MICHAEL K. KONTESSIS

MULTIOBJECTIVE TECHNIQUES FOR OPTIMUM PARALLEL REDUNDANCY WITH SEVERAL COST FUNCTIONS - A TRADE - OF FUNCTION APPROACH

CHAPTER 1

RELIABILITY OF A SYSTEM

1.1 *Introduction and System Reliability Definition*

Modern life is relied on a large number of types of machines and instruments. The term "system" is introduced in order to describe more generally a set of components built in some configuration satisfying one or more tasks. However, a system designed to accept some inputs and to derive some desirable outputs in time is accompanied by the uncertainty of its adequate operation which means that a system failure at a moment in time must be considered quite probable. This is what the experience points out. Many real world examples could be arranged varying between the simple electrical bulb system and the mostly sofisticated spacecraft one.

Therefore, for any reason out of our control, a system inadequate operation may happen at any time and thus problems about our confidence in this system arise. Of course, we would like to measure this confidence. Statistics enables us to do so and it can be done using the concept of the system reliability, that is "the probability of a system performing its purposes adequately for the period of time intended under the operating conditions encoutered" (Radio-Electronics Television Manufactures Association, 1955).

Thus, the reliability of a system is a function of the time and of the conditions of its operation. For example, an old in use light bulb is not as reliable as one that has recently been put into service and, also, a system is more reliable functioning under normal conditions of its environment.

Barlow and Proschan (1965) give a functional formula describing the reliability of a system, as it has been defined above. For, let the period of time intended is t (from 0 to t). The random variable T will represents the life of the system. For $T = t$ we define a state variable

X(t) taking the value 1 if the system functions adequately at the time t and taking the value 0 otherwise. Assuming that adequate operation at time t implies adequate performance during the period of time between 0 and t we can write

$$\text{System Reliability in } [0,t] = \text{Prob}(X(t) = 1) \tag{1.1}$$

Thus, we can summarize at this point that:

a) Reliability is a function of time. For example, an almost worn - out electrical bulb would be not as liable as a new in use one.

b) Reliability is a function of conditions of use or environment. For example our electrical bulb is not going to function very reliably if we hit it with a hammer.

1.2 *Failure Distributions Related to Reliability*

Giving the definition of a system in the foregoing section 1.2 we have describe it as a set of components. For example, a usual computer system could be considered as the set of the components: Card Reader Unit, C.P.U, Line Printer Unit, Disk Drive Unit and Magnetic Tape Unit. Therefore, in calculating the reliability of a system the failure distributions of its particular components are of essential interest. Denoting the time to failure of a component with T (T is a random variable) the failure distributions listed below are very important giving information about the component life.

a) Probability Distribution of the Time to Failure.

$$\text{Prob}(t \leqq T \leqq t + dt) = f(t)dt, \qquad t \geqq 0 \tag{1.2}$$

The function f(t) derives at each time the probability of component failing between the time t and t + dt.

b) Cumulative Probability Distribution of Time to Failure.

$$\text{Prob}(T \leqq t) = F(t) = \int_0^t f(t)dt, \qquad t \geqq 0 \tag{1.3}$$

The function F(t) gives the probability that a component has failed by the time t. The equation 1.3 gives the function F(t) in terms of the function f(t). Conversly, on differentiating 1.3 we find

$$f(t) = F'(t) \tag{1.4}$$

c) Survival function.

$$\text{Prob}(T > t) = S(t) = 1 - F(t) = \int_t^{\infty} f(t)dt\,, \quad t \geqq 0 \qquad (1.5)$$

The function S(t) often arises very naturally, for example in considering the probability that a component will survive a guarantee period. One can easily verify that

$$f(t) = -S'(t) \qquad (1.6)$$

d) Failure Rate.

$$\text{Prob}(t \leqq T \leqq t + dt \;/\; T > t) = r(t)dt = \frac{f(t)}{S(t)}\,dt \qquad (1.7)$$

