying the elements c_{12} and c_{23} of matrix [C]. Since the probability P_{series} associated with the event that m elements in series operate successfully, is

$$Pr\left\{ \bigcap_{i=k}^{m} E_k \right\} \quad \text{or} \quad P_{series} = \prod_{i=k}^{m} p_i \text{ ,} \tag{25}$$

The product is transferred to the entry of c_{13} and added to the existing value using parallel combination rules i.e.

$$c_{13_{new}} = c_{13_{old}} + c_{12}c_{23} - c_{13_{old}}c_{12}c_{23} \tag{26}$$

In fact, $c_{12_{new}}$ is the probability of the occurrence of two events that the element directly across nodes 1 and 3 is good as well as the two elements 1-2 and 2-3 in series.

In general, if node k has c_{ik} element incident to and c_{kj} incident from, then an entry $c_{ij} = c_{ik}c_{kj}$ is transferred to the location (i, j) and is added to the existing value using

$$c_{ij_{new}} = c_{ij_{old}} + c_{ik}c_{kj} - c_{ij_{old}}c_{ik}c_{kj} \qquad (27)$$

However, the entries c_{ik} and c_{kj} once they ~~have been used and the~~ node k has been eliminated are made zero.

The information about the node, needed for the elimination process, just described, can be had through the use of: what is called as degree matrix $\Delta \equiv [d_{ij}]$. There are two degree matrices defined for a di-graph, [D], one is out-degree matrix [Od(D)] which has only diagonal entries od_{ii}, indicating the number of branches 'going out' or directed away from the node i. The other matrix is in-degree matrix [Id(D)] for graph (D). This matrix also has diagonal entries id_{ii} indicating the number of branches 'coming in' or directed towards the node i. It is easier to understand that od_{ii} is the total number of non-zero entries of the row corresponding to node i, in [C].

Similarly, id_{ii} for node i will be the total number of non-zero entries corresponding to i^{th} column of matrix [C]. For example, just before elimination, [Od] and [Id] for network of Fig. 20 will be

$$[Od] \equiv \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{cccc} 3 & & & \\ & 1 & & \\ & & 1 & \\ & & & 0 \end{array}\right] \end{array} ; [Id] \equiv \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \left[\begin{array}{cccc} 0 & & & \\ & 1 & & \\ & & 2 & \\ & & & 2 \end{array}\right] \end{array} \qquad (28)$$

It will be interesting to note that id_{ii} will be 0 as node 1 happens to be a source-node. Similarly od_{44} will also be 0 as node 4 is a sink node having only incoming branches.

It may be remembered that elements of matrices [Od] and [Id] will keep on changing as the elimination proceeds. Finally when all intermediate nodes have been eliminated there will be only one entry in [Od] i.e.,for our example, $Od_{11} = 1$; rest of the entries will be zero. The same applies to [Id] which will also have only one entry i.e. $Id_{44} = 1$.

Since matrices [Od] and [Id] have only diagonal entries, it is economical to find a simpler way of storing them in memory. We can make use of the column corresponding to source-node of [C] for storing the diagonal elements of [Od] and the row corresponding to sink-node may be utilised for storing the diagonal elements of [Id] because both these column and row have zero entries throughout, always. Incidentally, the element of [C] corresponding to (sink, source) entry will always be zero, therefore overlapping of [Od] and [Id] elements at the corner have no problem because $od_{sink} = 0$ and $id_{source} = 0$. For example, Fig. 20 will have matrix [C] just before the elimination process as:

$$[C] \equiv \left[\begin{array}{c|ccc} 3 & 0.96 & 0.99 & 0.95 \\ 1 & 0 & 0.70 & 0 \\ 1 & 0 & 0 & 0.936 \\ \hline 0 & 1 & 2 & 2 \end{array}\right] \qquad (29)$$

As is evident from (29) we have been able to save lot of space by combining the features of three matrices [C], [Od] and [Id]. It is also easier to find total number of non-zero entries in any row and enter it in first column of that row and vice-versa.

It was pointed out earlier that the elimination starts with the node 'i' that has od_{ii} and id_{ii} equal to one. After eliminating and updating the entries of [C], again we look for the node which has in-degree and out-degree as one. This goes on till all such nodes have exhausted and finally the only entry in [C] left out will be that of $c_{source,\ sink}$ which will be the total transmittance or reliability of the network under consideration. The changes in [C], as nodes 2 and 3 are eliminated, are presented in (30) for the example under discussion.

$$\left[\begin{array}{c|ccc} 2 & 0 & 0.9967 & 0.95 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0.936 \\ \hline 0 & 0 & 1 & 2 \end{array}\right] \longrightarrow \left[\begin{array}{c|ccc} 1 & 0 & 0 & 0.9966 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad (30)$$

After node 2 is eliminated

After node 3 has been eliminated

The steps involved in the algorithm described can be summarised as follows:

1. Draw a di-graph for the network assigning proper direction to the elements and numbers to the nodes and elements.

2. From the data 'read in' a weighted-adjacency matrix is developed after combining the parallel elements across any two nodes.

3. Define od_{ii} and id_{ii} for each node.

4. Eliminate the node 'i' which has od_{ii} and id_{ii} as unity.

5. Transfer the product $c_{ik}c_{kj}$ to (i, j) entry and modify the old c_{ij} entry using,

$$c_{ij_{new}} = c_{ij_{old}} + c_{ik}c_{kj} - c_{ij_{old}}c_{ik}c_{kj}$$

Also make the entries c_{ik} and c_{kj} as zero.

6. Check whether all the intermediate nodes have been eliminated;

if not: go to step 3 otherwise print out the element $c_{source, sink}$ and stop.

This algorithm has a unique advantage of being fast and direct and requires minimum extra information or manipulation. Every information is containted in [C].

1.9.4. Non-series parallel networks

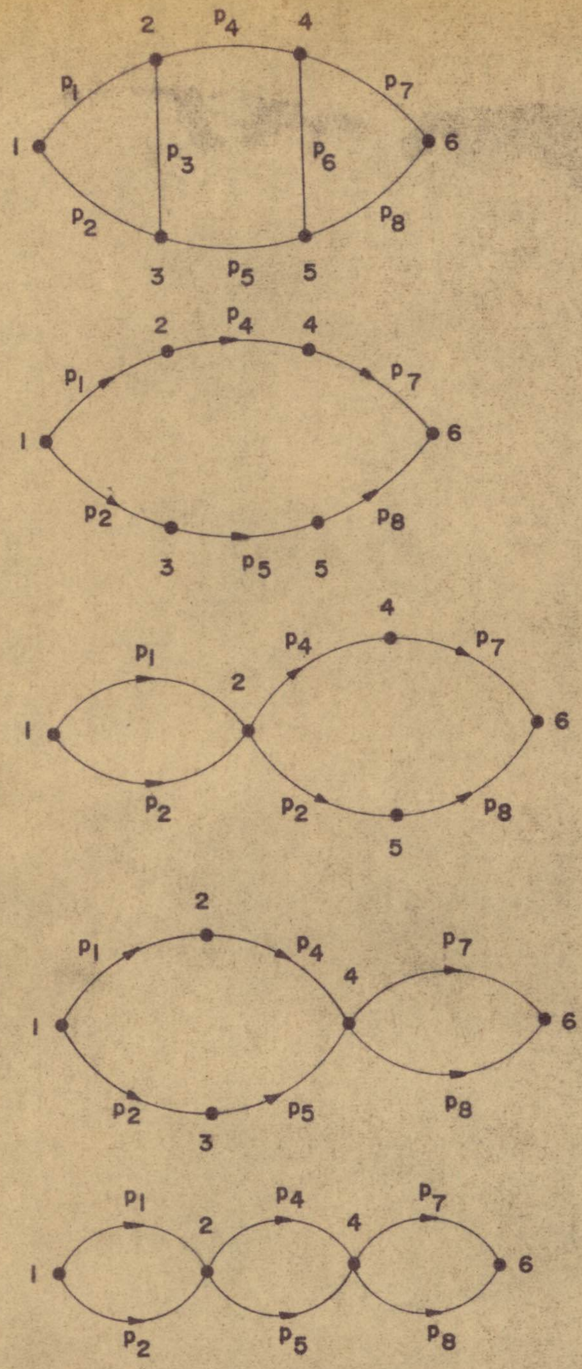
For non-series parallel networks, the same algorithm can be used effectively for the evaluation of reliability with somemanipulations. Reference [2] had suggested the use of Factoring Theorem. We will use the same theorem but in modified form and it becomes less cumbersome to use the theorem, than suggested in [2]. Actually the network as a whole, can be handled rather than breaking it into small units. The algorithm to be described will be found very convenient with the use of a computer and for large complex networks. The steps involved before applying the theorem can be enumerated as follows:

1. As in case of series parallel networks, elements across any two nodes can be combined first, as it is easier to work with reduced network. The weighted-adjacency matrix is developed.

2. Any series-node may be eliminated as discussed in earlier sections. This further reduces the network size. It may be pointed out that a network not decomposable finally to a single branch connecting source and sink nodes by algorithm described in section 1.9.3, is necessarily a non-series parallel network.

After the two steps mentioned above, we will be left with a small network (with interconnecting branches) which is quite convenient to handle.

Instead of usual procedure of factoring out one by one the



(a) NET WORK.

(b) BRANCHES 3 AND 6 OPEN.

(c) BRANCH 3 SHORTED AND 6 OPEN.

(d) BRANCH 3 OPEN AND 6 SHORTED.

(e) BRANCHES 3 AND 6 SHORTED.

FIG.21 A NON-SERIES PARALLEL NETWORK.

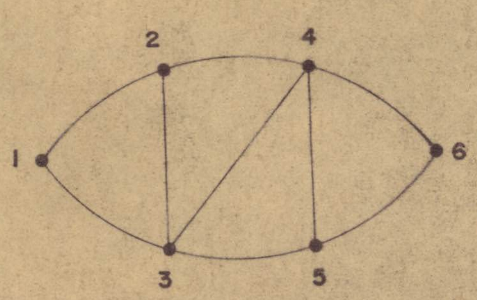


FIG.22 A NON-SERIES PARALLEL NETWORK WITH ADJACENT INTERCONNECTING BRANCHES.

interconnecting branches we will use different combinations of the states of interconnecting branches (much less in number usually, than the total number of elements) and define corresponding series-parallel networks to work with. Finally all the transmittances associated with such networks are added up algebraically to get the reliability of the network. The procedure involved will be illustrated with the help of an example, taking network of Fig. 21 (a)

Assuming, after going through the two steps mentioned in this section we end up with a network of 21 where branches with reliabilities p_3 and p_6 are the interconnecting branches. Now we consider all possible states of the branches 3 and 6, i.e. they may be shorted or opened. There are only four possibilities: branch 3 and 6 open, branch 3 shorted and 6 open, branch 3 open and 6 shorted and, finally, branches 3 and 6 may both be shorted. In general, if there are n interconnecting branches, then 2^n possibilities would be encountered. This should not be so disappointing as the interconnecting branches are usually very few. Secondly, it will be seen later that it is very convenient to work with the matrices associated with the graphs. Therefore, further manipulations on the matrices to simulate all the possible states of the interconnecting branches, are quite simple. Also any general approach for direct computation with minimum effort is preferable than otherwise.

It is easier to conceive from factoring theorem, that the total reliability R_{non} of network shown in Fig. 21 can be written as

$$R_{non} = q_3q_6 \text{ [reliability of network 21(b)]} + p_3q_6 \text{ [reliability of network 21(c)]} + q_3p_6 \text{ [reliability of network 21(d)]} +$$

$$p_3p_6\,[\text{reliability of network } 21(e)] \qquad (31)$$

Obviously, if one calculates the reliabilities of the networks 21(b), (c), (d) and e), R_{non} can be directly computed.

It is not necessary to rig up all the networks, and then computing the reliabilities separately using the method of section 1.9.3. Instead, we will make use of network 21(b) only and the other networks can be obtained by shorting one pair of terminals, and then two pairs of terminals at a time. This is simulated on the computer by first developing a matrix corresponding to 21(b) which can be obtained if the elements corresponding to interconnecting branches are removed from the weighted-adjacency matrix, viz.

$$[C]_{21(b)} \equiv \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & c_{12} & c_{13} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{24} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{35} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{46} \\ 0 & 0 & 0 & 0 & 0 & c_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad (32)$$

Thereafter the nodes 2 and 3 are shorted to get network of 21(c). The corresponding operation on (32) if the shorted nodes be recognised as single node 2, will be

1. Transfer all non-zero of 3rd column to corresponding positions in column 2.
2. Transfer all non-zero entries of 3rd row to corresponding positions in row 2.
3. All entries of row and column 3, are made to zero.

While the entries of m column are being transferred to column k, it must be remembered that if in i-row there is non-zero entry in k-column i.e. c_{ik} then the new c_{ik} after c_{im} is transferred to position c_{ik} will be given by

$$c_{ik_{new}} = c_{ik_{old}} + c_{im} - c_{ik_{old}} c_{im} \tag{33}$$

The same applies to the transfer of elements of 1-row to n-row. Following these rules the matrix $[C]_{21(c)}$ will have entries as

$$[C]_{21(c)} \equiv \begin{bmatrix} 0 & c_{12} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{24} & c_{25} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{46} \\ 0 & 0 & 0 & 0 & 0 & c_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{34}$$

Similarly, other networks can also be simulated using the above rules. All these series parallel networks are solved by the algorithm of section 1.9.3. At the end of one computation matrix [C] is initialised back to that corresponding to 21(b) to obtain a new network again. Once the reliability of a network is evaluated it is multiplied by proper combination of reliability or unreliability of the interconnecting branches according to (31) and stored in. This goes on till all networks have been considered. The final sum of all these will be the reliability of the non-series parallel network.

Network of Fig. 22, has adjacent interconnecting branches, in which case shorting any one interconnecting branch puts the other adjacent interconnecting branch in parallel with other elements. The total number of combinations if we remove interconnecting branches one by one will be less than what we will get following the procedure described. But extra labour involved cannot be compensated by the loss of generality of the algorithm. Moreover, the manipulations on the part of a user are also maintained as minimum.

1.9.5. Applications

The algorithm can be applied to any series parallel or non-series parallel redundant network. Nowhere loss of generality has occurred and thus this can be used in variety of cases.

The algorithm can also be applied to the networks (series parallel or non-series parallel) with elements having two types of failures viz. open circuit and short circuit, parallel to the method of section 8.2.

Applications of algorithm can be extended to evaluate the selective and non-selective operation probabilities [7] in case of any complicated relay networks. The procedure will be exactly similar to that of element with two types of failure.

Thus the algorithm can be effectively applied to solve variety of problems in the field of reliability evaluation which actually is essential in many reliability studies of a system.

1.10. A method of deriving reliability expression of redundant networks

The reliability expression of a redundant network series parallel or non-series parallel can be derived by first developing a digraph for the network, using the modelling described in section 1.9.1 and thereafter defining the associated boolean adjacency matrix [E]. Any entry E_{ij} indicates the state of the element lying between nodes i and j i.e. either good ($E_{ij} = 1$) or bad ($E_{ij} = 0$). It is obvious that probabilities associated with the boolean sum of events that all elements of all forward paths, are good, provide the reliability of the network. To generate all forward paths, one can multiply adjacency matrix n-2 times (n being the number of nodes). After each multiplication, the element of corresponding (source, sink) position is picked up and added (using boolean algebra rules) to the previous one. This method actually generates all forward paths of unit element length, two element length and so on. One can at the most have a forward path of maximum n-1 length if there are n nodes in the network. The number of multiplication can of course be reduced further to n-3. For example, for the network of 19(b), one requires only one full matrix multiplication, viz.

$$E_T = E_{14} + [0\ E_{12}E_{13}\ 0]\,[0\ E_{24}E_{34}\ 0]^T + [0\ E_{12}E_{13} 0] \cdot \begin{bmatrix} 0 & E_{12} & E_{13} & 0 \\ 0 & 0 & E_{23} & E_{24} \\ 0 & E_{32} & 0 & E_{14} \\ 0 & 0 & 0 & 0 \end{bmatrix} [0\ E_{24}E_{34}\ 0]^T \qquad (35)$$

The reliability of the network would then be given by $R = \Pr\left\{ E_T \right\}$. It may be stated here that $\Pr\left\{ E_{23} \cap E_{32} \right\} = 0$ as was indicated in section 2 for non-series parallel networks, therefore the terms involving these during the multiplications may be dropped right in the beginning. The method is particularly useful for non-series parallel redundant networks however complicated but with the condition that there exists only one branch between any two nodes. If there are more than one we replace them with an equivalent branch. It may also be remembered that + sign in (35) and the internal multiplication indicates the boolean sum of events.

CHAPTER 2

OPTIMISATION OF RELIABILITY WITH LINEAR CONSTRAINTS

In the previous chapter it has been amply emphasised that the reliability of a system can be increased by introducing redundancies in the sub-systems. Although one can obtain a high value of system reliability by providing as many redundancies as possible but to ensure that it is not a very costly, heavy or bulky system, the question of optimisation of system reliability with respect to cost, weight or volume etc. arises. The present chapter is, therefore, devoted to the problem of obtaining an optimal allocation of redundancy, i.e. maximum system reliability for the cost, weight, or volume etc. allowed.

2.1. Statement of the problem

Assuming there are k sub-systems or stages (all of them considered to be in series) in a system, stage i consists of n_i+1, similar units in parallel, each having independent probability q_i, $0 < q_i < 1$ of failure, the system reliability may be then given by

$$R(\bar{n}) = \prod_{i=1}^{k} (1 - q_i^{ni+1}) \qquad (2.1)$$

where $\bar{n}$ is a vector of non-negative integers such that $\bar{n} = (n_1, n_2, \ldots n_k)$ and represents the redundancies at each stage. There exist constraints on the allocation of redundancies which may be linear or non-linear. Assume linear constraints on $\bar{n}$ such that

$$\sum_{i=1}^{k} c_{ij}\, n_i \leqslant C_j, \quad j = 1, 2, \ldots r \qquad (2.2)$$

where $c_{ij} > 0$ and each C_j shows the allowable limit of cost, weight or volume etc. upto r constraints. The problem can therefore be stated as: the selection of vector $\bar{n}$ such that $R(\bar{n})$ is maximum subject to the constraints given in (2.2).

2.2. Domination

Assuming $C_j(\bar{n}) = \sum_{i=1}^{k} c_{ij}\, n_i$ represents the cost of the redundancy allocation $\bar{n}$, the allocation $\bar{n}^1$ is said to dominate $\bar{n}^2$ if $C_j(\bar{n}^1) \leqslant C_j(\bar{n}^2)$, $j = 1, 2, \ldots r$ while $R(\bar{n}^1) \geqslant R(\bar{n}^2)$. If in addition, at least one inequality is strict then $\bar{n}^1$ is said to dominate $\bar{n}^2$ strictly. A sequence S of redundancy allocation $\bar{n}^h$, $h = 1, 2, \ldots$ satisfying the constraints (2.2) is said to be a dominating sequence if no $\bar{n}^h$ is strictly dominated, and if every $\bar{n}$ satisfying the constraints (2.2), which is not strictly dominated, occurs in S.

Conversely, $\bar{n}^2$ is said to be undominated if $R(\bar{n}^1) > R(\bar{n}^2)$ implies $C_j(\bar{n}^1) > C_j(\bar{n}^2)$ for some j, whereas $R(\bar{n}^1) = R(\bar{n}^2)$ implies either $C_j(\bar{n}^1) > C_j(\bar{n}^2)$ for some j or $C_j(\bar{n}^1) = C_j(\bar{n}^2)$ for all j, where $C_j(\bar{n}^1) = \sum_{i=1}^{k} c_{ij}\, n_i$.

2.3. Approximate solution of redundancy allocation problem

An approximate solution to the problem (2.1) can be rapidly and easily obtained by generating an incomplete family of undominated allocations.

Let
$$R(\bar{n}) = \prod_{i=1}^{k} R_i(n_i) \tag{2.3}$$

where $R_i(n_i)$ is the reliability of sub-system using components of type i and that n_i redundant units of type i are provided.

Then
$$\log R(\bar{n}) = \sum_{i=1}^{k} \log R_i(n_i) \tag{2.4}$$

Since log x is a monotone-increasing functior of x, the problem of maximising $R(\bar{n})$ is equivalent to maximising log $R(\bar{n})$.

The procedure for generating an incomplete family of undominated allocation can be summarised as follows:

Starting with redundancy allocation of (0, 0 . . . 0), one adds a new component to that stage which yields greatest improvement in system reliability for the cost incurred in placing it. This continues till any one constraint is violated. The proof of the theorem that if log $R_i(n)$ is concave each redundancy allocation generated by above procedure is **undominated** is given in Appendix A. To prove that log $R(\bar{n})$ is a concave function of $\bar{n}$ one can show that

$$\triangle^2 \log R_i(n) = \triangle^2 \log(1-q_i^{n+1}) = \log \frac{(1-q_i^{n+3})(1-q_i^{n+1})}{(1-q_i^{n+2})^2} \qquad (2.5)$$

where $\triangle \log R_i(n) = \log R_i(n+1) - \log R_i(n)$.

The denominator is larger than numerator as

$$(1-q_i^{n+2})^2 - (1-q_i^{n+3})(1-q_i^{n+1}) = q_i^{n+1}\ (q_i-1)^2 > 0$$

Therefore $\triangle^2 \log R_i(n) < 0$, so also log $R(\bar{n})$ as the sum of concave functions is again a concave function.

Hence $\log R(\bar{n}) = \sum_{i=1}^{k} \log R_i\ (n_i)$ is concave.

2.3.1. Examples

(i) Single Cost Factor

Assuming that there is only one constraint in (2.1), i.e. cost of the item, the procedure for generating allocations will be to calculate desirability factor F_i for each stage given by

$$F_i = \frac{\triangle \log R_i(n_i)}{c_{i1}} = \frac{1}{c_{i1}}\left[\log R_i(n_i+1) - \log R_i(n_i)\right] \qquad (2.6)$$

Retaining the index i_o for which F_{io} is maximum amongst the stages, a component is added to that stage to find new allocation. If maximum occurs for more than one index, the lowest has been chosen for allocation.

Taking numerical example from reference [Kettelle 1962], in which data runs as,

Stage i	1	2	3	4
Reliability	0.8	0.7	0.75	0.85
Cost	1.2	2.3	3.4	4.5

the Table 2.1 gives the complete information about the undominated allocations. Fig. 2.1. shows the allocations on system reliability vs system cost. The allocations corresponding to a particular cost may be easily read from this figure. It may be noted here that allocations are given for the system and actual redundancy allocation can be found by subtracting (1, 1, 1, 1) from the system allocations. The computer program for this method is given in Appendix B.

(ii) Multiple Cost Factors

If there exist more than one 'cost' factors the desirability factors Fi's may be defined as

$$F_i = \frac{1}{\sum_{j=1}^{r} a_j c_{ij}} \left[\log R_i(n_i+1) - \log R_i(n_i) \right] \qquad (2.7)$$

$$i = 1, 2, \ldots k$$

where $a_1, a_2 \ldots a_r$ are non-negative weights with the condition that $\sum_{j=1}^{r} a_j = 1$. Here some of the a_j's may be zero but not all. In fact the vector $\bar{a}$ may be taken as (1, 0 . . . 0) to start with and successively a_j may be given a fixed increment $\triangle a_j$ till all possibilities of $\bar{a}$ may be exhausted and a final choice may be (0, 0,... 1).

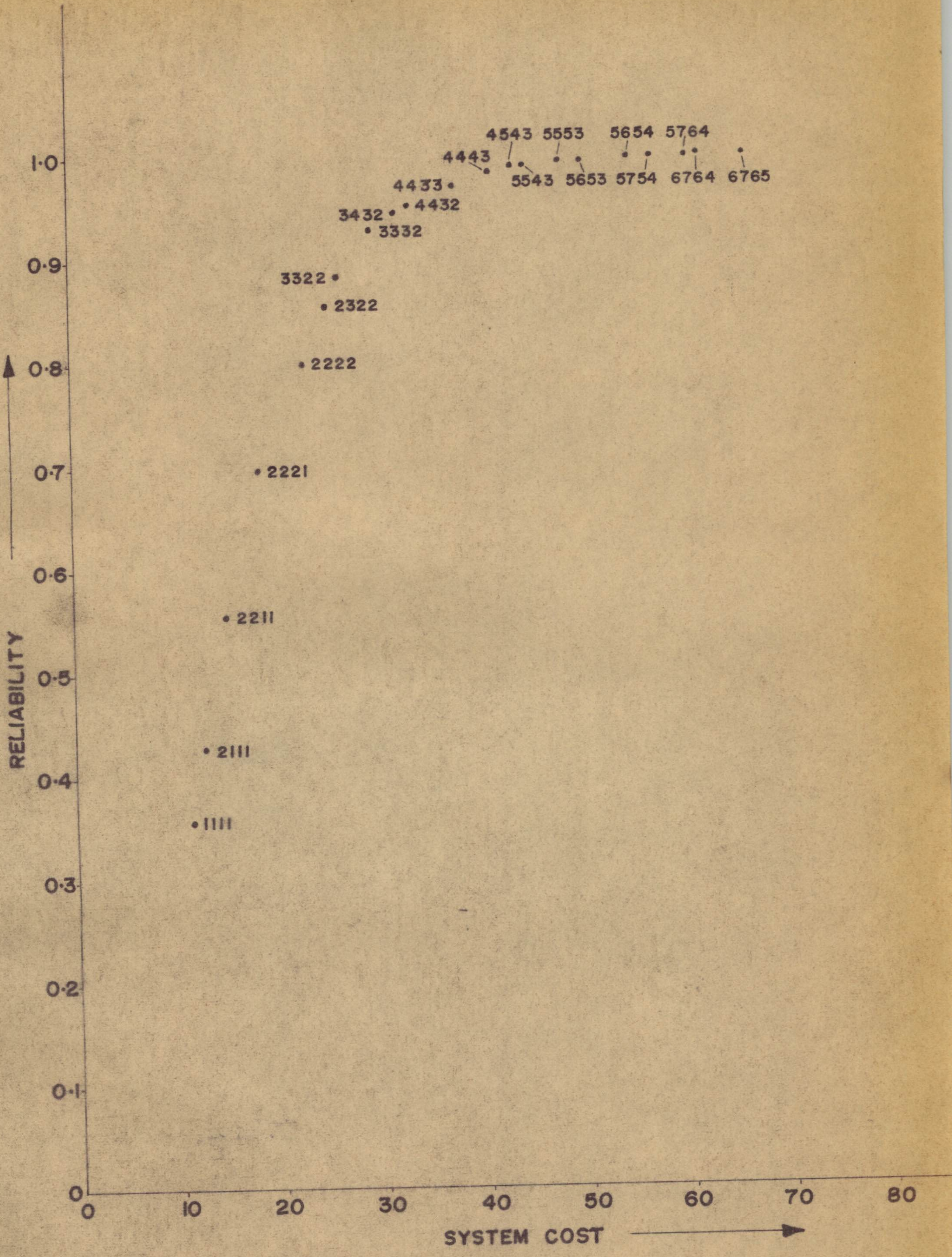


FIG.2·1 UNDOMINATED ALLOCATIONS.

Table 2.1 - Single Cost Allocation

System allocation	System Reliability	System cost	Desirability Factors			
			F_1	F_2	F_3	F_4
1 1 1 1	0.3570	11.4	0.15194	0.11407	0.06563	0.03106
2 1 1 1	0.4284	12.6	0.02732	0.11407	0.06563	0.03106
2 2 1 1	0.5569	14.9	0.02732	0.02910	0.06563	0.03106
2 2 2 1	0.6961	18.3	0.02732	0.02910	0.01435	0.03106
2 2 2 2	0.8005	22.8	0.02732	0.02910	0.01435	0.00431
2 3 2 2	0.8560	25.1	0.02732	0.00836	0.01435	0.00431
3 3 2 2	0.8845	26.3	0.00536	0.00836	0.01435	0.00431
3 3 3 2	0.9287	29.7	0.00536	0.00836	0.00348	0.00431
3 4 3 2	0.9468	32.0	0.00536	0.00248	0.00348	0.00431
4 4 3 2	0.9529	33.2	0.00107	0.00248	0.00348	0.00431
4 4 3 3	0.9715	37.7	0.00107	0.00248	0.00348	0.00064
4 4 4 3	0.9831	41.1	0.00107	0.00248	0.00086	0.00064
4 5 4 3	0.9887	43.4	0.00107	0.00074	0.00086	0.00064
5 5 4 3	0.9900	44.6	0.00021	0.00074	0.00086	0.00064
5 5 5 3	0.9929	48.0	0.00021	0.00074	0.00022	0.00064
5 6 5 3	0.9946	50.3	0.00021	0.00022	0.00022	0.00064
5 6 5 4	0.9974	54.8	0.00021	0.00022	0.00022	0.00010
5 7 5 4	0.9979	57.1	0.00021	0.00007	0.00022	0.00010
5 7 6 4	0.9987	60.5	0.00021	0.00007	0.00005	0.00010
6 7 6 4	0.9989	61.7	0.00004	0.00007	0.00005	0.00010
6 7 6 5	0.9994	66.2	-	-	-	-

The family of undominated allocations thus obtained is not complete even for all convex combinations of a_j's. However as the allocations are very close to each other, the true solution **to** the problem can be very closely found by proper selection of $\bar{a}$.

The method is based on the idea that an optimum balance has been struck in allocating among the different component types when increments in log reliability per unit **convex combination of costs** are the same for all component types within the limitations of discreteness of $(n_1, n_2 \dots n_k)$ variables.

Example:

The following example has been taken **for** illustration:

Stage i	1	2	3	4
Stage Reliability	0.80	0.70	0.75	0.85
Cost	1.2	2.3	3.4	4.5
Weight	5	4	8	7

Fig. 2.2 shows the system allocations on weight vs cost axes. The allocations for different combinations of a_j's have been listed in Table 2.2. These allocations corresponding to different a_j's have been clearly shown in Fig. 2.2. One can read off allocation to particular constraints on the weight and cost of the system from this figure. For example, if the system cost is not to exceed 56 and the weight should be less than 120 then system allocation may be given as (5, 6, 5, 4) with reliability of 0.99747 and actual cost and weight being 54.8 and 117.0, respectively. In Table 2.2 the last column gives the cases under which the allocations have been obtained. For brevity, the cases considered are listed below:

Table 2.2 - Multiple Cost Allocations

System Allocation	System cost	System Weight	System Reliability	Cases under which obtained
1 1 1 1	11.4	24.0	0.3570	1, 2, 3, 4, 5
1 2 1 1	13.7	28.0	0.4641	1, 2, 3, 4
2 1 1 1	12.6	29.0	0.4284	5
2 2 1 1	14.9	33.0	0.5569	1, 2, 3, 4, 5
2 2 2 1	18.3	41.0	0.6961	1, 2, 3, 4, 5
2 2 2 2	22.8	48.0	0.8005	1, 2, 3, 4, 5
2 3 2 2	25.1	52.0	0.8560	1, 2, 3, 4, 5
3 3 2 2	26.3	57.0	0.8845	1, 2, 3, 4, 5
3 4 3 2	32.0	69.0	0.9468	1, 2, 3, 4, 5
3 4 3 3	36.5	76.0	0.9653	1, 2, 3, 4
4 4 3 2	33.2	74.0	0.9529	5
3 4 4 3	39.9	84.0	0.9768	1, 2, 3
4 4 3 3	37.7	81.0	0.9715	4, 5
3 5 4 3	42.2	88.0	0.9824	1, 2
4 4 4 3	41.1	89.0	0.9831	3, 4, 5
4 5 4 3	43.4	93.0	0.9887	1, 2, 3, 4, 5
4 6 4 3	45.7	97.0	0.9904	1, 2, 3
4 5 5 3	46.8	101.0	0.9916	4
5 5 4 3	44.6	98.0	0.9900	5
4 6 4 4	50.2	104.0	0.9932	1, 2
4 6 5 3	49.1	105.0	0.9933	3, 4
5 5 5 3	48.0	106.0	0.9929	5
4 6 5 4	53.6	112.0	0.9961	1, 2, 3
5 6 5 3	50.3	110.0	0.9946	4, 5
5 6 5 4	54.8	117.0	0.9974	1, 2, 3, 4, 5
5 7 5 4	57.1	121.0	0.9979	1, 2, 3, 4, 5
5 7 6 4	60.5	129.0	0.9987	1, 2, 3, 4, 5
5 7 6 5	65.0	136.0	0.9991	1, 2
6 7 6 4	61.7	134.0	0.9989	3, 4, 5
6 7 6 5	66.2	141.0	0.9994	1, 2, 3, 4, 5
6 8 6 5	68.5	145.0	0.9995	1, 2, 3, 4, 5

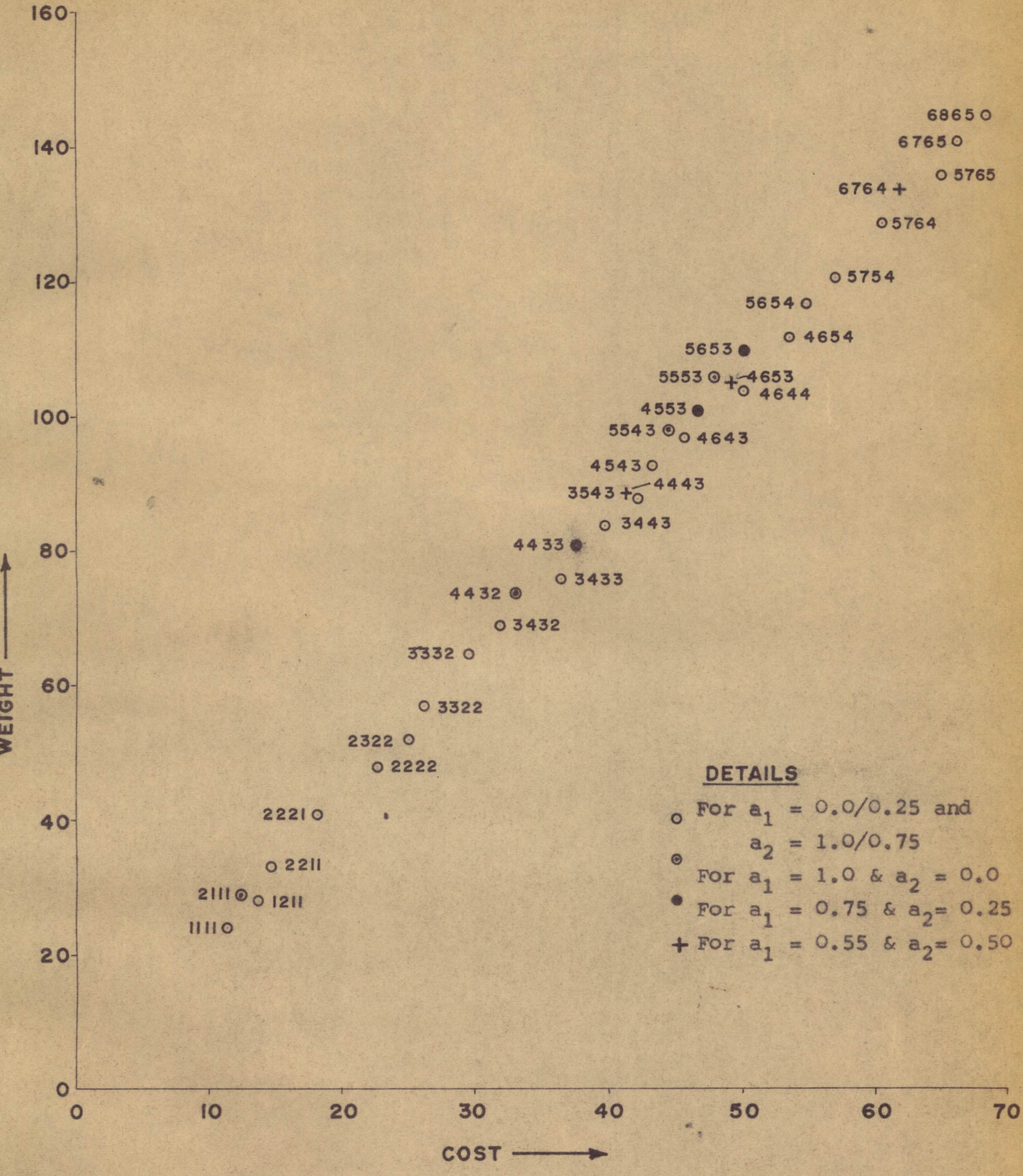


FIG. 2·2 MULTIPLE COST UNDOMINATED ALLOCATIONS.

Cases	Values of a_j	
	a_1	a_2
1	0.00	1.00
2	0.25	0.75
3	0.50	0.50
4	0.75	0.25
5	1.00	0.00

The computer program for the procedure outlined is given in Appendix C. One can easily solve for any allocation problem with given cost and weight constraints using this program. By using finer increment of Δa_j, all possible allocations can be obtained and the best with maximum reliability within the allowable limits on cost and weight can be selected.

2.3.2. Alternative Method

Another method which also generates an undominated allocation family for different values of vector $\bar{\lambda} = (\lambda_1, \lambda_2 \ . \ . \ . \ \lambda_r)$ for r 'cost' problem. By proper selection of $\bar{\lambda}$ one can arrive immediately at a larger allocation and in this method it is not necessary to generate the whole family of successively larger allocation as in the method of section (2.3.1). Therefore problem is to choose $\bar{\lambda} = (\lambda_1, \lambda_2 \ . \ . \ . \ \lambda_r)$ where each $\lambda_j \geqslant 0$ but not all $\lambda_j = 0$, for i = 1, 2 . . . k to calculate $n_i(\overline{\lambda})$ as the smallest integer m satisfying

$$\log R_i(m+1) - \log R_i(m) < \sum_{j=1}^{r} \lambda_j c_{ij} \qquad (2.8)$$

Here again if log $R(\bar{n})$ is concave, it can be proved on similar lines as the theorem given in Appendix A, the allocations will be undominated.

Inequality (2.8) can be further manipulated as follows:

$$\sum_{j=1}^{r} \lambda_j \, c_{ij} > \triangle \ \log R_i(n) = \log \frac{1 - q_i^{n+2}}{1 - q_i^{n+1}}$$

Exponentiating,

$$\exp\left[\sum_{j=1}^{r} \lambda_j \, c_{ij}\right] > \frac{1 - q_i^{n+2}}{1 - q_i^{n+1}} > \frac{\dfrac{1}{q_i^{n+1}} - q_i}{\dfrac{1}{q_i^{n+1}} - 1}$$

$$\text{or } \exp\left[\sum_{j=1}^{r} \lambda_j \, c_{ij}\right] > 1 + \frac{(1 - q_i)}{q_i^{-n-1} - 1}$$

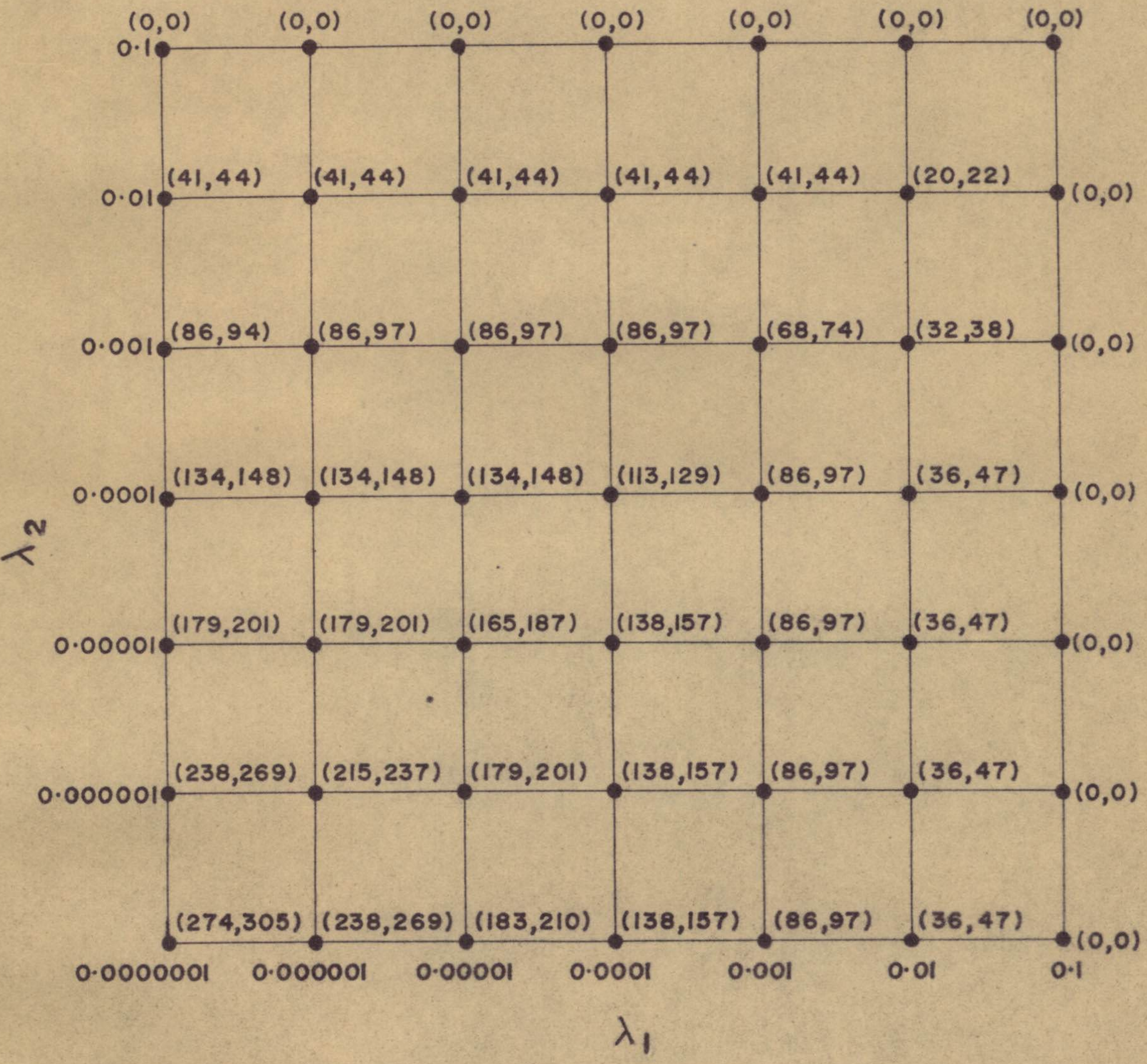
Writing $(1 - q_i) = p_i$ and further simplifying,

$$n > \frac{1}{\log \ q_i} \log \frac{\exp\left[\sum_{j=1}^{r} \lambda_j \, c_{ij}\right] - 1}{\exp\left[\sum_{j=1}^{r} \lambda_j \, c_{ij}\right] - q_i} - 1 \qquad (2.9)$$

Therefore $n_i(\overline{\lambda})$ can be written as (since n can only have integer values),

$$n_i(\overline{\lambda}) = \left[\frac{1}{\log \ q_i} \log \frac{\exp\left[\sum_{j=1}^{r} \lambda_j \, c_{ij}\right] - 1}{\exp\left[\sum_{j=1}^{r} \lambda_j \, c_{ij}\right] - q_i}\right] \qquad (2.10)$$

If the quantity within outermost brackets is denoted by x then the value of $n_i(\overline{\lambda})$ should be chosen as the largest integer not exceeding **x**. The procedure can therefore be outlined as finding out the $n_i(\overline{\lambda})$ for all stages, i.e. i = 1, 2 . . . k using a proper value of $\overline{\lambda}$ such that no cost constraint is violated. If in first choice of $\overline{\lambda}$, one does not arrive at the optimum value several trial values of $\overline{\lambda}$ may be used. In fact if one varies the vector $\overline{\lambda}$, different redundancy allocations may be obtained. The choice of $\overline{\lambda}$ is therefore



λ_2	0·0000001	0·000001	0·00001	0·0001	0·001	0·01	0·1
0·1	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)	(0,0)
0·01	(41,44)	(41,44)	(41,44)	(41,44)	(41,44)	(20,22)	(0,0)
0·001	(86,94)	(86,97)	(86,97)	(86,97)	(68,74)	(32,38)	(0,0)
0·0001	(134,148)	(134,148)	(134,148)	(113,129)	(86,97)	(36,47)	(0,0)
0·00001	(179,201)	(179,201)	(165,187)	(138,157)	(86,97)	(36,47)	(0,0)
0·000001	(238,269)	(215,237)	(179,201)	(138,157)	(86,97)	(36,47)	(0,0)
	(274,305)	(238,269)	(183,210)	(138,157)	(86,97)	(36,47)	(0,0)

FIG.2·3 COST-WEIGHT MAP FOR DIFFERENT λ_1 AND λ_2.

crucial in this method.

The author made several variations in the problem of a proper choice of the value of $\overline{\lambda}$ so as to get the correct allocation in a few trials. Several programs were written to study the different approaches. A few are reported herein.

(i) Discrete Steps Variation

The method has been illustrated by taking an example given below, in addition to the problem of section 2.3.1(ii).

Stage	1	2	3	4	5	Constraints
Reliability	0.90	0.75	0.65	0.80	0.85	
Cost	5	4	9	7	7	100
Weight	8	9	6	7	8	104

For the above problem, the allocations by varying λ_1 and λ_2 in discrete steps were found and the cost-weight map with λ_1 and λ_2 axes has been shown in Fig. 2.3. From the Fig. 2.3, it is obvious that if λ_1 is decreased cost of the system increases and if λ_2 is decreased weight increases. The conclusions thus derived can be summarised as follows:

1. Cost is a decreasing function of λ_1.
2. Weight is a decreasing function of λ_2.
3. By proper adjustments of λ_1 and λ_2 one can arrive at the desired allocation within the constraints assigned.

Therefore an algorithm was developed to satisfy the conditions listed below (Table 2.3), where C, W are the calculated cost and weight found from a particular allocation and CG and WG are given constraints on cost and weight, respectively.

Table 2.3.

C:CG	W:WG	Remarks
$<$	$<$	decreases both λ_1 and λ_2
$<$	=	stop
=	$<$	stop
=	=	stop
$>$	$<$	increase λ_1
$<$	$>$	increase λ_2
$>$	$>$	increase both λ_1 and λ_2
$>$	=	increase λ_1
=	$>$	increase λ_2

Using the above logic a computer program was written the flow chart for which is given in Fig. 2.4 and the results are listed below in Table 2.4.

Table 2.4.

Trial No.	λ_1	λ_2	Allocation	Cost	Weight
1	0.01	0.01	0 1 1 1 0	20	22
2	0.001	0.001	1 2 3 2 2	68	74
3	0.0001	0.0001	2 4 5 3 3	113	127
4	0.0005	0.0005	2 3 4 2 2	86	97
5	0.00005	0.00005	3 5 6 4 3	138	157
6	0.00025	0.00025	2 3 4 3 2	93	104

Here the decrease provided in λ_1 and λ_2 in a step was one-tenth and increase provided was 5 times to make the solution converge quickly.

Similarly, the four-stage problem of section 2.3.1(ii) with cost and weight constraints also was solved in seven trials as given below in Table 2.5.

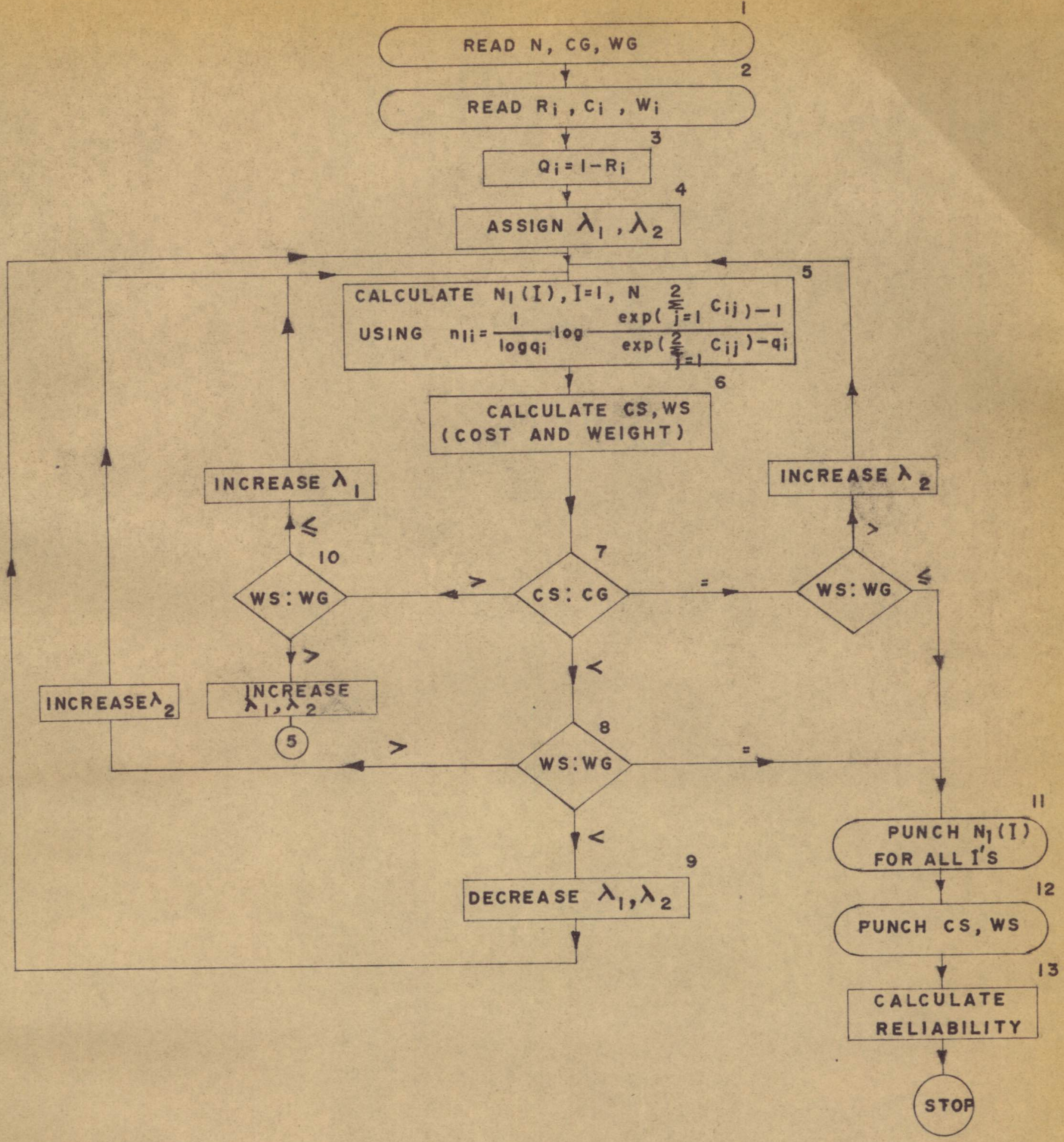


FIG.2·4 FLOW CHART FOR MULTIPLE COST ALLOCATIONS.

Table 2.5.

Trial No.	λ_1	λ_2	Allocation	Cost	Weight
1	0.01	0.01	1 2 1 1	13.7	28
2	0.001	0.001	3 3 3 2	29.7	65
3	0.0001	0.0001	4 5 4 3	43.4	93
4	0.00001	0.00001	5 7 6 4	60.5	129
5	0.00005	0.00005	4 6 5 3	49.1	105
6	0.000005	0.000005	6 8 6 5	68.5	145
7	0.000025	0.000025	5 6 5 4	54.8	117

In both the problems, however, it has been observed that either both cost and weight were less or more simultaneously than the allowable limits of cost and weight; therefore λ_1 and λ_2 were observed to be same at any trial.

One can, of course, stop after trail 3 in Table 2.4 and observe that the allocation lies somewhere between trial values of λ_1 and λ_2 used corresponding to trial 2 and 3. After step 2 the grid-meshes for λ_1 and λ_2 can be made of finer steps but it is usually difficult to ascertain as to what values of λ_1 and λ_2 exactly would lead to optimum allocation. To overcome this difficulty the author used random numbers generated to simulate the values of λ_1 and λ_2.

(ii) Random Numbers Approach

Best results would be obtained if the logic of Table 2.3 and the idea of generation of random numbers were combined to obtain exact allocation for the system. The author successfully used this approach for the four-stage problem of section 2.3.1(ii) results whereof are also reported in Table 2.5. Here again two variations were considered.