Roughly speaking r(t) gives the probability of almost immediate failure of a component known to be of age t. The failure rate r(t) is related to the survival function via probability distribution f(t)—formula 1.7—but a direct relation between them can be derived and it is:

$$S(t) = \exp\{-\int_0^t r(t)dt\}, \qquad t \geqq 0 \qquad (1.8)$$

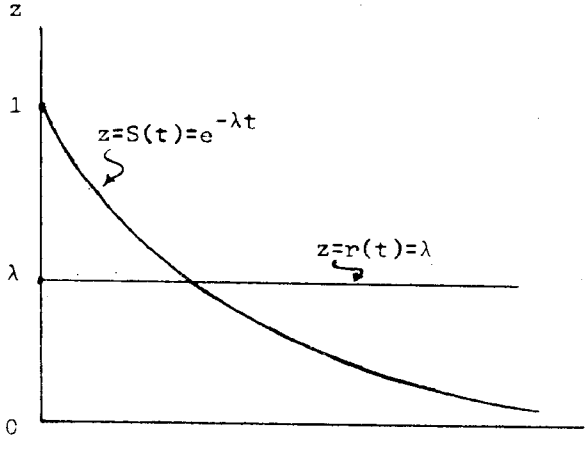
The concept of the failure rate is important, both theoretically and in practice and gives information about the life of a component according to the behaviour of the function r(t). Thus, constant failure rate means that the probability of component failing next moment is constant independently of the age of the component. Increasing r(t) means that the same probability increases as the component becomes older. Finally, decreasing failure rate means that this probability decreases as the component age increases. However, this last case is not often occured in practice.

Figure 1.1 and figure 1.2 illustrate two examples each for constant failure rate and proportional to t one respectively related with their own survival functions.

1.3 *Some of the most common in use Failure Distributions*

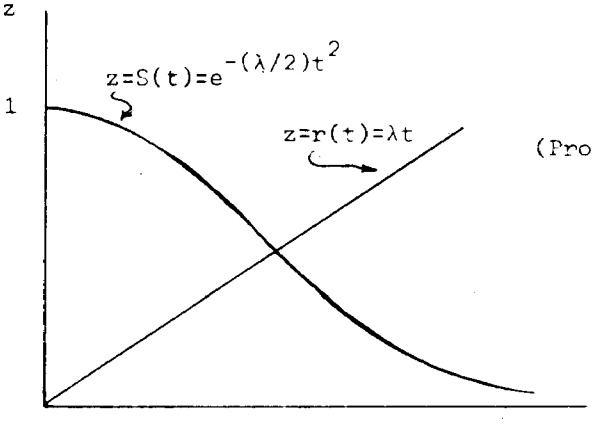
There exists a variety of families of distributions that could be well

used in order to describe the life of a component. The listing below contains those theoretical distributions that are often in use in the practical applications.



(Constant Failure Rate)

Figure 1.1



(Proportional to t Failure Rate)

Figure 1.2

a) Exponential failure distribution.

$$f(t) = \lambda e^{-\lambda t} \quad , \quad \lambda > 0 \quad , \quad t \geqq 0 \tag{1.9}$$

with:

Failure rate $r(t) = \lambda$
mean $\mu = 1/\lambda$
variance $\sigma^2 = 1/\lambda$

Such a failure density appears when the component failures occur randomly and independently. For example, this is the case of the failure occurence of certain types of electronic devices.

b) Gamma failure distribution.

$$f(t) = \lambda(\lambda t)^{\alpha-1}\Gamma^{-1}(\alpha)\exp(-\lambda t) \quad , \quad \alpha,\lambda > 0 \quad , \quad t \geqq 0 \tag{1.10}$$

with:

$$\text{failure rate } r(t) = f(t) \cdot \{1 - \int_0^t f(t)dt\}^{-1} \tag{1.10)'}$$

mean $\mu = \alpha\lambda^{-1}$
variance $\sigma^2 = \lambda^{-1}\sqrt{\alpha}$

When $\alpha = 1$ the Gamma failure density gives the exponential one.

c) Weibull failure distribution.