1. Random Numbers for both λ_1 and λ_2:

As is clear from Table 2.5 the values of λ_1 and λ_2 were somewhere between the values corresponding to step 3 and 4. To strike at the correct choice perhaps quickly, two random numbers for λ_1 and λ_2 were 'called' in the main program and a new allocation was found. The reference [24] gives many methods of generating random numbers between 0 and 1. The random number thus generated can be multiplied by a constant corresponding to the higher value of λ_1 and λ_2, respectively, in trial 3. It is startling to observe that the final allocation has been obtained in one trial only for the values of λ_1 and λ_2 as:

λ_1	λ_2	Allocation	Cost	Weight
0.00000614	0.00004131	5 6 5 4	54.8	117

2. Random Number for λ_1 only:

Instead of 'calling' random numbers from two random sequences, it may be easier to 'call' only one random number and for getting λ_2 one can multiply λ_1 by a preassigned constant or random integers in sequence. A typical observed case using this approach is given below:

m	λ_1	$m\lambda_1$	Allocation	Cost	Weight
0	0.00007424	0.0	5 6 5 4	54.8	117

In the opinion of the author the approach of both random numbers for λ_1 and λ_2 is quite promising and this can be very effectively used if the true optimum is to be searched and the allocations are very near to each other. In these cases this will be a useful approach and much nearer to true optimum.

(iii) A Graphical method

One can very conveniently prepare graphical chart to directly and quickly read the allocations in case of multiple or single cost constraints using equation (2.8), i.e.

$$\sum_{j=1}^{r} \lambda_j c_{ij} \rangle \Delta \log R_i(n) \rangle \log \frac{1-q_i^{n+2}}{1-q_i^{n+1}}$$

where n is the smallest integer satisfying above inequality. There in Fig. 2.4(a) the curves have been drawn for increasing values of n, i.e. the number of redundant units to be used at any stage corresponding to a particular value of unreliability q of a stage. To calculate the allocation at any stage. one has to simply look for the curve corresponding the unreliability of a unit of that stage and after choosing the values of (λ_1, $\lambda_2 .. \lambda_k$) and computing with the given values of c_{ij}'s, the left-hand side of the above expression one can read off the value of n for that stage which will provide $\Delta \log R_i(n)$ less than the computed value of $\sum_{j=1}^{r} \lambda_j c_{ij}$.

Once all the n_i , i=1,k have been calculated we can make a check about the constraints, if still there is scope of increasing the allocation then we decrease the values of $\bar{\lambda}$, so that we can arrive at a higher point on the curve. Moreover, the curves are almost linear upto a certain range; therefore extrapolation is also easier. It is also obvious from Fig. 2.4(a) that after a particular of n, the decrease in $\Delta \log R_i(n)$ is not at all pronounced and the curves seem to coincide and remain steady.

2.4. Kettelle's Algorithm

Kettelle [10] developed a simple computational procedure using dynamic programing algorithm for optimizing reliability without exceeding a constraint. Kettelle, however, presented it for a single-cost constraint only, i.e. cost of the equipment. The method actually develops a dominating sequence as the elements are successively added to the system at different stages. One can select the allocation within the total cost allowed which gives maximum reliability. The method requires either a rough estimate of the system realiability or the system reliability should be specified.

2.4.1. Dominating Sequences

The definition of dominating sequence has already been given earlier in section 2.2. In simpler language one can say that one configuration is said to dominate another if it has either (a) more reliability and no more cost, or (b) no less reliability and less cost. It is interesting to note that a dominating sequence contains only configurations that are undominated. One can generate whole family of undominated allocations starting with (0, 0 . . . 0) allocations in stages. The Kettelle's algorithm gives the complete family allocations which is not the case with the methods of earlier sections.

2.4.2. Illustrative example

Suppose it is required to have a system reliability of 0.99 with the data available about different stages as given below:

Stage	1	2	3	4
Reliability	0.80	0.70	0.75	0.85
Cost	1.2	2.3	3.4	4.5

The steps involved in the procedure can be outlined as follows:

1. Since all the stages can not be considered simultaneously for developing dominating sequence two stages at a time will have to be taken, therefore stages should be paired. In general for k stage system k-1 pairings can be done. In the illustrative problem there are two pairings of stages possible:

   1
   2
   3
   4

   i.e. first 1 & 2, then 3 & 4 and finally (1 & 2) and (3 & 4)

   1
   2
   3
   4

   i.e. first 1 & 2, then 2 & 3 and finally 3 & 4 are paired.

2. Minimum number of elements in each stage are found from the data available on system reliability. Assuming even if the reliability of each stage be equal to system reliability, one arrives at the minimum number of elements to start the algorithm. Otherwise the complete family starting with (0, 0) allocation will have to be generated. The minimum number of elements are calculated from the expression

$$n_i \text{ (min.)} = \frac{\text{Log } (1-R_g)}{\text{Log}(1-R_i)}$$

where R_g is the given system reliability and R_i is the reliability of element of i stage.

For the example of this section, the minimum elements calculated are:

Stage:	1	2	3	4
Minimum elements:	3	4	3	2

(1.2)

STAGE I

STAGE 2

ELEMENTS	3	4	5	6
COST	3.6	4.8	6.0	7.2
UNRELIABILITY	0.008	.0016	.00032	.000064
4 9.2 .0081	12.8 .0161	14.0 ✓ .0097	15.2 ✓ .00842	16.4 .008164
5 11.5 0.00243	15.1 .01043	16.3 ✓ .00403	17.5 ✓ .00275	18.7 .002494
(2.3) 6 13.8 0.000729	17.4 .008729	18.6 ✓ .002329	19.8 ✓ .001049	21.0 ✓ .000793

FIG.2·5 COMBINATION OF STAGE I AND 2 .

STAGE 3

STAGE 4

ELEMENTS	3	4	5	6
COST	10.2	13.6	17.0	20.4
UNRELIABILITY	.0156	.00391	.00098	.000245
2 9.0 .0225	19.2 .0381	22.6 .02641	26.0 .02348	29.4 .022745
3 13.5 .00338	23.7 .01898	27.1 .00729	30.5 .00436	33.9 .003625
4 18.0 .000506	28.2 .016106	31.6 .004416	35.0 .001486	38.4 .000751
5 22.5 .000076	32.7 .015676	36.1 .003986	39.5 .00105	42.9 .000321

FIG.2·6 COMBINATION OF STAGE 3 AND 4

3. A table as shown in Fig. 2.5 is developed where cost and unreliability are posted and starting with 3 and 4 elements for stage 1 and 2 the cost and unreliability of any other combination of elements for stage 1 and 2 greater than the minimum number of elements, are calculated successively. For calculating unreliability of the sub-system combined of stage 1 and 2 an approximation is usually made, that is: if Q_1 is the unreliability of first stage and Q_2 is the unreliability of stage 2 then the combined unreliability of stage 1 and 2 will be given by $Q_1 + Q_2$ leaving the third term of $(-Q_1Q_2)$. Kettelle has shown that the error introduced using this approximation is less than Q^2 where Q is system unreliability.

Another approximation that may reduce the length of dominating sequence is the following:
In comparing a pair of entries in the table developed one may introduce a tolerance factor ϵ_j for the j^{th} cost (here we have only one cost i.e. the cost of the equipment only) and/or a tolerance factor ϵ_q for unreliability. If two entries in the table differ by ϵ_j or less in the cost, they are considered alike as far as the cost is concerned; similarly, if they differ by ϵ_q or less in unreliability the result is that domination becomes more likely so that the lengths of the dominating sequences are reduced. If the dominating sequences are long one can introduce tolerance factors in cost and unreliability to reduce their lengths.

Another table for stage 2 and 3 combined is also prepared similarly and finally a table combining (1 & 2) and (2 & 3) stages is developed. They are shown in Fig. 2.6 and 2.7 respectively. The dominating sequences for all these three are given in Tables 2.6, 2.7 and 2.8

and Fig. 2.5, 2.6, 2.7.

Table 2.6 - Dominating Sequence for Stage 1 & 2

Dominating sequence	No. of elements per stage		Reliability	Unreliability	Cost
	Stage 1	Stage 2			
1	4	4	0.9903	0.0097	14.0
2	5	4	0.9916	0.0084	15.2
3	4	5	0.9960	0.0040	16.3
4	5	5	0.9973	0.0027	17.5
5	4	6	0.9977	0.0023	18.6
6	5	6	0.9980	0.0010	19.8
7	6	6	0.9992	0.0008	21.0

Table 2.7 - Dominating Sequence for Stage 3 & 4

Dominating sequence	No. of elements per stage		Reliability	Unreliability	Cost
	Stage 1	Stage 2			
1	4	3	0.9927	0.0073	27.1
2	5	3	0.9956	0.0044	30.5
3	6	3	0.9964	0.0036	33.9
4	5	4	0.9985	0.0015	35.0
5	6	4	0.9992	0.0008	38.4
6	6	5	0.9997	0.0003	42.9

Table 2.8 - Dominating Sequence for Stages (1 and 2) and (3 and 4) combined

Dominating sequence	No. of elements in a stage				Reliability	Unreliability	Cost
	1	2	3	4			
1	5	5	4	3	0.9900	0.0100	44.6
2	4	6	4	3	0.9904	0.0096	45.7
3	4	5	5	3	0.9916	0.0084	46.8
4	5	6	4	3	0.9917	0.0083	46.9
5	6	6	4	3	0.9919	0.0081	47.1
6	5	5	5	3	0.9929	0.0071	48.0
7	4	6	5	3	0.9933	0.0067	49.1
8	5	6	5	3	0.9945	0.0054	50.3
9	6	6	5	3	0.9948	0.0052	51.5
10	5	5	5	4	0.9958	0.0042	52.5
11	4	6	5	4	0.9962	0.0038	53.6
12	5	6	5	4	0.9975	0.0025	54.8
13	6	6	5	4	0.9977	0.0023	56.0
14	5	6	6	4	0.9982	0.0018	58.2
15	6	6	6	4	0.9982	0.0016	59.4
16	5	6	6	5	0.9987	0.0013	62.7
17	6	6	6	5	0.9990	0.0011	63.9

Obviously from Table 2.8, it is clear that the system with minimum cost should have allocation as (5, 5, 4, 3). On the other hand, if the constraint on cost is specified, one can find from the dominating sequence the allocation with cost less than or equal to the specified value.

2.4.3. Multiple Cost Constraints

In section 2.4.2, only single constraint was considered, however, the Kettelle's algorithm can be extended to multiple cost constraints such as cost, weight, volume etc. without much difficulty. Basically, the procedure remains same, except that an estimation of approximate reliability of the system is made to calculate the starting values for n_i for each stage.

The starting value in case of multiple costraint is found by adding one unit of each component type in succession until a constraint is violated upon the next addition. Then the reliability of the system is computed from the resulting value of $\bar{n}_o$ = $(n_1, n_2 \, . \, . \, n_k)$. Finally from the calculated value of system reliability minimum component types in each stage are calculated as in Kettelle's algorithm using formula

$$n_i = \frac{\log(1-R_s)}{\log(1-R_i)}$$

A proper solution of $\bar{n}_o$ reduces considerably the calculation in preparing the table.

2.4.4. Multiple Cost Problem

For illustration the following problem has been worked out in detail in Tables 2.10 and 2.11. The final dominating sequences for combined stages is shown in Table 2.12.

Example: To find the optimum allocation for the system given below with cost and weight not exceeding 56 and 120.

Stage	1	2	3	4
Cost	1.2	2.3	3.4	4.5
Weight	5	4	8	7

Finding the starting values of n_i table 2.9 is prepared, which

Table 2.9 -

Stage 1	Stage 2	Stage 3	Stage 4	Cost	Weight	Remarks
1	1	1	1	11.4	24	
2	1	1	1	12.6	29	
2	2	1	1	14.9	33	
2	2	2	1	18.3	41	
2	2	2	2	22.8	48	
3	2	2	2	24.0	53	
3	3	2	2	26.3	57	Cost constraint 56
3	3	3	2	29.7	65	
3	3	3	3	34.2	72	Weight constraint 120
4	3	3	3	35.4	77	
4	4	3	3	37.7	81	
4	4	4	3	41.1	89	
4	4	4	4	45.6	96	
5	4	4	4	46.8	101	
5	5	4	4	49.1	105	Attainable reliability = 0.99577
5	5	5	4	52.5	113	
5	5	5	5	57.0	120	

			STAGE I				
			4	5	6	7	8
		Cost	4.8	6.0	7.2	8.4	9.6
		Weight	20.0	25.0	30.0	35.0	40.0
		Unreliability	0.0016	0.00032	0.000064	0.0000128	0.00000256
		6.9	11.7	12.9	14.1	15.3	
	3	12.0	32.0	37.0	42.0		
		0.0081	0.0097	0.00842	0.008164	0.0081128	0.00810256
		9.2	14.0	15.2	16.4	17.6	
	4	16.0	36.0	41.0	46.0	51.0	
		0.00243	0.00403	0.00275	0.002494	0.0024428	0.00243256
		11.5	16.3	17.5	18.7	19.9	
STAGE II	5	20.0	40.0	45.0	50.0	55.0	
		0.000729	0.002329	0.001049	0.000793	0.0007418	0.00073156
		13.8	18.6	19.8	21.0	22.2	23.4
	6	24.0	44.0	49.0	54.0	59.0	
		0.0002187	0.0018187	0.0005387	0.0002827	0.0002315	0.00022126
		16.1	20.9	22.1	23.3	24.5	25.7
	7	28.0	48.0		58.0	63.0	68.0
		0.00006561	0.00166561	0.00038561	0.00012961	0.00007841	0.00006817
		18.4				26.8	28.0
	8	32.0				67.0	72.0
		0.000021183	0.001621183	0.000341183	0.000085183	0.000033983	0.000023743

Table 2.10 - Dominating Sequence for Stages 1 & 2

			STAGE III				
			4	5	6	7	8
		Cost Weight Unreliability	13.6 32.0 0.003906	17.0 40.0 0.000976	20.4 48.0 0.000244	23.8 56.0 0.000061	27.2 64.0 0.000015
	3	13.5 21.0 0.003375	27.1 53.0 0.007281	30.5 61.0 0.004351	33.9 69.0 0.003619		
	4	18.0 28.0 0.000506	31.6 60.0 0.004412	35.0 68.0 0.001482	38.4 76.0 0.000750	.8 0.000567	
STAGE IV	5	22.5 35.0 0.000075	37.1 67.0 0.003981	39.5 75.0 0.001051	42.9 83.0 0.000319	46.3 91.0 0.000136	49.7 99.0 0.000090
	6	27.0 42.0 0.000011	40.6 74.0 0.003917	0.000987	0.000255	50.8 98.0 0.000071	54.2 106.0 0.000026
	7	31.5 49.0 0.000001	×	×	×	×	58.7 113.0 0.000016
	8	36.0 56.0 0.000000	×	×	×	×	63.2 120.0 0.000015

Table 2.11 - Dominating Sequence for Stages 3 & 4

Table 2.12 -

Dominating Sequence for Stages 1, 2, 3 & 4 combined

Dominating sequence	No. of equipments Stage				Unreliability	Reliability	Cost	Weight
	I	II	III	IV				
1	4	5	4	3	.009610	.990390	43.4	93
2	4	4	5	3	.008381	.991619	44.5	97
3	5	5	4	3	.008330	.991670	44.6	98
4	5	4	5	3	.007101	.992899	45.7	102
5	4	5	5	3	.006770	.993230	46.8	101
6	5	5	5	3	.005400	.994600	48.0	106
7	4	4	5	4	.005512	.994488	49.0	104
8	6	5	5	3	.005144	.994856	49.2	111
9	5	4	5	4	.004232	.995768	50.2	109
10	4	5	5	4	.003811	.996189	51.3	108
11	5	5	5	4	.002531	.997469	52.5	113
12	6	5	5	4	.002275	.997725	53.7	118
13	5	6	5	4	.002020	.997980	54.8	117
14	5	5	6	4	.001799	.998201	55.9	121
15	6	6	5	4	.001764	.998236	56.0	122
16	6	5	6	4	.001543	.998457	57.1	126
17	5	6	6	4	.001288	.998712	58.2	125
18	6	6	6	4	.001032	.998968	59.2	130
19	7	6	6	4	.000981	.999019	60.6	135
20	6	7	6	4	.000879	.999121	61.7	134
21	5	6	6	5	.000857	.999143	62.7	132
22	7	7	6	4	.000828	.999172	62.9	139
23	6	6	6	5	.000601	.999399	63.9	137
24	7	6	6	5	.000550	.999450	65.1	142
25	6	7	6	5	.000448	.999552	66.2	141

gives attainable reliability. Then from the attainable system reliability the starting allocation will be (4, 3, 4, 3). Proceeding as indicated in earlier section 2.4.2, the allocation within the allowable limits on cost and weight, is found to be (5, 6, 5, 4).

2.5. Bellman Dynamic Programming Approach

Bellman's [16, 17] dynamic programming, can be applied conveniently to the problem of optimising reliability of a system with k stages in series subject to one or two constraints such as cost or weight or both. The allocation problem is solved as a multistage decision problem where at any stage j, the decision is made on how much to allocate to activity j that is x_j is selected. The dynamic programming approach to solving problem makes use of this fact and really solves a sequence of problems beginning with a one-stage problem, moving on to a two-stage one etc., until finally all stages are included. The solution for k stages is obtained from the solution for k-1 stages by adding the k^{th} stage and making use of k-1 stages. The optimal allocation x_j, $j = 1 \ldots k$ depends on the total quantity of resource ξ, which is available for allocation to k stages. The mathematical formulation of redundancy allocation is given in the following section.

2.5.1. Dynamic Programming Formulation

The non-linear programming problem to be solved can be states as follows:

$$\sum_{j=1}^{k} a_j x_j \leqslant b; \; a_j > 0, \;\; j = 1, \ldots, k, \; x_j \geqslant 0, \; j=1, \ldots, k;$$

$$\text{all } x_j \text{ integers, } \max z = \sum_{j=1}^{k} \phi_j(x_j) \qquad (2.11)$$

The above problem involves only one constraint b, and has a separable objective function, requiring all x_j's to be integers. The problem with two constraints can also be solved easily by the use of Lagrange's multiplier and introducing one of them in objective function as will be discussed later. The computational problem can be described as follows:

If the sequence of functions be defined as

$$f_n(\xi) = \max_{x_1,\dots x_n} \sum_{j=1}^{n} \phi_j(x_j);\ n = 1,\dots k \tag{2.12}$$

where maximisation is carried out for non-negative integers satisfying

$$\sum_{j=1}^{n} a_j x_j \leqslant \xi \tag{2.13}$$

Once $f_1(\xi)$ has been calculated directly, the remaining $f_n(\xi)$ can be computed recursively, since

$$f_n(\xi) = \max_{x_n} \left[\phi_n(x_n) + \max_{x_1,\dots x_{n-1}} \sum_{j=1}^{n-1} \phi_j(x_j) \right] \tag{2.14}$$

where in computing

$$\max_{x_1,\dots x_{n-1}} \sum_{j=1}^{n-1} \phi_j(x_j) \tag{2.15}$$

the maximisation is carried over non-negative integers $x_1,\dots,$ x_{n-1} satisfying

$$\sum_{j=1}^{n-1} a_j x_j \leqslant \xi - a_n x_n$$

Under these conditons, (2.15) is simply $f_{n-1}(\xi - a_n x_n)$. Therefore,

$$f_n(\xi) = \max_{x_n} \left[\phi_n(x_n) + f_{n-1}(\xi - a_n x_n) \right], \; n=2, \ldots, k \qquad (2.16)$$

and x_n varie over the values 0, 1, ..., $\left[\xi / a_n \right]$ (2.16)

Finally

$$z^* = f_k(b) \qquad (2.17)$$

Summarising the procedure, one can start with first stage by computing

$$f_1(\xi) = \max_{0 \leqslant x_1 \leqslant \left[\xi / a_1 \right]} \phi(x_1) \qquad (2.18)$$

where in computing $f_1(\xi)$ for a given ξ, x_1 ranges over integers in the interval $\left[\xi / a_1 \right]$. For each value of $\xi = 0, 1, \ldots b$ $f_1(\xi)$ is calculated. Denoting the value of x_1 by $\hat{x}_1(\xi)$ for which

$$f_1(\xi) = \phi_1 \left[\hat{x}_1(\xi) \right] \qquad (2.19)$$

that is, $\hat{x}_1(\xi)$ is a value of x_1 which maximises $f_1(x_1)$ when x_1 takes the values 0, 1, ..., $\left[\xi / a_1 \right]$, a table such as given below is built:

Table 2.13

ξ	$f_1(\xi)$	$\hat{x}_1(\xi)$
0	$f_1(0)$	$\hat{x}_1(0)$
1	$f_1(1)$	$\hat{x}_1(1)$
$\vdots$	$\vdots$	$\vdots$
b	$f_1(b)$	$\hat{x}_1(b)$

Once $f_1(\xi)$ have been calculated one can proceed to compute $f_2(\xi)$ for every value of $\xi = 0, 1, \ldots, b$ using

$$f_2(\xi) = \max_{0 \leqslant x_2 \leqslant [\xi/a_2]} \left[\phi_2(x_2)+f_1(\xi - a_2x_2)\right] \tag{2.20}$$

For a given ξ, $f_2(\xi)$ is computed as follows:

$$\psi_2(0;\xi) = \phi_2(0)+f_1(\xi)$$

$$\psi_2(1;\xi) = \phi_2(1)+f_1(\xi - a_2) \tag{2.21}$$

$$\psi_2([\xi/a_2];\xi) = \phi_2([\xi/a_2]+f_1(\xi - a_2[\xi/a_2])$$

The maximum of ψ_2's is stored as $f_2(\xi)$ and the corresponding value of x_2 as $\hat{x}_2(\xi)$. Again a table similar to 2.13 is prepared.

Similalry, the procedure is adopted to calculate $f_3(\xi)$ for $\xi = 0, 1, \ldots, b$ and finally for all $f_k(\xi)$.

To determine the optimum allocation at each stage, one can start with k^{th} stage where $f_k(b) = z^*$ and $\hat{x}_k(b)$ is the allocation at k^{th} stage x_k^*. With allocation at k^{th} stage known, allocation at k-1 stage will be given by

$$x_{k-1}^* = \hat{x}_{k-1}(b-a_kx_k^*) \tag{2.22}$$

This proceeds on backwards till, finally

$$x_1^* = \hat{x}_1(b-\sum_{n=2}^{k} a_nx_n^*) \tag{2.23}$$

2.5.2. Optimisation with two constraints and one control variable

If the optimisation problem is framed as

$$\sum_{j=1}^{k} a_{1j}x_j \leqslant b_1; \ \sum_{j=1}^{k} a_{2j}x_j \leqslant b_2 \ , \ x_j \geqslant 0, \ \ j=1, \ldots k; \text{ all integers,}$$

$$\max z = \sum_{j=1}^{k} \phi_j(x_j) \tag{2.24}$$

where all a_{ij}; b_i are assumed to be positive integers, obviously the sequence of functions defined analogous to previous section will be

$$f_n(\xi_1, \xi_2) = \max_{x_1 \dots x_n} \sum_{j=1}^{n} \phi_j(x_j), \; j=1, \dots k \qquad (2.25)$$

where maximisation is to be carried out over non-negative integers satisfying

$$\sum_{j=1}^{n} a_{1j}x_j \leq \xi_1 , \quad \sum_{j=1}^{n} a_{2j}x_j \leq \xi_2 \qquad (2.26)$$

If ξ_1 and ξ_2 are two state parameters, the state functions for first stage will be

$$f_1(\xi_1 , \xi_2) = \max_{0 \leq x_1 \leq \delta_1} \phi_1(x_1) \qquad (2.27)$$

or in general,

$$f_n(\xi_1 , \xi_2) = \max_{0 \leq x_1 \leq \delta_n} \left[\phi_n(x_n)+f_{n-1}(\xi_1-a_{1n}x_n , \xi_2-a_{2n}x_n) \right]$$

$$n = 2, \dots k \qquad (2.28)$$

and

$$z^* = f_k(b_1 , b_2) \qquad (2.29)$$

where

$$\delta_n = \min \left\{ \left[\xi_1/a_n\right], \left[\xi_2/a_{2n}\right] \right\} \qquad (2.30)$$

Once $f_n(\xi_1 , \xi_2)$ is determined, simultaneously $\hat{x}_n(\xi_1 , \xi_2)$ is stored. At k^{th} stage, x_k^* corresponding to $f_k(b_1 , b_2)$ is determined and remaining optimum allocation at each stage is found by tracing back the stored table of $f_n(\xi_1 , \xi_2)$ and $\hat{x}_n(\xi_1 , \xi_2)$ corresponding to two state parameters ξ_1 and ξ_2 instead of only one as given in (2.22) and (2.23).

It is much more difficult to solve this problem than (2.11) because f_n and $\hat{x}_n$ are now functions of two arguments. If both ξ_1 and ξ_2 can take 100 values, then in general one may have to tabulate $f_n(\xi_1, \xi_2)$ for 10000 possible combinations of ξ_1 and ξ_2. Moreover, maximisations have to be carried out 10000 times at each stage. Another trouble that may arise is that of storing such a large table in costly computer memory. Also the speed has to be high to reduce access-time. To overcome this difficulty one may use Lagrange's multiplier technique as will be discussed in the following section.

2.5.3. Lagrange's Multiplier technique

A Lagrange multipler λ can be used to reduce the number of state parameters by one.

Problem of (2.24) can be reframed as

$$\sum_{j=1}^{k} a_{1j}x_j \leqslant b_1$$

$$x_j \geqslant 0 \text{ , } j=1, \ldots k$$

$$\max z_1 = \sum_{j=1}^{k} \phi_j(x_j) - \lambda \sum_{j=1}^{k} a_{2j}x_j \qquad (2.31)$$

This problem can be easily solved as single constraint problem involving only one state parameter.

The recurrence relations for the state function will be

$$f_n(\xi) = \max_{x_n}\left[\phi_n(x_n) - \lambda a_{2n}x_n + f_{n-1}(\xi - a_{1n}x_n)\right] \quad n=2, \ldots k \qquad (2.32)$$

For the problem of (2.31) an obvious assumption is made that x_j are continuous variables and $\phi_j(x_j)$ are nondecreasing functions of x_j. It is therefore clear that one of the constraints holds a strict equality for any optimum solution and in fact in (2.31) second constraint i.e.

$$\sum_{j=1}^{k} a_{2j}x_j = b_2$$

is assumed to hold strict equality. This however does not present any difficulty in keeping x_j as integers. In this case it is not necessarily true that either constraint must hold as a strict equality. One can proceed by varying λ to make $z\left[x(\lambda)\right]$ as large as possible while not violating either of the constraints. If the eliminated constraint holds as strict equality when the x_j are not restricted to be integers, this is equivalent to that of determining λ such that the constraint comes closer to strict equality without violating the constraints.

The procedure of computation will be exactly the same as that of section (2.5.1). Often one has to use his own judgment in making a suitable choice of λ. Few trials or in fact the solution of problem (2.31) is usually required till one strikes at the correct value of λ to get the optimum solution. If two trials have been made for two different values of λ then one can usually make linear intrapolation or extrapolation to arrive at almost correct choice of new λ.