$$f(t) = \lambda\alpha t^{\alpha-1}\exp(-\lambda t^{\alpha}), \quad \lambda,\alpha > 0 \quad t \geqq 0 \tag{1.11}$$

with:

failure rate $r(t) = \lambda\alpha t^{\alpha-1}$
mean $\mu = \lambda^{-1}\Gamma(1 + 1/\alpha)$

For $\alpha = 1$ we again find the exponential distribution from 1.11. The Weibull density will be a suitable model for a situation in which all components on test fail around the same time.

d) Modified extreme value failure distribution.

$$f(t) = \lambda^{-1}\exp(-\frac{e^t - 1}{\lambda} + t) \quad , \quad \lambda > 0 \, , \quad t \geqq 0 \tag{1.12}$$

with:

failure rate $r(t) = \lambda^{-1}e^t$

e) Truncated normal failure distribution.

$$f(t) = (\alpha\sigma\sqrt{2\pi})^{-1}\exp\{-\frac{(t-\mu)^2}{2\sigma^2}\}, \quad \sigma,\mu > 0, \quad t \geq 0 \quad (1.13)$$

where α is a normalizing constant and the failure rate is calculated using a formula similar to the formula 1.10′.

f) Log-normal failure distribution.

$$f(t) = (t\sigma\sqrt{2\pi})^{-1}\exp\{-(2\sigma^2)^{-1}.(\log t - \mu)^2\}, \quad \sigma > 0, \quad \mu > 0, \quad t \geq 0 \quad (1.14)$$

with failure rate being calculated by the formula 1.10′.

All densities a) to e) possess a monotonic failure rate (either increasing or decreasing) and therefore can be considered suitable for handling failure rate data due to wear and tear. Log-normal, in the other hand, has a non-monotonic failure rate which possesses a maximum value, decreasing then to zero and for this reason it is not a good distribution for representing failure data.

For the purposes of the present work the component reliabilities of a complex system are assumed known and it will be valid everywhere in the subsequent pages.

1.4 *System Reliability Calculation*

Before passing to the discussion of the main subject of this work, which is the reliability improvement, a review of the techniques for reliability calculation of various types of systems in terms of the given component reliabilities is of practical interest.

Whatever the patricular system under consideration it is assumed that its components fail independently subjected to their own reliability laws.

1.4.1 *Series and Parallel Systems*

Consider a system consisting of k components and we assume that the failure of at least one component causes the failure of the entire system. Such a system is called «series - system».

Relating the electrical current flow through a series - electrical-circuit with the reliability flow through the components of a series - system, we can easily see that a similar to the first case network could be drown in order to give a graphical representation of the second case. Thus, for every series - system a reliability network is associated discribing the logic of failure of this system. The figure 1.3 illustrates a k-component series system.

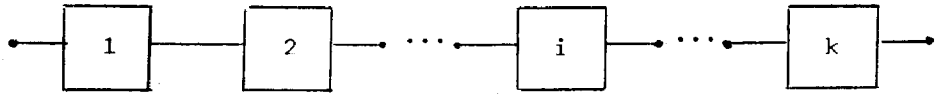


Figure 1.3

Given the reliabilities p_i, $i = 1, 2,..,k$ of the particular components (that is the probabilities of the component survival during a specified period of time, assuming that at $t = 0$ all components are new) we can calculate the over-all system reliability in terms of the component reliabilities using simply the probability multiplication law:

$$R = \prod_{i=1}^{k} p_i = \prod_{i=1}^{k} (1 - q_i) \tag{1.15}$$

where $q_i = 1 - p_i$ are the component unreliabilities.

Let us now consider another situation in which the configuration of the system is such that it fails if and only if all the components fail. This is what we call "parallel-system" and in a similar to series systems way its reliability logic can be represented by a reliability network as it is illustrated in figure 1.4 corresponding to an n-component parallel-system.

Given the reliabilities p_j, $j = 1, 2,...,n$ of the components of an n-component parallel-system we calculate the probabilities $q_j = 1 - p_j$ of the component failures; consequently, by the probability multiplication law, the probability of the entire system failing is

$$Q = \prod_{j=1}^{n} (1 - p_j) = \prod_{j=1}^{n} q_j \tag{1.16}$$

and therefore the over-all system reliability is:

$$R = 1 - \prod_{j=1}^{n} (1 - p_j) = 1 - \prod_{j=1}^{n} q_j \tag{1.17}$$

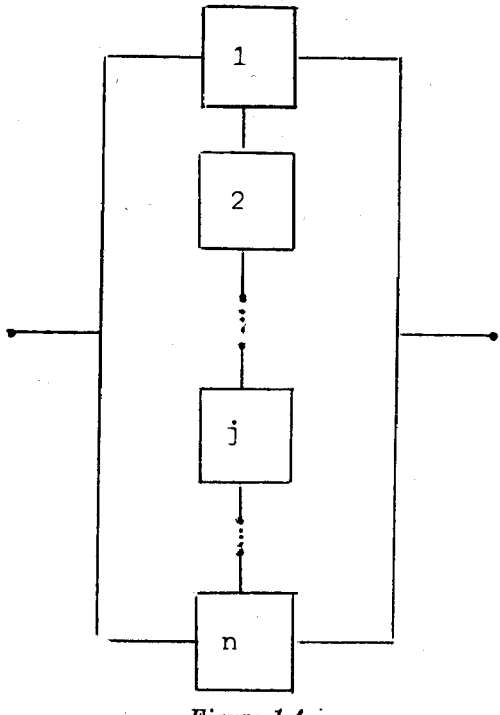


Figure 1.4

Of particular interest is the case in which all components in parallel have equal reliabilities. Such a case arises when all compontents are identical. The formula 1.17 is reduced then to:

$$R = 1 - (1 - p)^n = 1 - q^n \tag{1.18}$$

where p is the common component reliability.

1.4.2 *Parallel-Series Systems*

A simple system, mostly studied in its reliability behaviour, is the "parallel-series" one which is a series combination of parallel subsystems.

Figure 1.5 illustrates a parallel-series system structure with k subsystems (stages) in series and each subsystem i consists of n_i components in parallel.

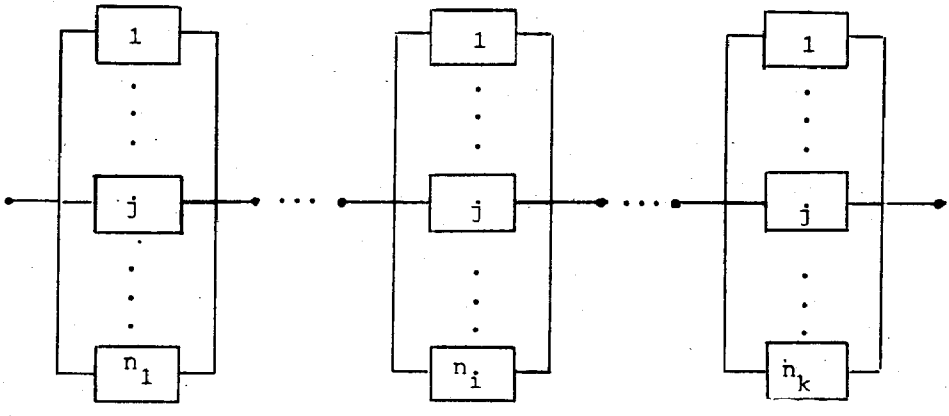


Figure 1.5

Given the reliabilities p_{ij} of the components $j = 1, 2, ..., n_i$ in each subsystem $i = 1, 2, ..., k$ we can calculate the over-all reliability taking

$$R_i = 1 - \prod_{j=1}^{n_i} (1 - p_{ij}) = 1 - \prod_{j=1}^{n_i} q_{ij} \tag{1.19}$$

the reliability of each subsystem i and then

$$R = \prod_{i=1}^{k} R_i = \prod_{i=1}^{k} (1 - \prod_{j=1}^{n_i} (1 - p_{ij})\} = \prod_{i=1}^{k} (1 - \prod_{j=1}^{n_i} q_{ij}) \tag{1.20}$$

the desirable over-all reliability, where

$$q_{ij} = 1 - p_{ij}$$

is the unreliability of the j^{th} component in the i^{th} subsystem.