If λ_0 and λ_1 are the two values of Lagrange multipliers tried and corresponding values of $\sum_{j=1}^{k} a_{2j}x_j$ are b_2^0 and b_2^1 then a new value of multiplier λ_2 can be used, given by the relation

$$\lambda_2 = \frac{\lambda_1 - \lambda_0}{b_2^1 - b_2^0}(b_2 - b_2^0) + \lambda_0 \qquad (2.33)$$

If more than two values of λ have been tried, the latest two can be used for intrapolation or extrapolation.

2.5.4. Optimum Redundancy allocation subject to two Linear Constraints

If $1+x_j$ components of reliability p_j are used at the j^{th} stage then the probability of successful operation $R_j(x_j)$ of the j^{th} stage is given by

$$R_j(x_j) = 1-(1-p_j)^{1+x_j} \tag{2.34}$$

and the overall reliability of the system may be written as

$$R_s = \prod_{j=1}^{k} R_j(x_j) \equiv \prod_{j=1}^{k} \left\{1-(1-p_j)^{1+x_j}\right\} \tag{2.35}$$

Expression (2.35) can be expressed as

$$Z \equiv \log R_s = \sum_{j=1}^{k} \phi_j(x_j) \tag{2.36}$$

where $\phi_j(x_j) \equiv \log R_j(x_j) = \log \left\{1-(1-p_j)^{1+x_j}\right\}$

This form is more convenient to use since each term of the sum depends only on a single variable. Moreover $\phi_j(x_j)$ is monotone-increasing concave function of x_j, maximising R_s is equivalent to maximising $\log R_s$.

The problem is therefore as follows:

$$\text{maximise } z = \sum_{j=1}^{k} \phi_j(x_j)$$

$$x_j \geqslant 0 \text{ , } j = 1 \text{ , } \ldots k, \text{ all } x_j \text{ integers}$$

subject to the cost and weight constraints

$$\sum_{j=1}^{k} c_j x_j \leqslant C \quad \text{and} \quad \sum_{j=1}^{k} w_j x_j \leqslant W \tag{2.37}$$

where c_j and w_j are cost and weight of the like-components at j^{th} stage, C and W are given cost and weight constraints; or, alternatively, introducing Lagrange's multiplier λ,

$$\max z_1 = \sum_{j=1}^{k} \phi_j(x_j) - \lambda \sum_{j=1}^{k} w_j x_j$$

$$x_j \geqslant 0 , \quad j=1,\ldots k$$

subject to the cost constraint

$$\sum_{j=1}^{k} c_j x_j \leqslant C \qquad (2.38)$$

The recurrence formula will be given by

$$f_n(\xi) = \max_{x_n} \left[\phi_n(x_n) - \lambda w_n x_n + f_{n-1}(\xi - c_n x_n)\right]$$

$$n = 2,\ldots k \qquad (2.39)$$

The author has used this form for the following reasons rather than using product formulation of reliability problem [16, 17, 18]:

1. Addition is faster on computers than multiplication and thereby reducing considerably the time for each run of the problem for a particular value of λ.
2. As is clear from the recurrence expression of (2.39) the terms corresponding to $\phi_j(x_j)$ and $\lambda w_j x_j$ appear in sum form, the values of $\phi_j(x_j)$ can be calculated once for all and may be subsequently used for all possible values of x_j at each stage and different values of λ. This saves re-computation of the product each time the value of x_j is changed from 0 to $[\xi/x_j]$ in each stage for all values of ξ from 0 to b_1.

This process reduces considerably the time due to the fact that a large number of multiplications are saved. This procedure infact reduced the total time of a run to almost 1/4. With high speed memory this may be even less.

3. This method also helps in estimating a correct value of λ quickly as the effect of the variation of $\lambda w_j x_j$ can be clearly observed in process of calculation.

2.5.5. Example

Using the computer program given in Appendix D, the author tried the above procedure and also after modifying the program to the product form, the following problem of table 2.14 has been solved, for justifying the time comparison as discussed in earlier section. The results are presented in tables 2.15 and 2.16.

Table 2.14

Stage	1	2	3	4	5
Reliability	0.90	0.75	0.65	0.80	0.85
Cost	5	4	9	7	7
Weight	8	9	6	7	8
Cost constraint	100 units				
Weight constraint	104 units				

The system weight corresponding to different values of λ is given in table 2.16.

Table 2.16

λ	0.002	0.0014	0.0012	0.001	0.0008
Allocation	1, 2, 3, 2, 2	1, 2, 4, 2, 2	1, 3, 4, 2, 2	2, 3, 4, 2, 2	2, 3, 4, 3, 2
System Weight	74	80	89	97	104
System Cost	68	77	81	86	93

ξ	Stage No.	$\lambda = 0.002$		$\lambda = 0.0014$		$\lambda = 0.0012$		$\lambda = 0.001$		$\lambda = 0.0008$	
		$f(\xi)$	$\hat{x}(\xi)$	$f(\xi)$	$\hat{x}(\xi)$	$f(\xi)$	$\hat{x}(\xi)$	$f(\xi)$	$\hat{x}(\xi)$	$f(\xi)$	$\hat{x}(\xi)$
0	1	-0.10536051	0	-0.10536051	0	-0.10536051	0	-0.10536051	0	-0.10536051	0
1		"	0	"	0	"	0	"	0	"	0
⋮											
5		-0.02605033	1	-0.02125033	1	-0.01965033	1	-0.01805033	1	-0.01645033	1
⋮											
10		"	1	"	1	"	1	-0.01700050	2	-0.01380050	2
⋮											
100		"	1	"	1	"	1	"	2	"	2
0	2	-0.28768207	0	-0.28768207	0	-0.28768207	0	-0.28768207	0	-0.28768207	0
⋮											
4		-0.10858885	1	-0.09838885	1	-0.09498885	1	-0.09053902	1	-0.08553902	1
⋮											
8		-0.07779869	2	-0.06219869	2	-0.05677869	2	-0.05074885	2	-0.04394885	2
12		"	2	"	2	-0.05596418	2	-0.04791434	3	-0.03931434	3
100		"	2	"	2	"	3	"	3	"	3
0	3	-0.43078291	0	-0.43078291	0	-0.43078291	0	-0.43078291	0	-0.43078291	0
10		-0.22047701	1	-0.20127701	1	-0.19384250	1	-0.18459266	1	-0.17479266	1
19		-0.14561997	2	-0.12281997	2	-0.11418546	2	-0.10373562	2	-0.09273562	2
28		-0.12891862	3	-0.10251862	3	-0.09268411	3	-0.0810342	3	-0.06883428	3
37		"	3	-0.10106463	4	-0.09003012	4	-0.07718028	4	-0.06378028	4
100		"	3	"	4	"	4	"	4	"	4
0	4	-0.22314355	0	-0.22314355	0	-0.22314355	0	-0.22314355	0	-0.22314355	0
8		-0.18374061	1	-0.15168662	1	-0.13925211	1	-0.12500228	1	-0.11020228	1
15		-0.16495079	2	-0.12869680	2	-0.11486229	2	-0.09921245	2	-0.08301245	2
22		"	2	"	2	"	2	"	2	-0.08218156	3
100		"	2	"	2	"	2	"	2	"	3
0	5	-0.16251892	0	-0.16251892	0	-0.16251892	0	-0.16251892	0	-0.16251892	0
8		-0.20370777	1	-0.16265378	1	-0.14721927	1	-0.12996944	1	-0.11133855	1
15		-0.20033149	2	-0.15447750	2	-0.13744299	2	-0.11859316	2	-0.09836227	2
100		"	2	"	2	"	2	"	2	"	2

Table 2.15

As is clear from table 2.15, a further reduction in memory requirements, is possible by storing the state functions $f_n(\xi)$ and optimal allocations $\hat{x}(\xi)$ for the values of ξ which changes the $f_n(\xi)$ or $\hat{x}(\xi)$. In this way for any stage the maximum number of times $f_n(\xi)$ or $\hat{x}(\xi)$ is to be stored can at the most be $[b/c_j]$, but programming may be slightly complicated. For large problems this technique may have an advantage; however, it requires high speed memory.

2.6. An Algorithm using Lagrangian multipliers technique

The author has evolved a computational technique for reliability optimisation subject to linear constraints using Lagrangian multipliers in cases where number of stages k is greater than the number of constraints m.

It has been observed previously in section 2.3.2 that the choice of $\bar{\lambda}$ is usually crucial and the solution obtained by the method of that section is usually an approximate due to the rounding off of the allocation results. However, in some cases this may yield true optimum also but no such assurance is valid.

The present algorithm aims at removing these snags.

Theorem: Let $f(\bar{x})$ be a concave function over the closed convex set X in E^k. Then any relative maximum of $f(\bar{x})$ in X is also the global maximum of $f(\bar{x})$ over X. If $f(\bar{x})$ is strictly concave then the point in X at which the global maximum is assumed is unique. If $f(\bar{x})$ is concave over a convex set X and if $f(\bar{x}) \in C^1$ (i.e. $f(\bar{x})$ and its first derivative are continuous over some subset of E^k), then $\nabla f(\bar{x}) \equiv 0$ at $\bar{x}^*$, $f(\bar{x})$ takes on its global maximum over X at $\bar{x}^*$.

Now the problem is to maximise $z = f(\bar{x})$ for $\bar{x} \geqslant 0$

satisfying $g_i(\bar{x}) = b_i$, $i = 1, ..m$; $m < k$ and $\bar{x} \equiv (x_1 , x_2 , ..x_k)$. It is assumed that $f \epsilon C^1$ and $g_i \subset C^1$, $i = 1, 2 ..m$. Also if $f(\bar{x})$ takes on a relative maximum at $\bar{x}^*$, the following k+m equations (necessary conditions) should be satisfied.

$$\frac{\partial f(\bar{x}^*)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(\bar{x}^*)}{\partial x_j} = 0 \qquad j=1, 2 ..k$$

$$g_i(\bar{x}^*) = b_i \qquad i=1, ...m \qquad (2.40)$$

The λ_i are uniquely determined for any such $\bar{x}^*$.

Obviously if solution to (2.40) is possible by some computational procedure this assumes

(i) The $\bar{x}$ are continuous variables.

(ii) Constraints are all having equality expressions.

The rounded off solution of $\bar{x}^*$ to integer form will be a feasible solution also to the problem where constraints are of the type $g_i(\bar{x}) \leqslant b_i$. Furthermore, the rounded off solution leaves some slack in each constraint and if it is possible to reduce these slacks with integer condition of the solution and at the same time modifying λ_i such that at least first k equations of (2.40) are satisifed and the k+1 to m equations of (2.40) are satisfied to an extent that no slack in resources b_i , is of the size that any equipment of j^{th} type $j=1..k$ is possible to be allocated.

Returning to the reliability problem we have similar to section 2.5.4

$$\text{maximise } z = \sum_{j=1}^{k} \log \left\{ 1-(1-p_j)^{x_j} \right\} \equiv f(\bar{x})$$

$$\text{subject to } x_j \geqslant 0 \qquad j = 1,..k$$

$$\text{and } \sum a_{ij} x_j \leqslant b_i \qquad i = 1, ..m \qquad (2.41)$$

where x_j are the system allocation (i.e. redundancies + 1 units) and p_j is the reliability of the j^{th} type component.

Modifying (2.41) constraints to equality form and writing the Lagrangian function we have,

$$F(\bar{x}, \bar{\lambda}) \equiv f(\bar{x}) + \sum_{i=1}^{m} \lambda_i \left[b_i - \sum_{j=1}^{n} a_{ij} x_j \right]$$

Applying necessary conditions of (2.40) and writing $q_j = 1 - p_j$, one gets

$$-\left\{ \frac{q_j^{x_j}}{1 - q_j^{x_j}} \right\} \log q_j - \sum_{i=1}^{m} \lambda_i a_{ij} = 0$$

$$\text{or} \left\{ \frac{q_j^{x_j}}{1 - q_j^{x_j}} \right\} \log q_j + \sum_{i=1}^{m} \lambda_i a_{ij} = 0 \qquad j=1..k$$

$$\text{and} \sum_{j=1}^{k} a_{ij} x_j = b_i \qquad i=1..m \qquad (2.42)$$

The algorithm now can be stated in the following steps:

1. Assume the values of λ_i, $i=1, ..m$ to start the process. (All λ_i can be assumed to be equal but $\neq 0$) so as to yield x_j within the feasible solution.

2. Calculate the values of x_j, $j=1..k$ using first k equations of (2.42).

3. Round off x_j's to lower integers say n_j's and calculate the slack s_i by

$$s_i = b_i - \sum_{j=1}^{k} a_{ij} n_j \qquad i=1..m$$

Obviously the slack will be non-negative since the n_j's form a feasible solution.

4. Calculate for each stage $w_j = \min \left\{ (s_i/a_{ij}),\ i=1..m \right\}$ $j=1,..k$ w_j provide an estimate of how much one can add to the existing allocation in each stage without violating any one of the m constraints.

5. Compute $w_j' = \langle w_j \rangle$ i.e. round off the w_j to lower integer. If all w_j are zero stop and print out the allocation n_j's.

6. If any one of w_j' is not zero then we compute $\Delta n_j = \langle \frac{w_j'}{l} \rangle$ where l is any arbitrary constant 1. This is done so as not to reach very close to any constraint 'too soon'; other-wise the local maximum situation will prevail.

7. Next the increase in objective function due to change in allocation from n_j to $(n_j + \Delta n_j)$ is computed for each stage. Here we can make use of the fact that increase in log (system reliability) will be equivalent to increase in log (reliability of the stage to which Δ n is added).

8. We add the increment in allocation n to that stage j' which gives maximum increase in the objective function and thus arrive at an allocation of $(n_1, n_2, \ldots n_j + \Delta n_j, \ldots n_k)$.

9. Using the allocation arrived in step 8 we compute λ_i, $i=1, ..m$ from the first k equation of (2.42). This indeed will result in a set of equations with their number greater than the number of unknowns i.e. $\lambda_i = 1, ..m$. The only possibility of solving such an over-determined system is by "the method of least squares" [25].

The solution can be obtained as follows:

Let $[A] \equiv \left[a_{ij}\ ,\ i=1,..k;\ \ j=1,..m\right]$ be the matrix of coefficients of λ_i's in (2.42)

Then

$$A^tA\ \bar{\lambda} = A^t\bar{d} \qquad (2.44)$$

where $\bar{\lambda}$ is a column vector of $[\lambda_i\ ,\ i=1,..m]$ and $\bar{d}$ is the right hand side of (2.42) after the values of m_j's have been substituted in first k equations of (2.42).

The system of equations of (2.44) would lead to unique values of λ_i , $i=1,..m$.

10. Using the above values λ_i's we re-compute x_j , $j=1,..k$ and return to step 3.

The above algorithm will terminate when λ_i , $i=1,..m$ and x_j , $j=1,..k$ are such that they satisfy the first k equations of (2.42) and the n_i's are such that slacks of m-constraint equations of (2.42) are reduced considerably and that no component of any type be added to the existing allocations without violating any one or more constraints.

Nothing can be said about the efficiency of the algorithm as at this time the general purpose program has not yet been developed.

However the algorithm seems to be quite convincingly appropriate.

CHAPTER 3

VARIATIONAL APPROACH

Moscowitz and Mclean [12] obtained the condition for minimum cost, if the reliability of the system is given a preassigned value, using variational approach. It may be stated here that the problem of finding minimum cost for a specified value of system reliability, is the same as optimising reliability with given cost constraint on the system. Moscowitz and Mclean method, however, was developed for the former case. The author has suggested an extension and generalisation of the technique for single and multiple cost constraints.

3.1. Condition for minimum cost

Let there be a basic system of k elements in series having $r_1, r_2 \dots r_k$ reliabilities and cost of $c_1, c_2 \dots c_k$ such that basic reliability of the system be

$$R_o = \prod_{i=1}^{k} r_i \tag{3.1}$$

and basic system cost

$$C_o = \sum_{i=1}^{k} c_i \tag{3.2}$$

The problem is therefore, to find redundancy allocation which gives minimum cost for the specified system realiability of R_g. Denoting the number of elements in stage i by m_i the reliability of stage i can be written as

$$R_i = 1 - q_i^{m_i} \tag{3.3}$$

where $q_i = 1 - r_i$, r_i is the reliability of each element in i^{th} stage and R_i is the reliability of m_i such elements in parallel. The system reliability therefore would be written as

$$R_s = \prod_{i=1}^{k} R_i \tag{3.4}$$

Since the cost of i^{th} group of parallel elements is $m_i c_i$, the total system cost is

$$C_s = \sum_{i=1}^{k} m_i c_i \tag{3.5}$$

The required result can be found by solving equations (3.3), (3.4), (3.5) as a variational problem and finding the distribution of m_i's for minimum cost. Introducing another variable a_i defined by

$$R_i = R_s^{ai} \tag{3.6}$$

It can be shown that a real positive number a_i between 0 and 1, can always be found to satisfy (3.6). Then from (3.3) and (3.6) each m_i can be written as

$$m_i = \frac{\log(1-R_i)}{\log q_i} = \frac{\log(1-R_s^{ai})}{\log q_i} \tag{3.7}$$

and the system cost and reliability can be given by

$$C_s = \sum_{i=1}^{k} m_i c_i = \sum_{i=1}^{k} \frac{c_i \log(1-R_s^{ai})}{\log q_i} \tag{3.8}$$

$$R_s = \prod_{i=1}^{k} R_i = \prod_{i=1}^{k} R_s^{ai} = R_s^{\left[\sum_{i=1}^{k} a_i\right]} \qquad (3.9)$$

In order (3.9) can be valid, it is required that

$$a = \sum_{i=1}^{k} a_i = 1 \qquad (3.10)$$

It is now possible to optimise cost with reliability. This occurs for distribution of a_i's which gives stationary value for the ratio C_s/R_s. The particular distribution of a_i's is to be found which satisfy

$$\delta\left(\frac{C_s}{R_s}\right) = 0 \quad \text{or} \quad \frac{\delta C_s}{C_s} - \frac{\delta R_s}{R_s} = 0 \qquad (3.11)$$

subject to the constraint that $\delta a = \sum_{i=1}^{k} \delta a_i = 0$ (3.12)

If λ is a real constant then simultaneous solution of (3.8), (3.9), (3.10) and

$$\frac{\delta C_s}{C_s} - \frac{\delta R_s}{R_s} - \lambda \delta a = 0 \qquad (3.13)$$

will provide the distribution of a_i's for stationary value of C_s/R_s. Now

$$\delta R_s = R_s(a+\delta a) - R_s(a)$$

$$= R_s^{\left[\sum_{i=1}^{k} (a_i + \delta a_i)\right]} - R_s^{\left[\sum_{i=1}^{k} a_i\right]} = R_s(R_s^{\left[\sum \delta a_i\right]} - 1)$$

since $\delta a = \sum \delta a_i = 0$

$$\text{therefore} \quad \frac{\delta R_s}{R_s} = 0 \tag{3.14}$$

Similarly the variation of C_s with a is given by,

$$\delta C_s = C_s(a+\delta a) - C_s(a)$$

$$= \sum_{i=1}^{k} \frac{c_i}{\log q_i} . \log\left[1-R_s^{(a_i+\delta a_i)}\right] -$$

$$\sum_{i=1}^{k} \frac{c_i}{\log q_i} . \log(1-R_s^{a_i})$$

$$= \sum_{i=1}^{k} c_i' \log\left[\frac{1-R_s^{(a_i+\delta a_i)}}{1-R_s^{a_i}}\right] \tag{3.15}$$

$$\text{where } c_i' = \frac{c_i}{\log q_i}$$

If it is assumed that R_s is quite high, i.e. very close to 1 and q_s is very small

$$\delta C_s = \sum_i c_i' \log\left[\frac{1-(1-q_s)^{a_i+\delta a_i}}{1-(1-q_i)^{a_i}}\right]$$

$$= \sum_i c_i' \log\left[1+\frac{\delta a_i}{a_i}\right]$$

then to the first approximation

$$\frac{\delta C_s}{C_s} \approx \frac{\sum_i c_i \frac{\delta a_i}{a_i}}{C_s} \tag{3.16}$$

Substitution of (3.14) and (3.16) in (3.13) yields

$$\sum_i \frac{c_i'}{C_s a_i} \delta a_i - \lambda \sum_i \delta a_i = 0 \qquad (3.17)$$

This can be satisfied if

$$a_i = \frac{c_i'}{\lambda C_s} \qquad (3.18)$$

Solving for λ, realising $\sum_i = a_i = 1$

$$\therefore \lambda = \sum_i \frac{c_i'}{C_s} \qquad (3.19)$$

Substituting (3.19) in (3.18),

$$a_i = \frac{c_i'}{c_o'} = \frac{c_i/\log q_i}{\sum_j c_j/\log q_j} \qquad (3.20)$$

Therefore minimum cost can be obtained for the distribution a_i given by (3.20) and substitution of (3.20) in

$$m_i = \frac{\log (1-R_s^{a_i})}{\log q_i}$$

yields the values of m_i (i = 1, k): the elements in each stage with the total cost as

$$C_s = \sum_{i=1}^{k} m_i c_i = \sum_{i=1}^{k} c_i \frac{\log(1-R_s^{a_i})}{\log q_i}$$

3.2. Procedure for calculating optimum allocations

The general procedure for determining the optimum allocations can be outlined as follows:

1. Using the cost and reliability data about each element type a_i's using equation (3.20) are calculated and the calculated values can be checked by finding their sum which should be equal to unity, i.e.

$$\sum_{i=1}^{k} a_i = 1$$

2. For the given system reliability R_g and unreliabilities of each element type one can calculate the values of m_i's, the probable number of elements in each stage, using equation (3.7).

3. Usually the values so calculated for m_i's will not be integers and as the m_i's can only have integer values, so the values of m_i's obtained in step 2 are rounded off to the lower integer values.

4. Now as the reliability of the system will fall short of the given system reliability due to truncation of the values of m_i's the further improvement in system reliability can be obtained by adding successively the element types that yield minimum increase in cost for a certain increase in reliability.

5. Therefore the desirability factors F_i's for each stage are calculated as defined by

$$F_i = \frac{\Delta R_s / R_s}{c_i / C_s} \qquad (3.21)$$

where, F_i = the desirability factor for adding a unit or element to the i^{th} group;

R_s , C_s= system reliability and cost before adding the unit to i^{th} group;

c_i = cost of adding a unit to i^{th} . tage.

However it can be shown that

$$\frac{\Delta R_s}{R_s} = \frac{\Delta R_i}{R_i} \qquad (3.22)$$

where R_i is the reliability of i^{th} group before the addition of new unit to that stage and ΔR_i is the increase in reliability

? of that stage after new unit has been added. Therefore (3.21) can be written as

$$F_i = \frac{\Delta R_i/R_i}{c_i/C_s} \qquad (3.23)$$

To show (3.22) holds good one can write that

$$R_s = \prod_{i=1}^{k} R_i$$

and the reliability of the system R_s', after a unit to i^{th} stage has been added will be

$$R_s' = \frac{1}{R_i} \left(\prod_{i=1}^{k} R_i \right) (R_i + \Delta R_i) = \frac{R_s(1 + \Delta R_i)}{R_i}$$

also $\Delta R_s = R_s' - R_s$, therefore,

$$\Delta R_s = R_s \frac{\Delta R_i}{R_i}$$

or $$\frac{\Delta R_i}{R_i} = \frac{\Delta R_s}{R_s}$$

6. Once all F_i's have been calculated in step 5, a new element is

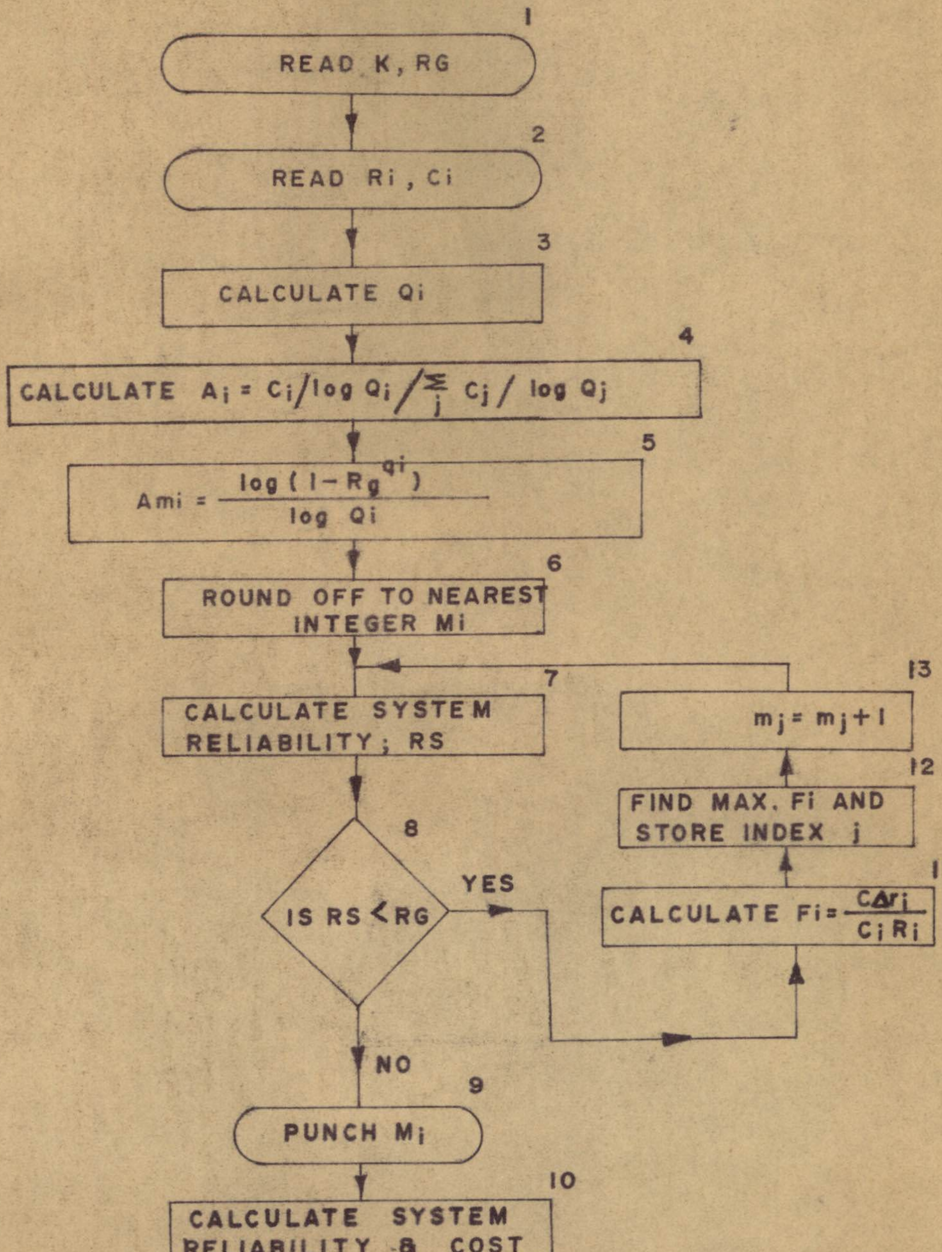

1
READ K, RG
2
READ Ri, Ci
3
CALCULATE Qi
4
CALCULATE $A_i = C_i / \log Q_i / \sum_j C_j / \log Q_j$
5
$Ami = \frac{\log(1-Rg^{qi})}{\log Qi}$
6
ROUND OFF TO NEAREST INTEGER Mi
7
CALCULATE SYSTEM RELIABILITY; RS
8
IS RS < RG
YES
NO
9
PUNCH M_i
10
CALCULATE SYSTEM RELIABILITY & COST
STOP
11
CALCULATE $Fi = \frac{C\Delta r_i}{C_i R_i}$
12
FIND MAX. Fi AND STORE INDEX j
13
$m_j = m_j + 1$

FIG. 3·1 FLOW CHART FOR VARIATIONAL METHOD.

added to the stage j for which the F_i calculated is maximum.

7. New reliability and cost of the system is calculated if the reliability of the system is now more than or equal to the given reliability. The allocation obtained so far is the optimum value, otherwise the steps 5 & 7 are repeated till the system reliability is at least equal to R_g or greater than this.

The method has been programmed on the computer and the flow chart for the same is given in Fig. 3.1. All the steps mentioned above have been shown clearly in the flow chart.

3.2.1. Illustrative Example

For the system reliability of 0.99, it is required to find the optimum allocations for the system of section (2.3.1). The results obtained by the procedure described in previous section (3.2) and programmed on IBM 1620 Computer, have been listed in table 3.1.

3.3. Allocations with given cost constraint

It has been pointed out in earlier sections that Moscowitz and Mclean's method requires a prior estimation of the reliability of a system. In other words, given the reliability index for a system the allocations to meet that requirement can be found by their method. In case the cost constraint on the system is given, the method described in earlier sections cannot be directly applied. However, the author tackled this problem by roughly estimating the reliability of a system within the specified cost constraint and thereby finding the allocation using the method of [12]. This allocation will, however, be updated by successive addition of units to the stages where normalised reliability to

Stage	1	2	3	4	Remarks
a_i's	0.09967	0.25537	0.32786	0.31709	$\sum a_i = 1$
am_i's (calculated)	4.29129	4.95561	4.12391	3.03107	
Truncated m_i's	4	4	4	3	
System reliability (R_s)	0.98321				$R_s < R_g$
System cost (C_s)	41.1				Cost ratio ($CR=C_s/c_o$) 3.60526
F_i's	0.04391	0.10215	0.03555	0.02629	Maximum F_i for stage 2
New allocation m_i's	4	5	4	3	
R_s	0.98874				$R_s < R_g$
C_s	43.4				CR = 3.80702
F_i's	0.04637	0.03218	0.03754	0.02776	Maximum F_i for stage 1
New allocation m_i's	5	5	4	3	
R_s	0.99000				$R_s=R_g$; calculation stops.
C_s	44.6				CR = 3.91228

Table 3.1

cost ratio is the highest, till the final allocation is nearer to the boundary of the constraint. This method will provide near-optimum or optimum solution conveniently fast and without much complexity involved. In all the cases studied by the author this method has yielded an optimum solution to the allocation problem with given cost constraints. Moreover, a fast and approximate solution is much better than slow, complex and an accurate method.

3.3.1. Example:

Taking the example of section (2.3.1) except that cost constraint specified is 56 units:

To assess an approximate reliability of the system the method of section (2.4.2) is used, i.e. one can start with system allocation as (1, 1, 1, 1) and go on adding one unit at a time to each stage till a constraint is violated. If this procedure is followed, then for the problem under consideration a table similar to table 2.9 is developed and the allocation after which cost constraint is violated will be (5, 5, 5, 4) and corresponding system reliability computed is given as 0.99577. With this system reliability, the allocation using the method described in section (3.2) is found and successive steps in the solution are provided in table 3.2.

Table 3.2

Allocation	Reliability	Cost
5, 5, 4, 3	0.99000	44.6
5, 5, 5, 3	0.99291	48.0
5, 6, 5, 3	0.99461	50.3
5, 6, 5, 4	0.99747	54.8

3.4. Allocations with Multiple Constraints

The variational method of section (3.1) gives approximately optimum allocation due to the rounding.off of the actual allocation having fractional parts also and then applying trial-and-error method to 'fill-in' the resources available. In fact, it provides a shortcut to the optimum in undominated sequence method of section (2.3). There instead of starting from (0, 0, 0...0) redundancy allocation, one starts from near-optimum allocation by the use of variational method.

The author makes use of this fact by simultaneous solution of allocation problem starting from the individual optimal solution of the multiple constraint problem. The basic facts supporting this approach are:

1. Reliability is an increasing function of allocations and thereby individual 'cost' variables such as cost, weight, volume etc.

2. The true optimal allocation should be a point on the undominated sequences generated for individual 'cost' variables as the problem is to optimise reliability with respect to all these variables.

3. In absence of a point mentioned in 2 above any allocation point on the higher ridge (reliability) corresponding to any one of the 'cost' variables will be good enough to be called near optimum without violating any of the constraints.

Therefore the procedure requires the following steps to be carried out in sequence:

1. Within the feasible solution domain, the attainable reliability can be roughly estimated using a table such as 2.9, by adding

one unit at a time till some constraint is violated.

2. Using this rough estimate of reliability (which is often too near to the actual optimal reliability) we find out the individual optimal allocations with respect to each 'cost' variable involved by the variational method of section (3.1).

3. From these individual optimal points, we pick up the allocation which is lying on higher reliability plane and add optimally (i.e. where the normalised change of reliability to individual 'cost' variable ratio is high) to each allocation other than the allocation corresponding to this maximum reliability allocation, i.e. following maximum gradient path.

4. Obviously, by moving all other allocations to next higher point we increase the reliability of the allocations corresponding to all cost variables except one. Now if some allocation other than the previous one giving maximum reliability provides a higher reliability point, we do not change that allocation and add optimally one unit to all other allocations and calculate new reliabilities corresponding to these new allocations. If after step 3, the point chosen earlier is still higher, then step 3 is repeated.

5. This process continues till all allocations give the same reliability or a common allocation, within the feasible solution domain and also check whether the allocation if changed to higher reliability point would violate any constraint or not. If not, then the process is continued; otherwise stopped. The common allocation is the optimum allocation.

6. If, on the other hand, the common reliability point is not

obtained, then the allocation (within the feasible solution domain near the boundary) with the highest reliability will provide a near optimal solution.

A computer program has been written using above approach and several problems available were tried by this method. In all cases, the author got the optimal allocation without difficulty. The flow chart for the procedure described above is given in Fig. 3.2.

3.4.1. Example

For illustration, the solution to problem of section 2.3.1(ii) is presented in the steps enumerated above.

The attainable reliability for the two-'cost' problem of section (2.3.1) without violating any of the two constraints, i.e. cost and weight, has been worked out earlier in table 2.9. The reliability figure thus found has been obtained as 0.99577 for the allocation (5, 5, 5, 4). Using this figure for the attainable reliability the optimum allocation with respect to cost and weight individually was found out by variational method as (5, 5, 4, 3) and (4, 6, 4, 3) respectively. Further simultaneous solution has been shown in table 3.3 in detail, leading to an optimal allocation of (5, 6, 5, 4).

As is clear from table 3.3, in almost six steps, the final allocation providing optimum has been obtained. This same computer program was used for other problems also. The author did not have any difficulty in arriving at the solution. The process is fast as there are no complicated calculations involved in the procedure. A three linear-constraint problem having cost, weight and volume as constraint was also tried and the result was arrived at fast.

Step	Cost-Allocation					Weight-Allocation				
	Allocation	Cost	Reliability	Max. F_i	Stage changed	Allocation	Weight	Reliability	Max. F_i	Stage changed
0	5 5 4 3	44.6	0.99000	-	-	4 6 4 3	97.0	0.99042	-	-
1	5 5 5 3	48.0	0.99291	0.03858	3	4 6 4 3	97.0	0.99042	-	-
2	5 5 5 3	48.0	0.99291	-	-	4 6 4 4	104.0	0.99327	0.03989	4
3	5 6 5 3	50.3	0.99461	0.03559	2	4 6 4 4	104.0	0.99327	-	-
4	5 6 5 3	50.3	0.99461	-	-	4 6 5 4	112.0	0.99619	0.03824	3
5	5 6 5 4	54.8	0.99747	0.03218	4	4 6 5 4	112.0	0.99619	-	-
6	5 6 5 4	54.8	0.99747	-	-	5 6 5 4	117.0	0.99747	0.02872	1
7	5 6 5 4	54.8	0.99747	-	-	5 6 5 4	117.0	0.99747	-	-

Table 3.3

The advantage of this method lies in the fact that this procedure can be applied to any allocation problem having many linear-constraints, without much difficulty. However, it should be mentioned that the assumption in this procedure is: the system has multi-component stages with heterogeneous component 'costs'.

3.5. Optimisation Using Maximum Principle

Tillman and others [21] presented a computational procedure using Di~~eece~~screte-Maximum Principle for the optimum design of a multi-stage parallel system. The author observed few drawbacks of this method which are worth mentioning before one makes a choice of using the procedure outlined. The objective function [21] maximised was taken as system profit expressible in the form (using the notations of [21])

$$N_p = PR_s - \sum_{n=1}^{k} C^n\theta^n \tag{3.24}$$

where N_p is the net profit accruing out of a system having reliability as R_s , assuming the profit provided on system operating successfully is P, C^n and θ^n being the cost of a unit and the number of units, used at n^{th} stage. The Hamiltonian and adjoint variables were defined as

$$H^n = z_1^n x_1^{n-1}\left[1-(1-R^n)^{\theta^n}\right] + z_2^n\left[x_2^{n-1}+C^n\theta^n\right] \tag{3.25}$$

$$n = 1, 2, ..k$$

and

$$z_1^{n-1} = \frac{\partial H^n}{\partial x_1^{n-1}} = z_1^n\left[1-(1-R^n)^{\theta^n}\right]$$

$$z_2^{n-1} = \frac{\partial H^n}{\partial x_2^{n-1}} = z_2^n \quad , \quad n = 1, 2, ...k$$

$$\text{with } z_1^k = P, \quad z_2^k = -1$$

where x_1^n and x_2^n are the reliability of n^{th} stage and sum cost upto and including n^{th} stage respectively. k is the total number of stages and R^n is the reliability of n^{th} type unit.

$$\text{Further } x_1^n = x_1^{n-1}\left[1-(1-R^n)^{\theta^n}\right]$$

$$x_2^n = x_2^{n-1}+C^n\theta^n \qquad n=1,\ 2,\ ..k \tag{3.26}$$

$$\text{with } x_1^0=1 \text{ and } x_2^0 = 0$$

After applying the necessary condition for optimality, i.e. $\frac{\partial H^n}{\partial \theta^n} = 0$ and further manipulation Tillman [21] arrives at a condition

$$P \prod_{n=1}^{k}(1-y^n) = \frac{a^1(1-y^1)}{y^1} = \frac{a^2(1-y^2)}{y^2} = \ldots = \frac{a^k(1-y^k)}{y^k} \tag{3.27}$$

where the different quantities are defined as

$$y^n = (1-R^n)^{\theta^n} = (U^n)^{\theta^n} \tag{3.28}$$

$$a^n = \frac{-C^n}{\log_e U^n}$$

Tillman [21] proposed to solve for θ^n, n=1, 2, ..k in (3.27) by t the Falsi iteration method and the steps involved can be summarised as follows:

1. $y^1(1)$ and $y^1(2)$ are assumed with the condition $0 < y^1 < 1$.

2. E is computed by

$$E = a^1(1-y^1)/y^1$$

3. From the following expression y^n, n= 2, 3..k are computed

$$y^n = \frac{a^n}{E+a^n}$$

4. Compute $$S = P \prod_{n-1}^{k} (1-y^n)$$

5. The error $E_R = S-E$ is computed.

6. A new trial value $y^1(3)$ is computed from the following extrapolation:

$$y^1(3) = \frac{y^1(2)-E_R(2)y^1(1)/E_R(1)}{1.0-E_R(2)/E_R(1)} \qquad (3.29)$$

7. Steps 2-5 are repeated to obtain ER(3).

8. A check is made if $\left\{|E_R(3)|-E_{Rmax}\right\} \leq 0$.
If the check is satisfied $y^1(3)$ is the required y^1 and also $y^n(3)$, n= 2, 3..k. θ^n can be computed from (3.28). If not, then $y^1(1)$ and $y^1(2)$ are replaced by $y^1(2)$ and $y^1(3)$ respectively and go to step 6.

The above procedure was programmed on IBM 1620 and tried and tested using the problems of [21]. The computer program is given in Appendix F and the results for the eight-stage problem of Table 3.4 are given in Table 3.5. Starting values of $y^1(1)$ and $y^1(2)$ were taken as 0.1 and 0.2.

It was observed that with the data given in Table 3.4 as such was used the solution converged to an absurd result with some θ^n being zero thereby giving system reliability as zero.

Table 3.4: P = 100.0

Stage	1	2	3	4	5	6	7	8
R	0.90	0.75	0.65	0.80	0.85	0.95	0.75	0.60
C	0.5	0.4	0.9	0.7	0.7	0.4	1.0	0.8

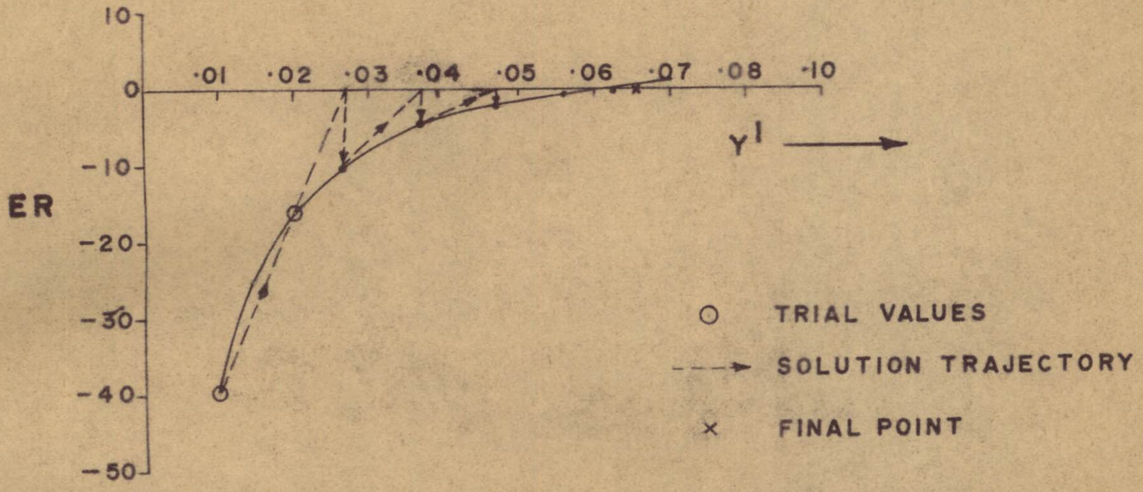


FIG.3·3 CONVERGENCE OF Y^I IN THREE STAGE PROBLEM.

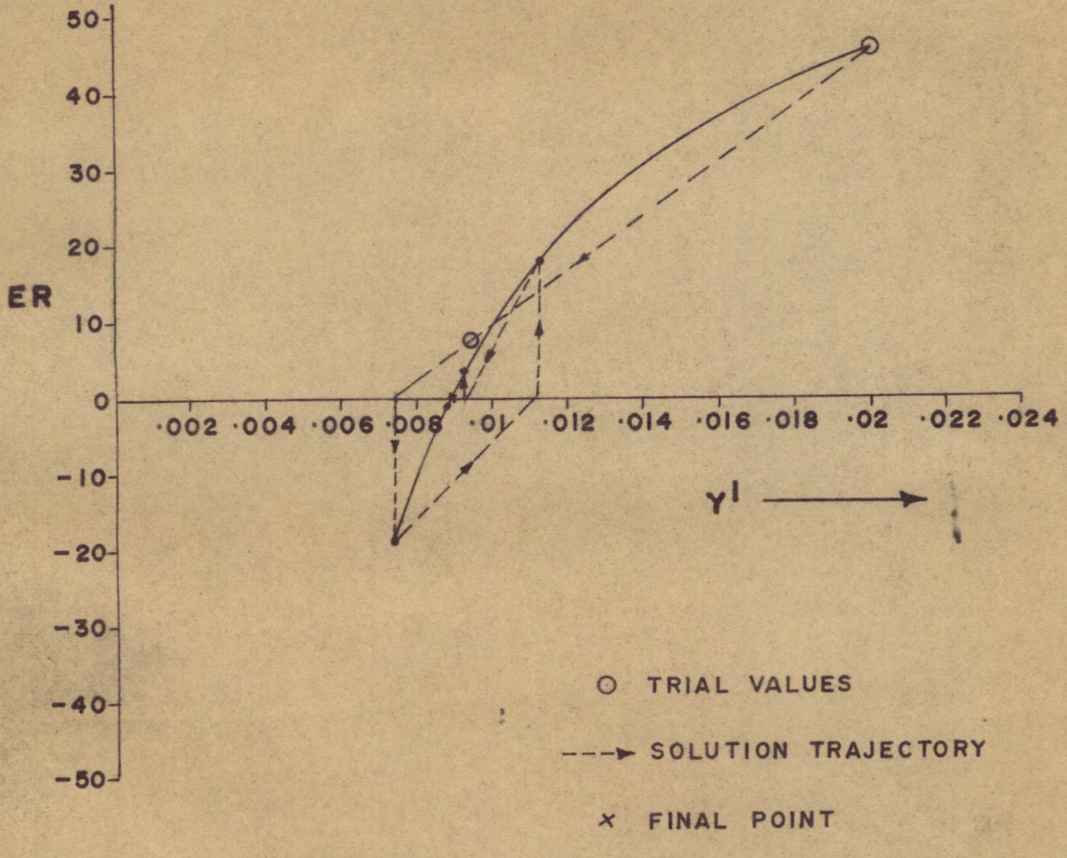


FIG.3·4 CONVERGENCE OF Y^I IN EIGHT STAGE PROBLEM.

However after slight data manipulation as regards the order of the stages the solution converged and the Table 3.5(a) pertains to such a case. The convergence was obtained in 9 steps. It really does not matter which stage comes first and which last as far as optimisation technique is concerned.

It was observed that if the first stage has low reliability the problem really gave sensible results, otherwise not. Several combinations of stage data were run and the experience substantiated the above statement. However, it may be noted that in either case the solution did converge. The values of $y^1(1)$ and $y^1(2)$ were also changed and interchanged. In some cases solution may diverge also but the values of $y^1(1)$ and $y^1(2)$ which converged the solution also may be absurd if the order of stage reliability was not heeded upon.

The table 3.5(b) also shows the order of stage data fed for which the converged solution was optimum. The convergence track followed is illustrated in Fig.3.4. The 3-stage problem of [21] was also tried and a typical solution pattern for y^1 is shown in Fig. 3.3.

The conclusion derived, therefore, is if one 'feeds in' data without any outsight then he would not know whether the solution is an optimum one or an absurd one.

3.6. Discrete Maximum Principle with Linear Constraints

The problem of maximising reliability of a system of ks stages with linear constraints on cost, weight, volume etc., is usually encountered in practice than the problem of section 3.5.

Tillman [23] gave analysis and outlined the procedure for non-linear constraints. The author ~~here presents the~~ analysis for linear ~~constraints problem~~ usually faced. It must be mentioned that the procedure would also slightly change with linear constraints. The problem of usual three constraints is discussed.

Let c^n, w^n, v^n, R^n be the cost, weight, volume and reliability of an element, respectively at n^{th} stage and θ^n be the number of elements in parallel at n^{th} stage. If the three constraints specified are

$$\sum_{n=1}^{k} v^n\theta^n \leqslant V \text{ , } \quad \sum_{n=1}^{k} c^n\theta^n \leqslant C \quad \text{and} \quad \sum_{n=1}^{k} w^n\theta^n \leqslant W$$

then the problem is to maximise the system reliability subject to above constraints.

State variables of the system may be defined as

$$\begin{aligned}
&\text{(volume)} && x_1^n = x_1^{n-1}+v^n\theta^n \text{ ; } x_1^0=0,\ x_1^k \leqslant V \\
&\text{(cost)} && x_2^n = x_2^{n-1}+c^n\theta^n \text{ ; } x_2^0=0,\ x_2^k \leqslant C \\
&\text{(weight)} && x_3^n = x_3^{n-1}+w^n\theta^n \text{ ; } x_3^0=0,\ x_3^k \leqslant W \\
&\text{(reliability)} && x_4^n = x_4^{n-1}+\ln\left\{1-(1-R^n)^{\theta^n}\right\} \text{ ; } x_4^0=0
\end{aligned} \qquad (3.30)$$

$$n=1,\ 2,..k$$

The objective function to be maximised is

$$S = \sum_{n=1}^{k} \ln \left\{1-(1-R^n)^{\theta^n}\right\} = \sum_{i=1}^{4} c_i x_i^k = x_4^k$$

$$c_i=0 \text{ , } i=1,\ 2,\ 3 \text{ and } c_4=1$$

The Hamiltonian and adjoint variables of the system are

Steps	1	2	3	4	5	6	7	8	9
y^1	0.01000	0.02000	0.00747	0.01120	0.00939	0.00890	0.00901	0.00900	0.00900
E	86.4354	42.7811	11.5991	77.0574	92.0111	97.1295	95.9898	96.0380	96.0386
$E_R(J)$	9.17437	45.44977	-19.28472	18.03360	3.85825	-1.04755	0.04680	0.00055	-0.00001

Table 3.5(a) : E_{Rmax} taken as 0.0001

Stage No.	8	3	7	2	4	5	1	6
θ^n	5.13977	4.50326	3.53378	4.19151	3.35634	2.93370	2.64667	2.19633
θ^n	5	5	4	4	3	3	3	3

Table 3.5(b) : Cost = 20.6, N_p = 75.64213, R_s = 0.962421

$$H^n = \sum_{i=1}^{4} z_i^n \, x_i^n$$

$$= z_1^n \left\{ x_1^{n-1} + v^n \theta^n \right\} + z_2^n \left\{ x_2^{n-1} + c^n \theta^n \right\} + z_3^n \left\{ x_3^{n-1} + w^n \theta^n \right\}$$

$$+ z_4^n \left\{ x_4^{n-1} + \ln(1-(1-R^n)^{\theta^n} \right\} \qquad (3.31)$$

$$n = 1,\ 2,\ ..k$$

$$z_i^{n-1} = \frac{\partial H^n}{\partial x_i^{n-1}} = z_i^n \quad , \quad i=1,\ 2,\ 3,\ 4 \ ; \quad z_4^k = c_4 = 1 \qquad (3.32)$$

Differentiating (3.31) and equating to zero, one gets

$$z_1^n v^n + z_2^n c^n + z_3^n w^n + z_4^n \quad \frac{-(1-R^n)^{\theta^n} \ \ln(1-R^n)}{1-(1-R^n)^{\theta^n}} = 0 \qquad (3.33)$$

Whenever the jth constraint, represented by x_j^k is active, this has the effect of fixing its boundary value. Therefore,

$$z_i^k = c_i \qquad i \neq j \qquad (3.34)$$

If the first constraint is active, we have

$$z_i^k = c_i = 0 \quad i = 2,\ 3$$

$$z_i^n = 0, \quad n = 1,2,..k, \quad i = 2,\ 3$$

Thus (3.33), if $U^n = 1-R^n$ provides

$$z_1^n = \frac{1}{v^n} \left[\frac{(U^n)^{\theta^n} \ln U^n}{1-(U^n)^{\theta^n}} \right] \qquad (3.35)$$

and

$$\theta^n = \frac{1}{\ln U^n} \left[\ln(z_1^n v^n) - \ln(\ln U^n + z_1^n v^n) \right] \qquad (3.36)$$

Similarly z_2^n and z_3^n can be derived and they will be identical with (3.35) except in place of v^n, there will be c^n or w^n respectively.

Expression for θ^n will also be similar to (3.36).

It may be remembered here that (3.36) although expresses θ^n in terms of the known quantities but this does not ease our labour due to the computational difficulties. The second ln term has argument as negative due to $(\ln U^n + z_i^n v^n)$ in (3.36) and the computer can just not calculate the log of a negative quantity. Therefore although at first sight it seemed that we can do away with Newton's method but unfortunately this is not so . We have to solve (3.35) with known values of z_1^n and other quantities for θ^n in convenient form by Newton's method. The rest of the procedure is actually the same as given in Appendix G. This method was tried by writing a computer program and taking the problem from reference [26] of three constraints. The results were highly satisfactory. The input data and output results are shown in tables 3.6 and 3.7 respectively. The computer program is shown in Appendix H-1.

Table 3.6 - Input Data

Stage	1	2	3	Constraints
Cost	4.0	8.0	6.0	CG = 50.0
Weight	6.0	6.0	10.0	WG = 52.0
Volume	10.0	5.0	10.0	PG = 65.0
Reliability	0.86	0.91	0.96	

Table 3.7 - Output Results

θ^1 Assumed	Allocations (Actual)		Allocations Rounded off			PS	CS	WS
	θ^2	θ^3	θ^1	θ^2	θ^3			
1.0	1.152	0.746	1.0	1.0	1.0	25.0	18.0	22.0
1.3	1.413	0.938	1.0	1.0	1.0	25.0	18.0	22.0
1.6	1.667	1.124	2.0	2.0	1.0	40.0	30.0	34.0
1.9	1.917	1.310	2.0	2.0	1.0	40.0	30.0	34.0
2.2	2.164	1.495	2.0	2.0	1.0	40.0	30.0	34.0
2.5	2.411	1.680	2.0	2.0	2.0	50.0	36.0	44.0
2.8	2.370	1.705	3.0	2.0	2.0	60.0	40.0	50.0

CHAPTER 4

OPTIMISATION OF SYSTEM RELIABILITY WITH NON-LINEAR CONSTRAINTS

In Chapter 2, optimising procedures were described which were formulated for linear constraint problems only. The obvious assumption was that the constraint variables increase linearly with the number of units used in redundancies. This, however, is not true whenever addition of a new unit to the existing one entails a huge amount of connecting accessories and the cost, weight or volume of these connecting accessories increase exponentially or according to some other law as the number of additional redundant units are introduced. Tillman [23] suggested some combination of a linear and exponential terms in the constraints.

4.1. Proschan and Bray Extension

The problem of multiple non-linear constraint can be solved by Proschan and Bray [15] approach very comfortably. Here instead of the usual linear constraint any non-linear constraint can be taken and still the problem can be solved in the same manner as for linear constraints. For example, if the problem of section (2.3.1) of Chapter 2 is taken except that the constraint now specified is a non-linear cost constraint given by

$$\sum_{i=1}^{k} c_i \left\{ (n_i) + \exp(\frac{n_i}{4}) \right\} \leqslant C$$

where c_i is the cost of i^{th} type of component and n_i is the number of redundant units at i^{th} stage with total system cost not to exceed C, say 100 units. The term

STAGE I

STAGE 2	COST UNRELIABILITY	3 6.13 .008	4 8.06 .0016	5 10.02 .00032	6 12.58 .000064
4	15.68 .0081	21.81 .0161	23.74 .0097	25.70 .00842	28.26 .008164
5	19.51 .00243	25.64 .01043	27.57 .00403	29.53 .00275	32.09 .002494
6	24.10 .000729	30.23 .008729	32.16 .002329	34.12 .001049	36.68 .000793

FIG. 4·1 UNDOMINATED ALLOCATION FOR NON-LINEAR COST PROBLEM.

STAGE 3

STAGE 4	COST UNRELIABILITY	3 17.39 .0156	4 22.81 .00391	5 28.84 .00098	6 35.61 .000245
2	16.41 .0225	33.80 × .0381	29.22 × .02641	45.25 × .02348	52.02 × .022745
3	23.00 .00338	40.39 × .01898	45.81 .00729	51.84 .00436	58.61 .003625
4	30.02 .000506	47.41 × .016106	52.83 × .004416	58.86 .001486	65.63 .000751
5	38.20 .000076	55.59 × .015676	61.01 × .003986	67.04 × .001056	73.81 .000321

FIG.4·2UNDOMINATED ALLOCATION FOR NON-LINEAR COST PROBLEM.

$c_i \left\{ \exp(\frac{n_i}{4}) \right\}$ gives the cost of the connecting accessories for providing n_i redundancies. Figs. 4.1 and 4.2 give the undominated allocation when stages 1-2 and 3-4 are combined. The dominating sequences arising out of these are shown in tables 4.1 and 4.2. Finally, table 4.3 provides the dominating sequence for combined stages 1, 2, 3 and 4. One can pick out the allocation for a particular value of given cost constraint. It is however not difficult to extend the same procedure for multiple non-linear constraint such as weight and volume etc. In preparing table 4.3, no effort has been made to put a tolerance on reliability or cost factor. However, one can specify a tolerance on cost or reliability so that the length of dominating sequence of table 4.3 can be reduced at will. The allocation from table 4.3 for cost not to exceed 100 units is (5, 6, 6, 4).

In calculating the starting values of redundant units in each stage one can first calculate the term

$$n_i + \exp(\frac{n_i}{4})$$

for all values of n_i $o \leqslant n_i \leqslant [C/c_i]$ or $[W/w_i]$. Obviously, one will not have the same number of redundant units as for linear constraint. They will be less in case of non-linear constraint than linear for the same C specified. After all the values of n_i and corresponding $n_i + \exp(\frac{n_i}{4})$ or some such expression have been calculated, we assess an approximate system reliability by adding one unit at a time till some constraint is violated or alternatively

$$[C/c_i] \text{ or } [W/w_i] \geqslant n_i + \exp(n_i/4)$$

after the process is same as described for linear constraints.

Table 4.1 - Dominating Sequence for Stages 1 & 2

Dominating sequence	No. of equipment		Unreliability	Reliability	Cost
	Stage I	Stage II			
1	4	4	.0097	.9903	23.74
2	5	4	.0084	.9916	25.70
3	4	5	.0040	.9960	27.57
4	5	5	.0027	.9973	29.53
5	6	5	.0025	.9975	32.09
6	4	6	.0023	.9977	32.16
7	5	6	.0010	.9980	34.12
8	6	6	.0007	.9993	36.68

Table 4.2 - Dominating Sequence for Stages 3 & 4

Dominating sequence	No. of equipment		Unreliability	Reliability	Cost
	Stage III	Stage IV			
1	4	3	.0073	.9927	45.81
2	5	3	.0074	.9956	51.84
3	6	3	.0036	.9964	58.61
4	5	4	.0015	.9985	58.86
5	6	4	.0008	.9992	65.63
6	6	5	.0003	.9997	73.81

Table 4.3 - Dominating Sequence for Stages (1 & 2) and (3 & 4) combined

Sequence	No. of equipment in stages				Unreliability	Reliability	Cost
	I	II	III	IV			
1	5	5	4	3	.0100	.9900	75.34
2	6	5	4	3	.0098	.9902	77.90
3	4	6	4	3	.0096	.9904	77.96
4	4	5	5	3	.0084	.9916	79.41
5	5	6	4	3	.0083	.9917	79.93
6	5	5	5	3	.0071	.9929	81.37
7	6	5	5	3	.0069	.9931	83.93
8	4	6	5	3	.0067	.9933	84.00
9	5	6	5	3	.0054	.9946	85.96
10	5	5	5	4	.0042	.9958	88.39
11	6	5	5	4	.0040	.9960	90.95
12	4	6	5	4	.0038	.9962	91.02
13	5	6	5	4	.0025	.9975	92.98
14	6	6	5	4	.0022	.9978	95.54
15	5	6	6	4	.0018	.9982	99.75
16	6	6	6	4	.0015	.9985	102.31
17	5	6	6	5	.0013	.9987	107.93
18	6	6	6	5	.0010	.9990	110.49

4.2. Discrete Maximum Principle

The maximum principle [23]can also be applied to the problems of optimisation of reliability with non-linear constraints. The approach is simple and can be applied to any number of non-linear or combination of linear and non-linear constraints. As the variable of number of redundant units is considered as continuous, the method however does not provide an exact solution or true optimum of the problem and one has to round off the allocation obtained by this method to the nearest integer. However, in the absence of any other fast method, the method has a definite advantage.

The problem of maximisation and its condition of optimality has been discussed in Appendix G. The detailed flow chart for the procedure of solution on computer for a three-constraint problem is given in Fig. 4.3

The 5 stage problem solved for illustration is given as

Stage No. n	Reliability R^n	Volume p^n	Cost c^n	Weight w^n	Constraints
1	0.80	1	7	7	Volume 110
2	0.85	2	7	8	Cost 175
3	0.90	3	5	8	Weight 200
4	0.65	4	9	6	
5	0.75	2	4	9	

The constraints are of the form in the notations of Appendix G.

(1) on weight and volume

$$\sum_{n=1}^{k} g_1^n(\theta^n) = \sum_{n=1}^{k} p^n(\theta^n)^2 \leqslant P$$

where $p^n = w^n v^n$ is the product of weight per unit and volume per unit at the n^{th} stage.

(2) on cost

$$\sum_{n=1}^{k} g_2^n\ (\theta^n) = \sum_{n=1}^{k} c^n(\theta^n + \exp(\theta^n/4)) \leqslant C$$

where $c^n\theta^n$ is the cost of units at n^{th} stage and $c^n(e)^{\theta^n/4}$ is the cost of additional connecting equipment.

(3) on weight

$$\sum_{n=1}^{k} g_3^n\ (\theta^n) = \sum_{n=1}^{k} w^n\theta^n\ \exp(\theta^n/4) \leqslant W$$

where $w^n\theta^n$ is the weight of the total units at the n^{th} stage. This is increased by the factor $\exp(\theta^n/4)$ due to the weight of interconnecting accessories.

The detailed computer program along with Newton's method subroutine is given in Appendix H. The results obtained by taking different values of θ^1 are shown in table 4.4 where for the value of θ^1 equal to 3 the solution is obtained and the final system allocation of (3, 2, 2, 3, 3) with volume, cost and weight of 83, 146.12 and 192.48 respectively. The redundancy allocation to each stage is therefore (2, 1, 1, 2, 2) with system reliability of 0.9045

Table 4.4

θ^1	System volume	System cost	System weight	Allocation to other stages			
				θ^2	θ^3	θ^4	θ^5
1.0	6.31	6.32	35.61	0.7495	0.6136	0.6569	0.8177
1.5	15.91	81.76	62.09	1.1485	0.9413	1.1167	1.2996
2.0	31.35	102.16	96.33	1.5568	1.2770	1.6602	1.8153
2.5	53.58	124.3	140.08	1.9707	1.6176	2.2713	2.3534
3.0	83.19	148.22	195.44	2.3881	1.9612	2.9311	2.9054

CHAPTER 5

INTEGER PROGRAMMING FORMULATION

5.1. Introduction

Basically the redundancy allocation problem is an Integer programming problem where the allocations are restricted to integer values only. From previous chapters it is clear that the objective function to be optimised is non-linear and the constraints specified may be linear or non-linear. The problem therefore is that of Non-linear programming where the objective function and constraints are expressible in separable form. Many investigators [26, 27, 28, 31] have used different formulations using either some approximations or elaborate and complex formulations. However, in the opinion of the author there does not seem to be a convincingly straight and simpler approach to the problem of redundancy allocation. Where, there are exact or accurate formulations, the complexity of the problem in computations increses tremendously. For example, the 5-stage problem of [27] required 27 constraints specified for formulation, thereby handling of such a large system for a small problem is definitely not economical. Moreover, the system size that can be handled might be one of the shortcomings.

For the sake of completeness a brief review of all the approaches is being given here.

5.2. Mizukami Formulation [26]

The formulation suggested by Mizukami requires the

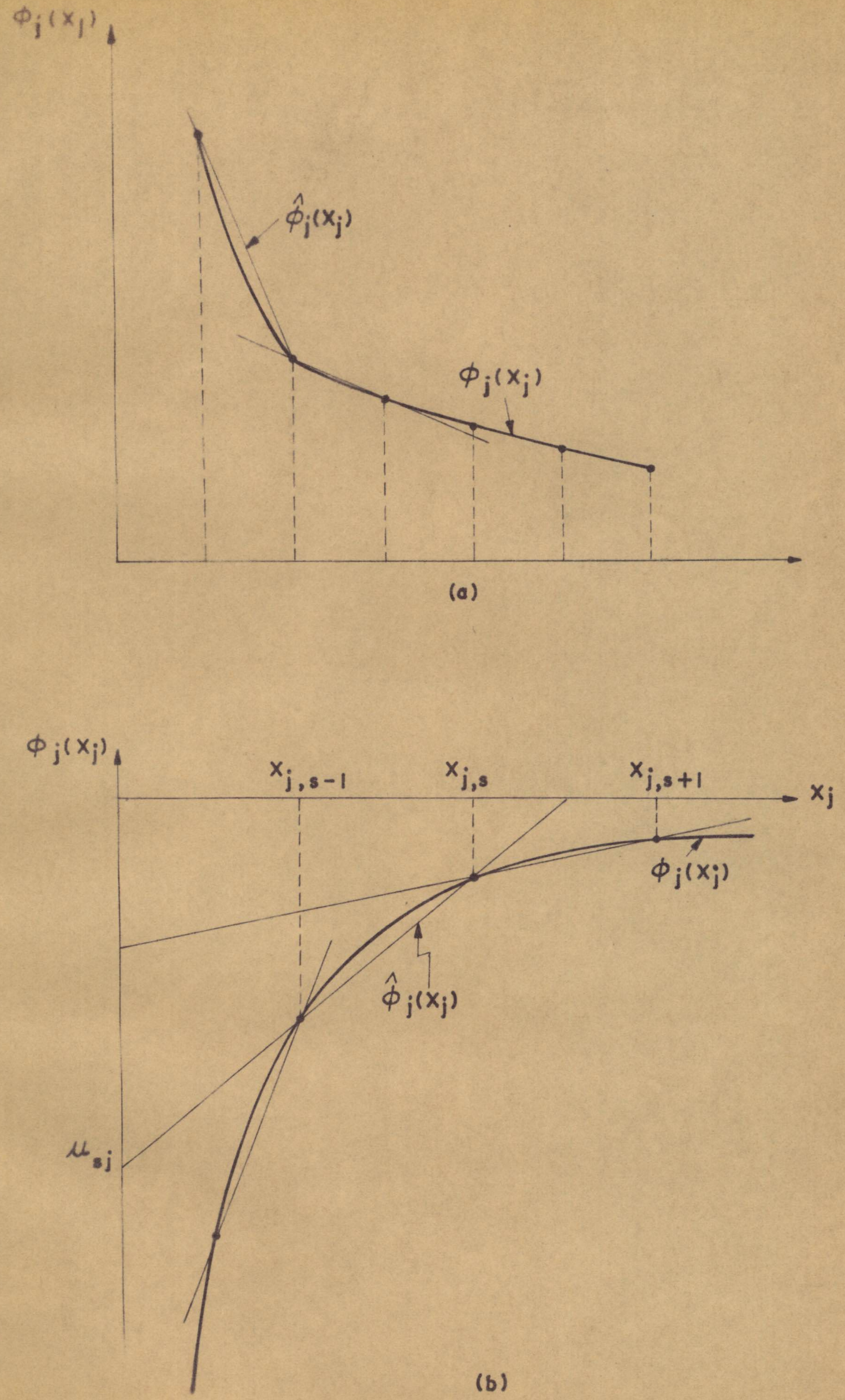


FIG. 5.1 APPROXIMATING CONCAVE RELIABILITY FUNCTION $\phi_j(x_j)$

objective function to be replaced by the approximate straight lines between any two values of x_j as is shown in Fig. 5.1(b). The objective function to be optimised is actually the reliability of the system as shown by $\phi_j(x_j)$ curve in Fig. 5.1.(a). It is found convenient to maximise the log of reliability function as shown in Fig. 5.1(b). Maximisation with a separable convex function is equivalent to that of maximisation with separable concave function of Fig. 5.1(b). Therefore the problem can be stated as follows:

Maximise

$$\hat{z} = \sum_{j=1}^{k} \hat{\phi}_j(x_j) \tag{5.1}$$

subject to the constraints

$$\hat{\phi}_j(x_j) \leqslant \lambda_{sj} x_j + \mu_{sj}$$

$$\sum_{j=1}^{k} a_{ij}(1+x_j) \leqslant b_i$$

$x_j \geqslant 0$ and all x_j to be integers

$s = 1, 2, \ldots n; \; i = 1, 2, \ldots m;$

$j = 1, 2, \ldots k$

where $\hat{\phi}_j(x_j)$ is the linear function approximation to the concave function $\phi_j(x_j)$ in the section lying between $x_{s,\, j-1}$ and $x_{s,j}$ and such n sections have been chosen. All x_j are required to integers, x_j being the redundancies at the stage j and k is the total number of stages. λ_{sj} and μ_{sj} are defined as

$$\lambda_{sj} \equiv \left[\phi_j(x_{j,s}) - \phi_j(x_{j,\, s-1})\right] / (x_{j,s} - x_{j,s-1})$$

$$\mu_{sj} \equiv \left[x_{j,s}\, \phi_j(x_{j,s-1}) - x_{j,s-1}\phi_j(x_{j,s})\right] / (x_{j,s} - x_{j,s-1}) \tag{5.2}$$

The problem (5.1) is solved by means of simplex method as it is purely a linear programming problem if the integer condition of x_j is removed. If the solution happens to be an integer one then the original problem is solved; otherwise, Gomory's [29] algorithm can be applied to obtain the solution. Infact, Mizukami [26] used the Mixed-Integer linear programming technique (also due to Gomory) [19] as the variables $\hat{\phi}_j(x_j)$ and $\hat{z}$ are allowed to be continuous whereas the x_j's are restricted to be integers for all $j = 1, 2, \ldots k$. The results obtained by Mizukami [26] are certainly satisfactory but an error due to large computation and approximation of actual function by broken lines, might certainly cause some discomfort.

The author is afraid to comment how far such an approach will be useful for large systems where there are more number of stages involved.

5.3. Tillman Formulation [27]

Tillman, however, used a different formulation for the integer programming formulation. His approach can be applied to reliability problem without much difficulty and it is all the more useful for problems where the constraints are non-linear also. Both versions are available for the optimisation technique, i.e. maximising reliability of a system subject to some given constraints and the other one is minimisation of cost with a specified index of reliability. Thus, any type of configuration, i.e. parallel or series is allowed while considering the the optimisation problem. The only requirement seems to be that the objective function and the constraint functions should necessarily be of separable form and need not satisfy any convexity or concavity conditions.

The general optimisation problem can be formally stated as:

Optimise

$$z = \sum_{j=1}^{k} \phi_j(x_j)$$

subject to the condition

$$\sum_{j=1}^{k} g_{ij}(x_j) \leqslant b_i \qquad i=1,..m \qquad (5.3)$$

$$\prod_{j=1}^{k} R_j \geqslant M$$

and $x_j = 0, 1, ...x_j'$ $\qquad j = 1, 2, ...k$

where $\phi_j(x_j)$ is any objective function at stage j as a function of x_j, the number of redundant units;

$g_{ij}(x_j)$ are the constraint functions linear or non-linear in x_j;

b_i the amount of i^{th} resource available;

M being the minimum level of reliability acceptable;

R_j is the stage reliability given by

$$\left[1-q_j^{x_j+1} \right]$$

where q_j is unreliability of an element of j^{th} type

x_j' is the maximum number of units allowed at stage j.

Tillman [27] finally converts the above problem (5.3) to a problem of integer programming problem where the variables x_{jn}

represent the n^{th} redundancy at stage j where x_{jn} = .

$n \leqslant x_j$ and $x_{jn} = 0$ for $x_j < n \leqslant x_j'$.

The problem therefore can be written as

Optimise

$$z = \sum_{j=1}^{k} \sum_{n=0}^{x_j'} \Delta \phi_{jn} x_{jn}$$

subject to (5.4)

$$\sum_{j=1}^{k} \sum_{n=0}^{x_j'} \Delta g_{ijn} x_{jn} \leqslant b_i \quad i=1,\ 2,\ ..m$$

also

$$\sum_{j=1}^{k} \sum_{n=0}^{x_j'} \Delta \ln R_{jn}\ x_{jn} \geqslant \ln M$$

and $x_{jn}=1$ for n=0 (i.e. there should be at least one unit in each stage)

$$x_j - x_{j,\ -1} \leqslant 0 \qquad n=1,\ 2, \ldots x_j' \quad j=1,\ 2, \ldots k$$

$$x_{jn} \geqslant 0 \qquad \text{for all j and n}$$

where $x_j = \sum_{n=1}^{x_j'} x_{jn}$, is the number of redundant units at stage j;

$\Delta \phi_{jn} = \phi_{jn}$ for n = 0

$= \phi_{jn} - \phi_{j,n-1}$ for $n=1, \ldots x_j'$

- is the change in objective function when n^{th} redundancy is added.

Similarly,

$$\triangle g_{ijn} = g_{ijn} \quad \text{for } n=0$$

$$= g_{ijn}-g_{ij,\ n-1} \quad \text{for } n=1,\ldots x_j$$

also,

$$\triangle \ln Rj_n = \ln R_{jn} \quad \text{for } n=0$$

$$= \ln R_{jn}-\ln R_{jn,n-1}$$

represents the change in reliability due to the addition of n^{th} redundancy.

$$x_{jn} - x_{j,n-1} \leqslant 0 \quad \text{for } n=1,\ldots x_j'$$

ensures that at each stage j, the n^{th} redundant unit x_{jn} equals one if it is in the solution and that it is in the solution only if the (n-1) unit is included.

The formulation is of course elaborate but easy and straightforward.

For example the problem of page 102 originally from Mizukami [26] paper can be formulated in a tabular form as given in Table 5.1. In this table, the Group I, II & III equalities and inequalities represent the constraints specified. Group I represent that there should be at least one unt in each stage. Group II represent the constraints $x_{jn}-x_{j,n-1} \leqslant 0$ and finally Group III represent the three constraints on cost, weight and volume, respectively. Finally, $\triangle \ln R_{jn}$ are the coefficients of the linear separable function of x_{jn} for $j=1,\ldots k$ and $n=1,\ldots x_j'$ x_j' being 3 here. The obvious solution to the above problem using integer programming technique would be as follows:

$$m_{10} = 1\ ,\quad m_{11} = 1\ ,\quad m_{12} = 1\ ,\ m_{20} = 1\ ,\quad m_{21} = 1\ ,$$

	Stage 1				Stage 2				Stage 3				
	x_{10}	x_{11}	x_{12}	x_{13}	x_{20}	x_{21}	x_{22}	x_{23}	x_{30}	x_{31}	x_{32}	x_{33}	
	1												= 1
Group I					1								= 1
									1				= 1
	-1	1											$\leqslant 0$
		-1	1										$\leqslant 0$
			-1	1									$\leqslant 0$
					-1	1							$\leqslant 0$
Group II						-1	1						$\leqslant 0$
							-1	1					$\leqslant 0$
									-1	1			$\leqslant 0$
										-1	1		$\leqslant 0$
											-1	1	$\leqslant 0$
		4	4	4		8	8	8		6	6	6	$\leqslant 32$
Group III		6	6	6		6	6	6		10	10	10	$\leqslant 30$
		10	10	10		5	5	5		10	10	10	$\leqslant 40$
$\Delta \ln R_{jn}$	-0.15083	0.13103	0.01709	0.00231	-0.09431	0.08526	0.00834	0.00061	-0.04083	0.03922	0.00151	0.00010	

Table 5.1 - Integer Programming Formulation

$$m_{30} = 1 \,,\quad m_{31} = 1$$

signifying that the redundancy allocation to each stage would be (2, 1, 1) respectively.

Here, it can be seen that even for a small problem like this the number of constraints specified totally comes to 15 besides the objective function consisting of 12 variables in all.

However, the advantage of this formulation lies in the fact that different kinds of optimisation problems can be handled with ease. TillmanIs another paper [28] justifies this statement. There it has been possible to take different types of modes of failure also into account due to the fact that objective function can be any arbitrary function regardless of convexity or concavity requirement. The same can be stated about the constraint functions. Non-linear constraints not satisfying these requirements could be handled. Infact, Barlow and Hunter's paper [6] can be considered as a special case of optimisation formulation as given by Tillman [28]. Minimisation of cost in case of series type system is also a special case.

5.4. Future Approaches

There are some other approaches also applicable to the problem of redundancy applications. The method proposed by Lawler [34] and also by Lawler and Wood [30] are worth investigating. The author infact feels that the method of Lawler [34] as suggested for discrete optimisation technique is quite promising. A simple formulation even if it requires somewhat lengthy and time-consuming computations is welcome than a complex formulation with not much time advantage.

5.4.1. Lawler's Approach [34]

Lawler [34],infact, describes a simple,easily programmed method for solving discrete optimisation problems with monotone objective functions and arbitrary (possible non-convex) constraints. The problem can be stated as

$$\text{minimise} \quad z = g_0(x)$$

subject to

$$\begin{aligned} &g_{11}(x) - g_{12}(x) \geqslant 0 \\ &g_{21}(x) - g_{22}(x) \geqslant 0 \\ &\vdots \\ &g_{m1}(x) - g_{m2}(x) \geqslant 0 \end{aligned} \qquad (5.5)$$

where $x = (x_1 , x_2 \ldots x_n)$

and $x_j = 0$ or 1 $(j=1, 2\ldots n)$

with the restrictions that each of the functions g_0 , $g_{11} \ldots g_{m2}$ is monotone non-decreasing in each of the variables $(x_1 , x_2 \ldots x_n)$. Here it is possible to transform non-negative integers into binary variables also and if necessary an arbitrary objective function of the form

$$\text{minimise } g_0(x)$$

can be replaced by a monotone non-increasing objective function by the formulation as

$$\text{minimise} \quad \bar{z}$$

subject to

$$\bar{z} - g_0(x) \geqslant 0$$

An arbitrary inequality constraint of the form $g_i'(x) \geqslant 0$ can be replaced by an inequality constraint $g_i(x) \geqslant 0$ involving

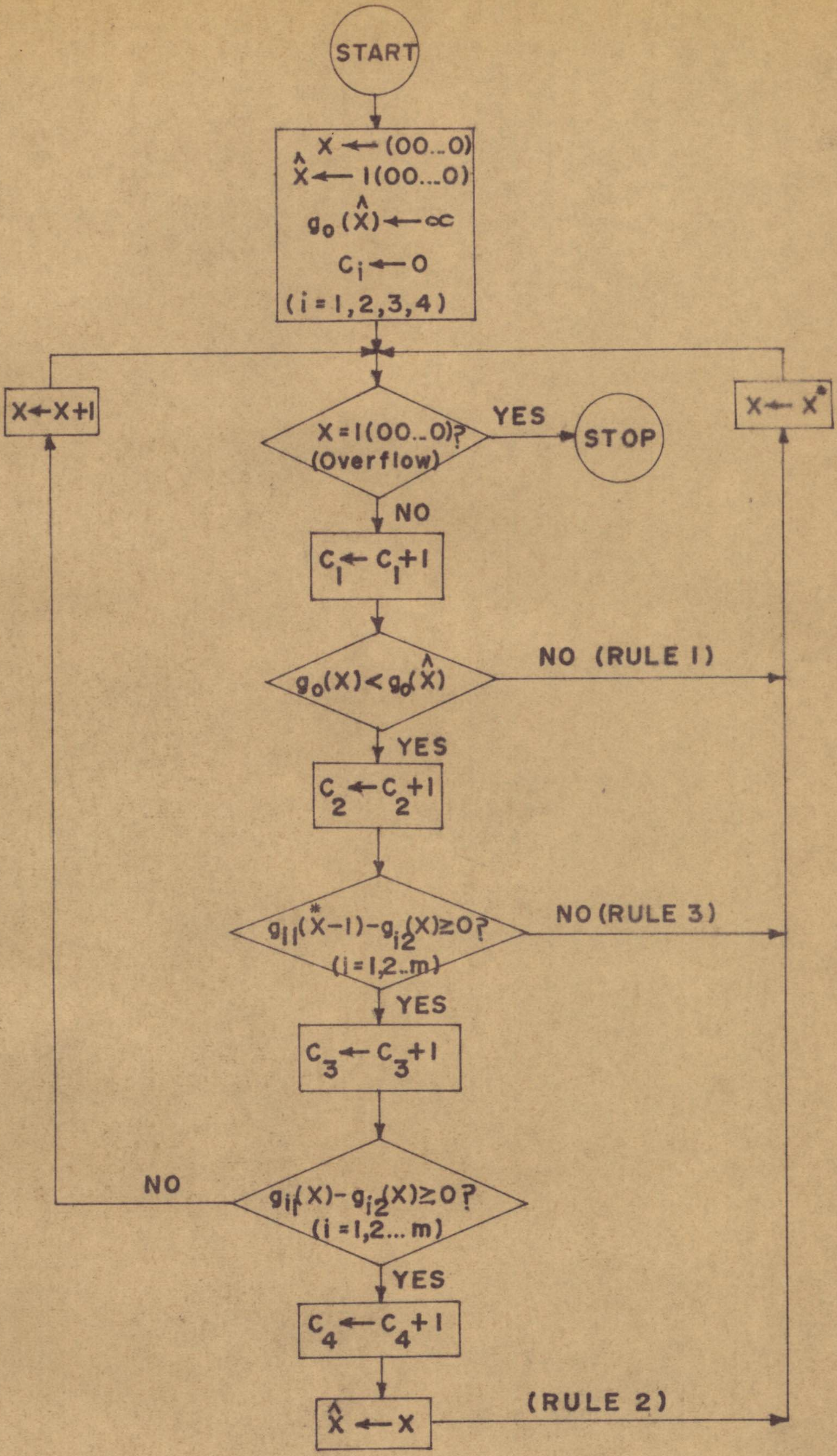


FIG. 5·2 FLOW CHART OF COMPUTATION.

a polynomial of degree 2^{n-1}. This polynomial can be separated into two monotone parts, $g_{i1}(x)$ and $g_{i2}(x)$ of (5.5). Thus any problem can be transformed to a type of (5.5), provided its variables can be made to assume a finite number of discrete values.

The algorithm begins a feasible vector $x = (0, 0, 0..0)$ and examines 2^n possible solution vectors in numerical order. But the labour of examination is cut down considerably by following some rules. As the examination proceeds one can keep the least costly uptodate solution. If $\hat{x}$ be this solution having cost as $g_o(\hat{x})$ and let x be the vector currently being examined then the following rules indicate the conditions under which certain vectors may be skipped over. Let x* be the first vector following x in the numerical order, that has the property that

$x \nleq x^*$,

then

Rule 1: If $g_o(x) \geq g_o(\hat{x})$, skip to x*

Rule 2: If x is a feasible solution, i.e. $g_{i1}(x)-g_{i2}(x) \geq 0$ for all i=1,...m then skip to x*.

Rule 3: If for any i, (i=1, 2...m), $g_{i1}(x^*-1)-g_{i2}(x) \ngeq 0$, skip to x*.

All the steps in the search method discussed above are indicated in Flowchart of computation of Fig. 5.2.

The actual formulation of reliability problem along this method is possible and is being explored by the author. This has the same advantage as the method of search discussed in earlier section. Once the formulation of the problem is done the computation is much easier to program and may be applied conveniently. Only drawback might be that the problem of large number of stages could not be solved possibly by this method. However, this certainly would be the case with others also.

5.4.2. Branch and Bound Method

Branch and Bound method has found application to reliability problem and one such application appears in the report by L. J. Jacobson submitted to the University of California. Jacobson [31] utilised this method to the problem of minimising cost of a system subject to the constraint on reliability, i.e. $\geqslant$ an index of reliability of the system using variables of zero and one type. The algorithm suggests (although not tried) the use of simplex technique successively each time branching takes place. The author is currently working on this problem and hopes to provide some useful conclusions in future only.

CHAPTER 6

CONCLUSIONS

Designing reliability into electronic circuits is often required when these are to perform some important functions and where their mal-functioning may cause heavy loss of money and time. The same applies to the electronic relay circuits where, for the successful operation and maintenance of the system they protect, their reliability is of prime importance.

Proper reliability evaluation of such circuits when they are designed will help to provide more satisfactory operation than otherwise. As long as the components they employ cannot be made 100 percent reliable, recourse has to be taken to having some redundant arrangements. The present thesis, therefore, had the aim of providing general techniques for systematic application of redundancies to these circuits.

Prior to any reliability studies modelling of the system under consideration is often needed. With the reliability parameters available for each of the constituent elements, the system reliability can be evaluated once the modelling of the system is completed. The model of the system will fall in either of the types discussed in Chapter 1, i.e. series, parallel or non series-parallel (planar or non-planar). With the usual existing methods actual evaluation of overall reliability parameter or polynomial of the system is quite tedious. The flow-graph method and thereby the method of inspection described herein is easier to solve such problems directly without much mathematical manipulations. For large and complex systems specially non-series networks the approach discussed is found to be straightforward and easier than that by the method of Factoring Theorem.

The correctness of the results can be checked and necessary changes, if required, could be made at any stage.

The method given in this thesis for the analysis of networks whose elements can short or open (such as diodes etc.) is also easier to apply and can be directly used for any network configuration, series-parallel or non series-parallel as well.

Using the same modelling, the method can be extended to the systems where the elements have more than one mode of failure. Further application is to the analysis of selective and non-selective operation of relays with any configuration.

The algorithm described in section 1.9 is quite convenient for use on computer for large networks of any type mentioned above. This is simple and requires minimum effort on the part of the user in data-preparation.

One can obtain high reliability figure for a system by providing as many redundancies as possible but to ensure that this does not become a very costly, heavy or bulky system the question of optimisation of system reliability with respect to cost, weight or volume arises. The other problem (usually for series type redundancies) is to minimise the cost of a system maintaining a pre-assigned index for the system reliability. Therefore a study was undertaken to comparatively assess the usefulness of the methods used in optimisation problem.

The other special requirement of the redundancy allocation problem is that the number of redundancies at any stage can only be integers. This makes the problem all the more difficult. The usual methods of optimisation, assuming continuous variables cannot be directly used. Since treating the variable continnuous and arriving at an optimum value may not be a true optimum if the final result obtained is rounded off. One can still use the methods devised for continuous variable but the results thus obtained will only be approximate ones.

The methods discussed in section 2.3 are all approximate ones, however to arrive at a reasonably good result without much manipulation is preferred than otherwise. In most of the cases one may get a true optimum allocation by the use of the method of 2.3. This is due to the reason that our objective function is not a bad-behaved function. The choice of for multiple cost problem usually causes some difficulty initially. The techniques given in that section if used however make it easier to assess and arrive at an answer quickly.

The method of section 2.4, is indeed a search method where all the combinations of stage redundancies within the feasible solution are made. A dominating sequence with tolerance specified on unreliability and cost or weight (for multiple cost problem) provides much of the information needed in selecting a proper allocation and its alternatives if desired. The computation is usually more as all possible combinations are tried. This method can be called as direct method of solution. The number of combinations to be tried can be reduced if initially the minimum redundancies at each stage are determined as described therein. The maximum error in the unreliability of the whole system is less than its square if the approximation stated in 2.4 is used for each entry.

Dynamic programming formulation of 2.5 provides an alternative search method of section 2.4. The search is made systematic starting with first stage to the last stage for different values of the resources available. The method is definitely an easier one but requires extensive calculation procedure. The original formulation suggested by Bellman [16] took almost four times than that as described in 2.5 of this thesis. Instead of using reliability of the system as objective function, the log of reliability is used which happens to be a concave function and maximising the system reliability is same as maximising the log function. Dynamic programming method becomes difficult to use when the constraints are more than one. One has again to try several values of lagrangian multiplier before arriving at a correct value using extrapolation or intrapolation. This may be stated here that the formulation used in this thesis is more helpful in this respect than the other as has been mentioned in 2.5. If the lagrangian multipliers are not used, the computations become formidably huge to make this method less appealing. Specially with large number of stages, this may be prohibitory.

An algorithm presented in section 2.6 using the general conditions of optimality has been presented for constrained optimisation problems. The algorithm makes the assumption that the number of constraints is less than the number of stages. The algorithm described is computationally feasible.

Variational method of 3.2 is easier to use but this is available only for single constraint problems. However an algorithm has been devised in the thesis for use with multiple-cost constraints. This method infact is computationally feasible and less laborious but offers an approximate solution to the problem. One can of course 'fill in' the slacks left by rounding off the answer obtained by this method by a systematic trial-and-error method. The method also requires a pre-assigned value of reliability of the system for which minimum cost could be found. In absence of any direct and less laborious methods, this method offers results quickly, however approximate. Although for all the problems tried by this method, the result always comes out as true optimum.

Discrete maximum principle although versatile offers once again an approximate solution to the problem. However, one can use either linear or non-linear separable constraints with this method. The programming of such a method is usually difficult and requires more labour in formulation. The programming and the formulation would be tremendous for large number of constraints specified and infact the programming cannot be generalised for the type and number of constraints used.

The direct search method of section 2.4 can also be applied to the problem of maximising reliability subject to non-linear constraints. Non-linear constraints arise due to the extra interconnecting equipment or auxiliaries required when the number of redundancies increases. The above method is easier to apply and subject to limitations as mentioned for the method of 2.4. However, tolerance in cost, weight etc. has to be specified clearly due to non-linearity of the constraints, if the length of dominating sequence is to be restricted.

Integer programming formulation provides an appropriate answer to the allocation problem but the system becomes computationally large even for small number of stages as has been pointed out in Chapter 5. Moreover, the computational algorithm (Gomory's) for all integer or for the mixed integer-continuous variable case converge in finite steps, however the experience with these algorithms has been rather disappointing. The number of iterations required may be huge even for modest size of the problem. It cannot be claimed at the present time that efficient numerical techniques are available for solving integer programming problems. However the most interesting integer programming problems are those for which the integer variable can be either zero or one.

Formulation of one Non-linear problem into that of 'zero' or 'one' type of integer programming problem, would again increase the size of the system of equations to be handled as prohibitorily large enough to be used on small computations for a few stage problems.

Branch and bound methods are in the wake of development and although academically they may be promising, computationally how far they might be efficient can only be judged after they have been tried on a variety of problems. The author infact is working on the branch and bound method of solution and hopes to give certain definite conclusions in future only.

Therefore in conclusion one can only say that a method which requires minimum effort and time on computer or otherwise, to arrive at a solution (however approximate) is welcome than in comparison to the method requiring elaborate formulation and tedious computation. Making complex problems simple is well received than making simple ones complex.

APPENDIX A

Theorem: If log R(n) is concave, each redundancy allocation generated by the procedure of 2.3 is undominated.

Proof: Let i_o be the index of the last component type added in arriving at $\bar{n}^*$ by the procedure of 2.3 and that

$$= \frac{1}{\sum_{j=1}^{r} a_{ij} c_{i_o j}} \left[\log R_{i_o}(n^*_{i_o}) - \log R_{i_o}(n^*_{i_o} - 1) \right]$$

Let $\bar{n}$ be any other allocation such that $R(\bar{n}) > R(n^*)$. Designating the set of indices for which $n_i > n_i^*$ by I_1 and the set of indices for which $n_i < n_i^*$ by I_2. Then

$$0 < \log R(\bar{n}) - \log R(\bar{n}^*) = \sum_{i \in I_1} \left[\log R_i(n_i) - \log R_i(n_i^*) \right]$$

$$- \sum_{i \in I_2} \left[\log R_i(n_i^*) - \log R_i(n_i) \right]$$

$$= \sum_{i \in I_1} \sum_{h=1}^{n_i - n_i^*} \left[\log R_i(n_i^* + h) - \log R_i(n_i^* + h - 1) \right] - \sum_{i \in I_2} \sum_{h=0}^{n_i^* - n_i - 1}$$

$$\left[\log R_i(n_i^* - h) - \log R_i(n_i^* - h - 1) \right]$$

$$\leq \sum_{i \in I_1} (n_i - n_i^*) \left[\log R_i(n_i^* + 1) - \log R_i(n_i^*) \right] - \sum_{i \in I_2} (n_i^* - n_i)$$

$$\left[\log R_i(n_i^*) - \log R_i(n_i^* - 1) \right] \qquad (1)$$

by concavity of each $\log R_i(n)$ but (1) does not exceed

$$\sum_{i \in I_1} (n_i - n^*_i) \lambda \sum_{j=1}^{r} a_j c_{ij} - \sum_{i \in I_2} (n_i^* - n_i) \lambda \sum_{j=1}^{r} a_j c_{ij}$$

since in procedure 2.3 increments in long reliability are decreasing. As $\lambda > 0$

$$0 < \sum_{j=1}^{r} a_j \sum_{i=1}^{k} c_{ij} n_i - \sum_{j=1}^{r} a_j \sum_{i=1}^{k} c_{ij} n_i^*$$

It is obvious that for some index j

$$\sum_{i=1}^{k} c_{ij} n_i > \sum_{i=1}^{k} c_{ij} n_i^*$$

Similarly, assuming $R(\bar{n}) = R(\bar{n}^*)$, it can be proved that either

$$\sum_{i=1}^{k} c_{ij} n_i > \sum_{i=1}^{k} c_{ij} n_i^* \text{ for some j or } \sum_{i=1}^{k} c_{ij} n_i = \sum_{i=1}^{k} c_{ij} n_i^* \text{ for}$$

all j. Thus $\bar{n}^*$ is undominated.

APPENDIX B

```
C  C  K.B.MISRA.  UNDOMINATED ALLOCATIONS SINGLE COST PROBLEM
      DIMENSION R(10),Q(10),C(10),F(10),M1(10),M2(10)
C     N IS NO OF STAGES AND  RG IS GIVEN RELIABILITY
      READ1,N,RG
    1 FORMAT(I3,F10.5)
C     R(I)AND C(I) ARE STAGE RELIABIUTIES AND COSTS RESPTLY
      READ2,(R(I),C(I),I=1,N)
    2 FORMAT(8F8.5)
      PUNCH2,(R(I),C(I),I=1,N)
      DO 3 I=1,N
      Q(I)=1.-R(I)
C     M1(I) IS ALLOCATION
      M1(I)=1
    3 M2(I)=2
      K=1
   12 CS=0.
      RS=1.
      DO 4 I=1,N
      AM1=M1(I)
      CS=CS+C(I)*AM1
      GOTO (13,14),K
   13 RS=RS*R(I)
      GOTO 4
   14 QI=Q(I)
      M11=M1(I)
      RP=(1.-QI**M11)
      RS=RS*RP
    4 CONTINUE
C     RS,CS ARE SYSTEM RELIABILITY AND COST
      PUNCH5,RS,CS
    5 FORMAT(2F10.5)
      PUNCH 6,(M1(I),I=1,N)
    6 FORMAT(8I5)
      IF(RS-RG)10,11,11
   10 DO 7 I=1,N
      QI=Q(I)
      M11=M1(I)
      M22=M2(I)
      CI=C(I)
    7 F(I)=(LOGF(1.-QI**M22)-LOGF(1.-QI**M11))/CI
C     F(I) ARE DESIRABILITY FACTORS
      PUNCH2,(F(I),I=1,N)
      X=F(1)
      N1=N-1
      J1=1
      DO 9 J=1,N1
      IF(X-F(J+1))8,8,9
    8 X=F(J+1)
      J1=J+1
    9 CONTINUE
      PUNCH1,J1,F(J1)
      M1(J1)=M1(J1)+1
      M2(J1)=M2(J1)+1
      K=2
      GOTO 12
   11 STOP
      END
```

APPENDIX C

```
C  C  K.B. MISRA UNDOMINATED ALLOCATIONS MULTIPLE FACTOR
      DIMENSION R(10),Q(10),C(10),W(10),M1(10),M2(10),F(10)
C     N NO OF STAGES,CG GIVEN COST,WG GIVEN WEIGHT
      READ1,N,CG,WG
1     FORMAT(I3,2F10.5)
      READ2,(R(I),C(I),W(I),I=1,N)
2     FORMAT(9F8.5)
      PUNCH2,(R(I),C(I),W(I),I=1,N)
      DO3I=1,N
      Q(I)=1.-R(I)
      M1(I)=1
3     M2(I)=2
      K=1
      A1=0.25
      A2=0.75
12    CS=0.
      PUNCH2,A1,A2
      WS=0.
      RS=1.
      DO4I=1,N
      AM1=M1(I)
      CS=CS+C(I)*AM1
      WS=WS+W(I)*AM1
      GOTO(13,14),K
13    RS=RS*R(I)
      GOTO4
14    QI=Q(I)
      M11=M1(I)
      RP=(1.-QI**M11)
      RS=RS*RP
4     CONTINUE
      PUNCH5,RS,CS,WS
5     FORMAT(3F10.5)
      PUNCH6,(M1(I),I=1,N)
6     FORMAT(8I5)
      IF(CS-CG)10,10,11
10    IF(WS-WG)15,15,11
15    DO7 I=1,N
      QI=Q(I)
      M11=M1(I)
      M22=M2(I)
      CI=C(I)
      WI=W(I)
      D=A1*CI+A2*WI
7     F(I)=(LOGF(1.-QI**M22)-LOGF(1.-QI**M11))/D
      PUNCH2,(F(I),I=1,N)
      X=F(1)
      N1=N-1
      J1=1
      DO9 J=1,N1
      IF(X-F(J+1))8,8,9
8     X=F(J+1)
      J1=J+1
9     CONTINUE
      PUNCH1,J1,F(J1)
      M1(J1)=M1(J1)+1
      M2(J1)=M2(J1)+1
      K=2
      GOTO 12
11    A1=A1+0.25
      A2=1.-A1
      DO21I=1,N
      M1(I)=1
   21 M2(I)=2
      K=1
      IF(A1-1.)12,12,20
   20 STOP
      END
```

APPENDIX D

```
C  C  K.B.MISRA BELLMAN DYNAMIC PROGRAMMING METHOD
      DIMENSION R(5),Q(5),C(5),W(5),PM(30),FIJ(30),FP(30),F(5,101)
      DIMENSION M(5,101),X(5)
      READ1,N,CG,WG,ALMDA,DALMD
1     FORMAT(I3,4F10.6)
      DO2I=1,N
      READ30,R(I),C(I),W(I)
30    FORMAT(3F10.6)
      Q(I)=1.-R(I)
2     PUNCH3,R(I),Q(I),C(I),W(I)
3     FORMAT(4F10.6)
      ACG=CG+1.
      NCG=ACG
26    DO27I=1,N
      I1=I-1
      CI=1./C(I)
      PUNCH11,I
11    FORMAT(12H STAGE NO IS,I5)
      AIA=CG*CI
      IA=AIA+1.
      DO4MJ=1,IA
      AMJ=MJ-1
      BR=ALMDA*W(I)*AMJ
      BR=-BR
      PM(MJ)=BR
      XY=1.-Q(I)**MJ
      FIJ(MJ)=LOGF(XY)
4     FP(MJ)=FIJ(MJ)+PM(MJ)
      PUNCH5,(PM(MJ),FIJ(MJ),FP(MJ),MJ=1,IA)
5     FORMAT(3E19.8)
      DO10J=1,NCG
      AJ=J-1
      AIJ=AJ*CI
      IJ=AIJ
      IF(IJ)7,7,8
7     PUNCH12,J,IJ,AIJ
12    FORMAT(11H MJ IS ZERO,2I5,F10.6)
      XLM=0.
      NN=2
8     K=0
      B=-1.E30
19    AK=K
      MJ=K+1
      IF(I-1)9,9,13
9     XXX=FP(MJ)
      GOTO16
13    EPS=CG-AK*C(I)
      IP=EPS+1.
      GO TO(31,32),NN
31    XLM=F(I1,IP)
32    XXX=FP(MJ)+XLM
      NN=1
16    IF(B-XXX)14,15,15
14    B=XXX
      MMK=K
15    IF(K-IJ)17,18,18
17    K=K+1
      GOTO19
18    F(I,J)=B
      M(I,J)=MMK
10    CONTINUE
      PUNCH20,(F(I,J),M(I,J),J=1,NCG)
20    FORMAT(2(E20.8,I4))
27    CONTINUE
C     CALCULATES ALLOCATION
      X(N)=M(N,NCG)
      N1=N-1
      AACG=ACG
      DO21I=1,N1
      N2=N-I
      N3=N2+1
      AACG=AACG-C(N3)*X(N3)
      NACG=AACG
21    X(N2)=M(N2,NACG)
      WS=0.
      DO22I=1,N
22    WS=WS+W(I)*X(I)
C     COMPARES SYSTEM WEIGHT
      IF(WS-WG)23,24,25
23    ALMDA=ALMDA-DALMD
      GOTO26
25    ALMDA=ALMDA+DALMD
      GOTO26
24    PUNCH3,(X(I),I=1,N)
      STOP
      END
```

Appendix E

Kuhn-Tucker Condition of Optimality

Neglecting the integer requirement of the variables x_j if the following general Non-linear programming problem is considered

$$g_i\ (x_1..x_n) \leqslant b_i \quad i=1,..u$$
$$g_i\ (x_1..x_n) \geqslant b_i \quad i=u+1,..v \qquad (1)$$
$$g_i\ (x_1..x_n) = b_i \quad i= v...m$$

$$x \geqslant 0; \quad \max z = f(x_1..x_n)$$

where g_i's are the constraint function and f is objective function to be optimised, then we can deduce a condition of optimality including lagrangian multipler by the use of Kuhn-Tucker Theorem. If it is assumed that f and g_i's alongwith their first derivatives are continuous in the entire non-negative orthant, then one can write after adding positive surplus and slack variables, (1) as

$$g_i(\bar{x}) + x_{si} = b_i \quad i=1,..u$$
$$g_i(\bar{x}) - x_{si} = b_i \quad i=u+1, ...v$$
$$g_i(\bar{x}) \qquad = b_i \quad i=v+1,...m \qquad (2)$$

$$\left[\bar{x}\ ,\ \bar{X}_s\right] \geqslant 0\ ;\ \max z = f(\bar{x})$$

Assuming that $f(\bar{x})$ takes absolute maximum at $\bar{x}^*$ and the rank of $[G]$ is equal to rank of $[G_f]$ at $\bar{x}^*$ where $[G]$ and $[G_f]$ are defined as

$$[G] \equiv \|\partial g_i/\partial x_j\| \text{ and } [G_f] \equiv \begin{bmatrix} \partial g_1/\partial x_1 \cdots \partial g_1/\partial x_n \\ \partial g_m/\partial x_1 \cdots \partial g_m/\partial x_n \\ \partial f/\partial x_1 \ \cdots \ \partial f/\partial x_n \end{bmatrix} \qquad (3)$$

and [G] contains columns for positive x_j^* , x_{si}^* so that one can form lagrangian function such that $\lambda_o^* = 1$ [19]. Let J be the subset of the indices j, j=1,..n containing the indices j for which $x_j^* > 0$ and $\hat{J}$ be the subset containing the indices j for which $x_j^* = 0$. Similarly I be the subject of the indices i, i=1,..v containing the indices i for which constraint i is active at $\bar{X}^*$ and $\hat{I}$ be the subset containing i for which constraint i is inactive then it can be proved [19] that there exists a set of m Lagrange multipliers λ_j which are unique if r[G] is rank of [G]=m at $\bar{x}_j^*$ and not unique otherwise, such that

$$\frac{\partial f(\bar{x}^*)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i^* \frac{\partial g_i(\bar{x}^*)}{\partial x_j} = 0 \quad j \in J, \ \lambda_i^* = 0, \ i \in \hat{I} \qquad (4)$$

About the sign of

$$\frac{\partial f(\bar{x}^*)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i^* \frac{\partial g_i(\bar{X}^*)}{\partial x_j} \ , \qquad j \in \hat{J} \qquad (5)$$

it can be shown as in [19] that

$$\frac{\partial f(\bar{X}^*)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i^* \frac{\partial g_i(\bar{X}^*)}{\partial x_j} \leqslant 0 \qquad j \in \hat{J}$$

The above results can be put in the form given below.

If $\bar{X}^*$ is the absolute maximum of $f(\bar{x})$, it is necessary that there exist a $\bar{\lambda}^*$ such that

$$\nabla_x F(\bar{x}^*, \bar{\lambda}^*) = \nabla f(\bar{x}^*) - \sum_{i=1}^{m} \lambda_i \nabla g_i(X^*) \leqslant 0 \qquad (6)$$

with strict equality holding for $j \in J$, where $f(\bar{X}, \bar{\lambda})$ is the Lagrangian function

$$F(\bar{x}, \bar{\lambda}) = f(\bar{x}) + \sum_{i=1}^{m} \lambda_i \left[b_i - g_i(\bar{X}) \right] \qquad (7)$$

Also

$$\nabla_x F(\bar{X}^*, \bar{\lambda}^*)\bar{X}^* = \sum_{j=1}^{n} x_j^* \left\{ \frac{\partial f(\bar{X}^*)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i^* \frac{\partial g_i(\bar{X}^*)}{\partial x_j} \right\} = 0 \quad (8)$$

Similalrly, the first u components of

$$\nabla_\lambda F(\bar{X}^*, \bar{\lambda}^*) = (b_1 - g_1(\bar{x}^*), \ldots, b_m - g_m(\bar{X}^*)) \quad (9)$$

are non negative, while components u+1, ...v are non positive and components v+1, ...m vanish. Furthermore

$$\nabla_\lambda F(\bar{X}^*, \bar{\lambda}^*)\lambda^* = \sum_{i=1}^{m} \lambda_i^* \left[b_i - g_i(\bar{X}^*) \right] = 0 \quad (10)$$

If the point $[\bar{X}^*, \bar{\lambda}^*]$ satisfies the necessary condition then the Lagrangian functions have a saddle point at $[\bar{X}^*, \bar{\lambda}^*]$. Also, if $f(\bar{x})$ is concave over the non-negative orthant, while $g_i(\bar{X})$ is convex if $\lambda_i^* > 0$ and $g_i(\bar{X})$ is concave if $\lambda_i^* < 0$, $i=1, \ldots m$, then $f(\bar{x}^*)$ is the absolute maximum of $f(\bar{x})$.

APPENDIX F

```
C  C  K.B.MISRA OPTIMIZATION OF SYSTEM RELIABILITY
C     DISCRETE MAX. PRINCIPLE (LINEAR CONSTS)
      DIMENSION R(10),C(10),U(10),TH(10),Z(30),ER(30),A(10),Y(30)
      DO 51 MM=1,4
      READ10,N,P,ERM
10    FORMAT(I3,2F20.8)
      DO12I=1,N
      READ11,R(I),C(I)
11    FORMAT(2F20.8)
      U(I)=1.-R(I)
12    A(I)=-C(I)/LOGF(U(I))
      PUNCH50,(R(I),C(I),I=1,N)
   50 FORMAT(8F9.4)
      READ11,Z1,Z2
      Z(1)=Z1
      Z(2)=Z2
      Y(1)=Z(1)
      S1=1.
      DO15J=1,2
      E=A(1)*(1.-Y(1))/Y(1)
      PUNCH52,E
      DO14I=2,N
      Y(I)=A(I)/(E+A(I))
14    S1=S1*(1.-Y(I))
      S=P*S1*(1.-Y(1))
      ER(J)=S-E
      PUNCH24,ER(J)
      X=ABSF(ER(J))
      IF(X-ERM)16,16,17
17    Y(1)=Z(2)
15    CONTINUE
      K=2
20    Z(K+1)=(Z(K)-ER(K)*Z(K-1)/ER(K-1))/(1.-ER(K)/ER(K-1))
      Y(1)=Z(K+1)
      PUNCH24,Y(1)
      E=A(1)*(1.-Y(1))/Y(1)
      PUNCH52,E
52    FORMAT(4E20.8)
      S1=1.
      DO18I=2,N
      Y(I)=A(I)/(E+A(I))
18    S1=S1*(1.-Y(I))
      S=P*S1*(1.-Y(1))
      ER(K+1)=S-E
      K1=K+1
      PUNCH24,ER(K1)
      X=ABSF(ER(K+1))
      IF(X-ERM)16,16,19
```

```
19    K=K+1
      PUNCH10,K,X
      GOTO20
   16 PUNCH10,K,X
      DO21I=1,N
      TH(I)=LOGF(Y(I))/LOGF(U(I))
      PUNCH24,TH(I)
      IT=TH(I)
      T=IT
      B=TH(I)-T
      IF(B-0.5)25,22,22
22    TH(I)=T+1.
      GOTO21
25    TH(I)=T
21    CONTINUE
      PUNCH26,(TH(I),I=1,N)
26    FORMAT(8F9.2)
      Q=1.
      D=0.
      DO23I=1,N
      M=TH(I)
      RG=1.-U(I)**M
      Q=Q*RG
23    D=D+C(I)*TH(I)
       PUNCH11,Q,D
      PR=P*Q-D
      PUNCH24,PR
24    FORMAT(F20.8)
   51 CONTINUE
      STOP
      END
```

APPENDIX G

Maximise

$$R_s = \prod_{n=1}^{k} (1-(1-R^n)^{\theta^n}) \tag{1}$$

subject to

$$\sum_{n=1}^{k} g_i^n(\theta^n) \leqslant b_i \quad i=1, 2, \ldots r \tag{2}$$

where

R_s system reliability

k total number of stage

R^n reliability of one element at n^{th} stages

θ^n number of elements at n^{th} stage,
(θ^n-1) is the number of redundant units.

$g_i^n(\theta^n)$ function representing amount of i^{th} resource consumed at n^{th} stage as a function of θ^4

r number of constraints

b_i total amount of i^{th} resource available

Let x_i^n be the i^{th} resource corresponding to the i^{th} constraint which is consumed in first n stages, i=1, ...r.
Then, the performance equations for the k-stage system may be written as

$$x_i^n = x_i^{n-1} + g_i^n(\theta^n) \quad n=1, 2, \ldots k$$

$$\text{and} \quad x_i^o = 0 , \quad x_i^k \leqslant b_i \quad i=1, 2, \ldots r \tag{3}$$

By defining

$$x^{n}_{r+1} = x^{n-1}_{r+1} + \log(1-(1-R^{n})^{\theta^{n}}) \quad n=1,\ 2\ldots k$$

$$x^{o}_{r+1} = 0 \tag{4}$$

the objective function to be optimised is

$$S = \log R_{s} = x^{k}_{r+1} = \sum_{i=1}^{r+1} c_{i} x^{k}_{i} \tag{5}$$

where $c_i = 0$, $i=1,\ 2\ldots r$ and $c_{r+1}=1$

The Hamiltonian and the adjoint variables of the system can be defined as

$$H^{n} = \sum_{i=1}^{r+1} z^{n}_{i}\ x^{n}_{i}$$

$$= \sum_{i=1}^{r} z^{n}_{i} \left\{ x^{n-1}_{i} + g^{n}_{i}(\theta^{n}) \right\} + z^{n}_{r+1} \left\{ x^{n-1}_{r+1} + \log(1-(1-R^{n})^{\theta^{n}}) \right\} \quad n=1,\ 2,\ldots k \tag{6}$$

$$z^{n-1}_{i} = \frac{\partial H^{n}}{\partial x^{n-1}_{i}} = z^{n}_{i} \quad \begin{matrix} n=1,\ 2,\ \ldots k \\ i=1,\ 2,\ r,\ r+1 \end{matrix} \tag{7}$$

$$z^{n}_{r+1} = c_{r+1} = 1 \tag{8}$$

From equation (7) and (8)

$$z^{k}_{r+1} = 1 \quad n = 1,\ 2,\ldots k \tag{9}$$

Assuming that nontrivial and unique Hamiltonian and adjoint variables exist, the condition for local optimality can be given as

$$\frac{\partial H^n}{\partial \theta^n} = 0 = \sum_{i=1}^{r} z_i^n \frac{\partial g_i^n(\theta^n)}{\partial \theta^n} + \frac{-(1-R^n)^{\theta^n} \log(1-R^n)}{1-(1-R^n)^{\theta^n}} \qquad (10)$$

Here θ^n are assumed to be a continuous variable although they are in fact integers.

Now if one of the constraints in (2), say the j^{th} constraint, is active and rest are free, i.e. the end condition corresponding to j^{th} constraint is fixed

$$\text{Then } z_i^k = c_i = 0 \qquad \begin{array}{l} i = 1, 2, \ldots r \\ i \neq j \end{array} \qquad (11)$$

From (7) and (11),

$$z_i^n = 0 \qquad \begin{array}{l} i = 1, 2, \ldots r \\ i \neq j \\ n = 1, 2, \ldots k \end{array}$$

Therefore (10) becomes

$$z_j \frac{\partial g_j^n(\theta^n)}{\partial \theta^n} - \frac{(1-R^n)^{\theta^n} \log(1-R^n)}{1-(1-R^n)^{\theta^n}} = 0 \qquad (12)$$

The procedure for solving the problem involves the following steps:

a. Assuming a value for θ^1 in (12), obtain z_j and therefrom z_j^n, $n = 2, \ldots k$ as $z_j = z_j^n$ due to (7).

b. θ^n for $n = 2, \ldots k$ is calculated from (12) using the values of z_j^n calculated from a.

c. x_i^k, $i = 1, 2, \ldots r$ is computed from (3).

d. One of the conditions will occur:

(i) If $x_i^k < b_i$ for all $i = 1, 2, \ldots r$, then a higher value of θ^1 is assumed and return to step a.

(ii) If $x_j^k > b_j$ and $x_i^k < b_i$ for $i \neq j$, $j=1, 2, \ldots r$ then a smaller value of θ^1 is assumed and return to step a.

(iii) If $x_m^k > b_m$ $m \neq j$ and $x_i^k < b_i$ for $i=1, 2, \ldots j, \ldots r$, $i \neq j$ where j is the active constraint, then go to step e.

(iv) If $x_j^k = b_j$ and $x_j^k < b_i$, for $i = 1, 2, \ldots r$, $i \neq j$, i.e. the j^{th} constraint reaches its limit while none of the other constraints are violated, we have a case for optimal solution.

e. Replace constraint j by constraint m. Accordingly j is replaced by m in (12) and in steps a and b and procedure is repeated from a - d.

APPENDIX H

```
C  C  K.B.MISRA DISCRETE MAX PRINCIPLE NONLINEAR CONSTRAINTS
      DIMENSION XO(10),R(10),Q(10),Z(3),P(10),C(10),W(10),AX(10),M(10)
      COMMON N,XO,Q,K,Z,P,EPSL,AX,C,W
      READ1,N,PG,CG,WG,EPSL,XIN
1     FORMAT(I5,5F15.8)
      READ2,(R(I),P(I),C(I),W(I),I=1,N)
2     FORMAT(8F10.5)
      READ2,(XO(I),I=1,N)
      XOI=XO(1)
      DO3I=1,N
3     Q(I)=1.-R(I)
      K=1
  100 ZP=Q(1)**XOI
      ZL=LOGF(Q(1))
      ZN=ZP*ZL
      ZN2=1.-ZP
      ZD=2.*P(1)*XOI*ZN2
      Z(1)=ZN/ZD
      ZD1=EXPF(XOI*0.25)
      ZD2=C(1)*(1.+ZD1*0.25)*ZN2
      Z(2)=ZN/ZD2
      ZD3=W(1)*(ZD1+XOI*0.25*ZD1)*ZN2
      Z(3)=ZN/ZD3
300   CALL  NEWTON
      X1=0.
      X2=0.
      X3=0.
      AX(1)=XOI
      DO4I=1,N
      X1=X1+P(I)*AX(I)*AX(I)
      X2=X2+C(I)*(AX(I)+EXPF(AX(I)*0.25))
4     X3=X3+W(I)*AX(I)*EXPF(AX(I)*0.25)
      PUNCH50,X1,X2,X3,XOI
50    FORMAT(4E20.8)
      GOTO(11,12,18),K
11    IF(X1-PG)5,6,13
5     IF(X2-CG)7,14,16
7     IF(X3-WG)8,10,17
8     XOI=XOI+XIN
      GOTO100
6     IF(X2-CG)9,9,16
9     IF(X3-WG)10,10,17
10    PUNCH2,(AX(I),I=1,N)
      GOTO200
13    XOI=XOI-XIN
      GOTO100
14    IF(X3-WG)10,10,17
16    K=2
```

```
        GOTO300
 17     K=3
        GOTO300
 12     IF(X1-PG)19,21,23
 19     IF(X2-CG)20,22,13
 20     IF(X3-WG)8,10,17
 21     IF(X2-CG)22,22,13
 22     IF(X3-WG)10,10,17
 23     K=1
        GOTO300
 18     IF(X1-PG)24,26,23
 24     IF(X2-CG)25,27,16
 25     IF(X3-WG)8,10,13
 26     IF(X2-CG)27,27,16
 27     IF(X3-WG)10,10,8
200     DO30I=1,N
        IX=AX(I)
        XXX=IX
        XDF=AX(I)-XXX
        IF(XDF-0.5)31,32,32
 32     M(I)=IX+1
        GOTO30
 31     M(I)=IX
 30     CONTINUE
        RS=1.
        DO33I=1,N
        RP=1.-Q(I)**M(I)
 33     RS=RS*RP
        PUNCH50,RS
        END
        SUBROUTINE  NEWTON
        DIMENSION XO(10),Q(10),Z(3),P(10),AX(10),C(10),W(10)
        COMMON N,XO,Q,K,Z,P,EPSL,AX,C,W
        DO20I=2,N
        X=XO(I)
        A=Q(I)
        AA=LOGF(Q(I))
        GOTO(1,2,3),K
 1      AD=2.*Z(1)*P(I)
        A1=AA/AD
 6      A2=A**X
        A3=X+A1
        F=X-A2*A3
        FD=1.-A2*(1.+A3*AA)
        FM=F/FD
        X1=X
        X=X-FM
        TEST=ABSF(X1-X)
        IF(TEST-EPSL)5,5,6
 5      AX(I)=X
```

```
        GOTO20
 2      BD=Z(2)*C(I)
        B4=AA/BD
 8      B1=EXPF(0.25*X)*0.25
        B2=0.25*B1
        B3=1.+B1
        A2=A**X
        F=B3*(1.-A2)+B4
        FD=B2-A2*(B3*AA+B2)
        FM=F/FD
        X1=X
        X=X-FM
        TEST=ABSF(X1-X)
        IF(TEST-EPSL)7,7,8
 7      AX(I)=X
        GOTO20
 3      CD=Z(3)*W(I)
        C1=AA/CD
 10     C2=1.+0.25*X
        C3=EXPF(0.25*X)
        A2=A**X
        C4=C2*C3
        C5=0.25*C3
        C6=0.25*C4
        C7=C4+C1
        F=C4-A2*C7
        C8=C6+C5
        FD=C8-A2*(C7*AA+C8)
        FM=F/FD
        X1=X
        X=X-FM
        TEST=ABSF(X1-X)
        IF(TEST-EPSL)9,9,10
 9      AX(I)=X
 20     CONTINUE
        PUNCH30,K
        PUNCH40,(AX(I),I=2,N)
 30     FORMAT(I10)
 40     FORMAT(7F10.4)
        RETURN
        END
```

APPENDIX H-1

```
C   C   K B MISRA  LINEAR CONSTRAINTS DISCRETE MAX. PRINCIPLE
        DIMENSION R(5),P(5),C(5),W(5),Q(5),X(5),Z(3),AX(5),M(5),XO(5)
        READ1,N,PG,CG,WG,EPSL,XIN
1       FORMAT(I5,5F13.8)
        READ2,(R(I),P(I),C(I),W(I),I=1,N)
2       FORMAT(8F9.5)
        READ2,XOI,(XO(I),I=2,N)
        DO 3I=1,N
3       Q(I)=1.-R(I)
        K=1
100     ZP=Q(1)**XOI
        AX(1)=XOI
        KEY=2
        TEST = XOI
        ZL=LOGF(Q(1))
        ZN=ZP*ZL
        ZN2=1.-ZP
        ZD=P(1)*ZN2
        Z(1)=ZN/ZD
        ZD2=C(1)*ZN2
        Z(2)=ZN/ZD2
        ZD3=W(1)*ZN2
        Z(3)=ZN/ZD3
300     DO39I=2,N
         PUNCH1,K
        GO TO(35,36,37),K
35      ZPN=Z(1)*P(I)
        GO TO 38
36       ZPN=Z(2)*C(I)
        GO TO 38
37      ZPN=Z(3)*W(I)
38      YX=XO(I)
        FL=LOGF(Q(I))
42      FX=Q(I)**YX
        FXN=FX-ZPN/(ZPN+FL)
        FXDN=FX*FL
        YXD=FXN/FXDN
        IF(ABSF(YXD)-EPSL)40,40,41
41      YX=YX-YXD
        GO TO 42
40      PUNCH2,YX
        NYX=YX+0.55
        AX(I)=NYX
39      CONTINUE
        PUNCH2,(AX(I),I=1,N)
        GO TO(110,120),KEY
110     IF(ABSF(TEST-AX(1)))150,140,150
140     XIN=0.5*XIN
150     KEY=2
```

```
      TEST=AX(1)
      GO TO 130
120    KEY=1
130   X1=0.
      X2=0.
      X3=0.
      DO4I=1,N
      X1=X1+P(I)*AX(I)
      X2=X2+C(I)*AX(I)
4     X3=X3+W(I)*AX(I)
      PUNCH50,X1,X2,X3
50    FORMAT(3E20.8)
      GO TO(11,12,18),K
C     REST OF THE PROGRAM IS SAME AS THAT OF
C     APPENDIX H FROM STATEMENT NO 11 ONWARDS.
```

APPENDIX I

```
C   C   K B.MISRA  SUBROUTINE SIMPLEX
        DIMENSION A(10,20),W(10),L(10)
        COMMON II,JJ,III,A,L,W
        KKK=0
 22     I=1
 23     I=I+1.
        IF(I-III)24,40,40
 24     IF(L(I))23,25,23
 25     DO 27 J=1,JJ
        IF(A(I,J))26,27,26
 26     A(III,J)=A(III,J)-A(I,J)
 27     CONTINUE
        GOTO 23
 40      K=III
 44     J=0
        W(K)=0.
        LK=0
 42     J=J+1
        IF(J-JJ)41,45,45
 41     IF(A(K,J))43,42,42
 43     IF(W(K)-A(K,J))42,42,47
 47     W(K)=A(K,J)
        L(K)=J
        GOTO 42
 45     IF(L(K))46,62,46
 46     KJ=L(K)
        DO 120 I=2,II
        IF(A(I,KJ))120,120,121
 120    CONTINUE
        PUNCH 130
 103    FORMAT(8HFEASIBLE)
        GOTO 70
 121    I=1
        JK=0
 50     I=I+1
        IF(I-II)52,52,56
 52     IF(A(I,JK))50,50,51
 51     X=A(I,JJ)/A(I,KJ)
        IF(JK)55,53,55
 55     IF(X-XMIN)53,50,50
 53     XMIN=X
        JK=1
        GOTO 50
 56     X=A(JK,KJ)
        L(JK)=KJ
        DO 57 I=1,III
 57     W(I)=A(I,KJ)
        IJ=JK-1
```

```
      DO 59 I=1,IJ
      DO 59 J=1,JJ
      IF(A(JK,J))58,59,58
58    IF(W(I))580,59,580
580   A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
59    CONTINUE
      IJ=JK+1
      DO 61 I=IJ,III
      DO 61 J=1,JJ
      IF(A(JK,J))60,61,60
60    IF(W(I))600,61,600
600   A(I,J)=A(I,J)-W(I)*(A(JK,J)/X)
61     CONTINUE
      DO 205 J=1,JJ
205   A(JK,J)=A(JK,J)/X
      KKK=KKK+1
      PUNCH 105,KKK,A(K,JJ),L(JK)
105   FORMAT(1X,I4,6X,F15.2,10X,I4)
      GOTO44
62    IF(K-1)70,70,63
63    IJ=JJ-1
      DO 65 J=1,IJ
      IF(A(K,J)-.0001)65,65,66
65    CONTINUE
      PUNCH 103
130   FORMAT(9HUNBOUNDED)
      PUNCH 101
101   FORMAT(46HITERATION      OBJ. FUNCTION      NEW BASIC VAR.)
      DO 140 J=1,JJ
140   A(III,J)=0.
      K=1
      KKK=0
      GOTO44
66    PUNCH 6
6     FORMAT(10HINFEASIBLE)
70    PUNCH8,A(1,JJ)
8     FORMAT( ///13HOBJ. FUNCTION,F20.8/)
      PUNCH 7
7     FORMAT(23HVARIABLE           VALUE)
      DO 71 I=2,II
71    PUNCH 5,L(I),A(I,JJ)
5     FORMAT(I4,F20.8)
      PUNCH 100
      DO 78 I=1,III
100   FORMAT(////16HTHE FINAL MATRIX)
      PUNCH150,I
150   FORMAT(//35X,4HROW,I2/)
78    PUNCH 4,(A(I,J),J=1,JJ)
      RETURN
      END
```

List of Publications by the Author relevant to the Thesis

1. "Reliability Theory as applied to System Protection," presented at the 37th Annual Research Session of the Central Board of Irrigation & Power, Bhubaneswar (Orissa, India), June 1967.

2. "Analysis of Failure of Components in Electronic and Electrical Circuits," Journal of the Institution of Engineers (India), Vol. 49, No. 8, Pt. EL4, April 1969.

3. "Reliability Analysis of Selective and Non-selective Operations of Relays," Journal of the Institution of Engineers (India), Vol. 50, No. 4, Pt. EL2, Dec. 1969.

4. "Topological Methods for the Analysis of Redundant Networks," Journal of the Institution of Engineers (India), Pt. EL Control Group (In press).

5. "Reliability Analysis of Redundant Networks using Flowgraph," IEEE Trans. on Reliability, Vol. R-19, Feb. 1970.

6. "An Algorithm for Reliability Evaluation of Redundant Networks," communicated to IEEE Trans. on Reliability, Paper No. TR-112.

Being communicated on the basis of work contained in the thesis

1. "Dynamic Programming Formulation of Redundancy Allocation-Problem," to Journal of the Institution of Engineers (India).

2. "An Algorithm for Reliability Optimisation Problem," to IEEE Trans. on Reliability.

Besides the above publications, all the general purpose programs written in FORTRAN-II, are also being communicated to the Proceedings, I.E.E., London, for their Computer Library.

REFERENCES

1. Shanon, C., & Moore, F. S., "Reliable circuits using less reliable relays," Journal of Franklin Institute, Philadelphia, Pa, Sept. and Oct. 1956, pt 1,p.191 and pt 2, p. 281

2. Fred Moskowitz, "The analysis of Redundancy Networks," A.I.E.E. Trans., pt I Communication and Electronics, Nov. 1958, p. 627.

3. Lorens, Charles S., "Flow Graphs," (a book). McGraw-Hill Book Co., New York, 1964, p.113.

4. H. Walter Price, "Reliability of parallel Electronic Components," IRE Trans.. on Reliability and Control, April 1960, pp. 35-39.

5. Von Alven, W. H., "Reliability Engineering," (a book). ARINC Research Corporation, Prentice Hall, Inc., 1964.

6. Barlow, R. E., & Hunter, L. C., "Criteria for determining optimum redundancy," IRE Trans. on Reliability and Control, April 1960, pp. 73-77.

7. Fabrikant, U. L., "Applying Reliability Theory to Apprisal of the Performance of Relay Protection Gear," Electrichestvo No. 9, 1965, p.36.

8. Barlow, Richard E.; Proschan, F., & Hunter, L. C., "Mathematical Theory of Reliability," (a book). John Wiley & Sons, Inc., New York, 1965.

9. Bazovsky, I., "Reliability Theory and Practice," (a book). Prentice Hall, Englewood Cliffs, New Jersey, 1961.

10. Kettelle, J. D. (Jr.), "Least Cost Allocation of Reliability Investment," Operations Research, Vol. 10, No. 2, March-April 1967, pp. 249-265.

11. Hees, R. V., & Meerendonk, H. W., "Optimum Reliability of Parallel Multicomponent System," Operational Research Quarterly (U.K.), Vol. 12, May 1961, pp. 16-26.

12. Moskowitz, F., & McLean, J. B., "Some Reliability Aspects of System Design," IRE Trans. of Reliability and Quality Control, PGRQC-8, Sept. 1956, pp. 7-35.

13. Gordon, K., "Optimum component redundancy for maximum system redundancy," Operations Research, Vol. 5, April 1957, pp. 229-243.

14. Black, G., & Proschan, F., "On optimal Redundancy," Operations Research, Vol. 7, No. 5, Sept.-Oct. 1959.

15. Proschan, F., & Bray, T. H., "Optimum Redundancy under Multiple Constraints," Operations Researwh, Vol. 13, No. 5, Sept. Oct. 1965.

16. Bellman, R.E., & Dreyfus,S. E., "Dynamic Programming and the Reliability of Multicomponent Devices," Operations Research, Vol. 6, No. 2, March April 1958, pp. 200-206.

17. Bellman, R. E., & Dreyfus, S. E., "Applied Dynamic Programming," (a book), Princeton University Press, Princeton, No. 1, 1962.

18. Fyffe, D. E.; Hines, W. W., & Nam Kee Lee, "System Reliability Allocation and a Computational Algorithm," IEEE Trans. on Reliability, Vol. R-17, No. 2, June 1968, pp.64-69.

19. Hadley, G., "Non-linear and Dynamic Programming," (a book). Addison-Wesley Publishing Company, Inc., Reading, Mass., 1964.

20. Fan, L. T., & Wang, C. S., "The Discrete Maximum Principle: A Study of Multistage System optimisation," (a book). John Wiley & Sons, New York, 1964.

21. Fan, L. T.; Wang, C. S.; Tillman, F. A.; & Hwang, C. L., "Optimisation of System Reliability," IEEE Trans. on Reliability, Vol. R-16, No. 2, Sept. 1967, pp. 81-86.

22. Fan, L. T., & Wang, C. S., "On the optimisation of multi-stage feedback processes," J SIAM, Vol. 12, March 1964, pp. 226-232.

23. Tillman, F. A.; Hwang, C. L.; Fan, L. T.; & Balbale, S. A., System Reliability subject to Multiple non-linear constraint," IEEE Trans. on Reliability, Vol. R-17, No. 3, Sept. 1968, pp. 153-157.

24. Misra, K. B., "Monte Carlo methods for Field Problems," J.I.E.(India), Vol. 50, No. 1, Pt. ET1, Sept. 1969, pp. 47-53.

25. Lanczos, C., "Linear Differential Operations," (a book), Van Nostrand, Princeton, N. J., 1961, p. 141.

26. Koichi Mizukami, "Optimum Redundancy for Maximum System Reliability by the Method of Convex and Integer Programming," Operations Research, Vol. 16, March-April 1968, pp. 392-406.

27. Tillman, F. A., & Liittschwager, "Integer Programming Formulation of Constrained Reliability Problems," Management Science, Vol. 13, No. 11, July 1967, pp. 887-899.

28. Tillman, F. A., "Optimisation by Integer Programming of Constrained Reliability Problems with several modes of failure," IEEE Trans. on Reliability, Vol. R-18, No. 2, May 1969, pp. 47-53.

29. Gomory, R., "An Algorithm for Integer Solution to Linear Programming," Princeton IBM Mathematics Research Project, Technical Report No. 1, November 17, 1958.

30. Lawler, E. L., & Wood, D. E., "Branch and Bound Methods: A Survey," Operations Research, Vol. 14, No. 4, July-Aug. 1966, pp. 699-719.

31. Jacobson, L. J., "An Application of the Branch and Bound Method to the Catalogue Ordering Problem," Report submitted to the Operations Research Center, University of California, Berkeley, July 1968.

32. Bodin, L. D., "Optimisation Procedure for the Analysis of Coherent Structures," IEEE Trans. on Reliability, Vol. R-18, No. 3, Aug. 1969, pp. 118-126.

33. Gomory, R. E., "An Algorithm for Integer Splutions to Linear Progams," in "Recent Advances in Mathematical Programming," R. Graves and P. Wolfe, editors (a book), McGraw-Hill, New York, 1963, p.269 & 284.

34. Lawler, E. L., & Bell, M. D., "A Method for solving Discrete Optimisation Problems," Operations Research, Vol. 14, No. 6, Nov.-Dec. 1966, pp. 1098-1112.

35. Ghare, P. M., "Optimal Redundancy for Reliability in Series Systems," Operations Research, Vol. 17, No. 5, Sept.-Oct. 1969, pp. 838-847.

36. Cabot, A. V., & Hurter, A. P., "An Approach to Zero-One integer Programming," Operations Research, Vol. 16, No. 6, Nov.-Dec. 1968, p. 1206.

37. Vajda, S., "Mathematical Programming," (a book), Addison-Wesley, Reading, Mass., 1961.

Mr. K. B. Misra was born on January 23, 1943, in Kota, India. He received his Bachelor of Engineering degree in Electrical Engineering in 1963 from the University of Baroda, Baroda, India, and the Master of Engineering degree in Power System Engineering in 1966 from the University of Roorkee, Roorkee, India.

Since 1966 he is on the staff of the Department of Electrical Engineering, University of Roorkee. He has been teaching various subjects to the undergraduate and post-graduate students. He has also been actively interested in the studies of digital computer applications to power system problems. His current interest is in the field of reliability engineering and the present thesis is an outcome of the research he carried out besides his regular teaching assignments.

Mr. Misra is a member of IEEE and subscribes to power and reliability groups.