Of interest is also the special case where all the components in the subsystems are equally reliable that is when

$$p_{ij} = p_i , \quad j = 1, 2, ..., n_i$$

The formulas 1.19 and 1.20 become respectively

$$R_i = 1 - (1 - p_i)^{n_i} = 1 - q_i^{n_i} \tag{1.21}$$

and

$$R = \prod_{i=1}^{k} \{1 - (1 - p_i)^{n_i}\} = \prod_{i=1}^{k} (1 - q_i^{n_i}) \tag{1.22}$$

The reliability improvement via parallel redundancy going to be discussed in next Chapter is directly related with this situation.

1.4.3. *More Complicated Systems*

When a system appears in a more complicated configuration a direct calculation of its reliability, based on the probability theory laws, becomes laborious.

In such a case an indirect method for calculating system reliability is discussed by Caufman (1972) which is considered more efficient than the direct one. This method is built on the concept of the "system structure function".

In order to define this function for a given system we introduce the "component state variables" setting

$x_i = 1$ if the i^{th} component functions
$x_i = 0$ otherwise

The structure function depends on the state variables $x_1, x_2, \ldots, x_k$ (if k components form the system) with the requirement to be equal to 1 when the entire system functions and equal to 0 otherwise, that is:

$$f(x_1, x_2, \ldots, x_k) : \{0,1\}^k \rightarrow \{0,1\} \tag{1.23}$$

The set $\{0,1\}$ is assumed to be equipped with the usual real numbers algebra. The way in order to derive an analytical formula for the function f of a given system is explained below where some general examples are given.

SERIES SYSTEMS

In such a system it is not difficult to verify that its structure function $f(x_1 x_2, \ldots, x_k)$ must be equal to 1 if the state vector $(x_1, x_2, \ldots, x_k)$

equals to (1,1,...,1) and 0 otherwise. One can immediately derive the formula of the function f to be

$$f(x_1,x_2,...,x_k) = x_1.x_2...x_k = \prod_{i=1}^{k} x_i \tag{1.24}$$

and to justify that this function equals to 1 when $(x_1,x_2,...,x_k) = (1,1,...,1)$ while it is zero otherwise.

PARALLEL SYSTEMS

In this situation we are looking for a function f taking the value 0 when the state vector equals to (0,0,...,0) and taking the value 1 otherwise. A function verifying this requirement is

$$f(x_1,x_2,...,x_k) = 1 - \prod_{i=1}^{k} (1 - x_i) \tag{1.25}$$

PARALLEL-SERIES SYSTEMS

Another system configuration in which it is rather easy to calculate its structure function is that of a parallel-series one. Combining the formulas 1.24 and 1.25 we obtain the structure function of such a system (illustrated in figure 1.5) which is:

$$f(x_{11},...,x_{1n_1},...,x_{k1},...,x_{kn_k}) = \prod_{i=1}^{k} \{1 - \prod_{j=1}^{n_i} (1 - x_{ij})\} \tag{1.26}$$

SERIES-PARALLEL SYSTEMS

This is the situation illustrated in figure 1.6 representing a system consisting of k series subsystems in parallel. Each subsystem i contains n_i components in series. For this system we can calculate its structure function and it is:

$$f(x_{11},...,x_{1n_1},...,x_{k1},...,x_{kn_k}) = 1 - \prod_{i=1}^{k} (1 - \prod_{j=1}^{n_i} x_{ij}) \tag{1.27}$$

Note that the component state variables x_{ij} in the formulas 1.26 and

1.27 have been defined so that $x_{ij} = 1$ if the j^{th} component of the i^{th} subsystem functions and $x_{ij} = 0$ otherwise.

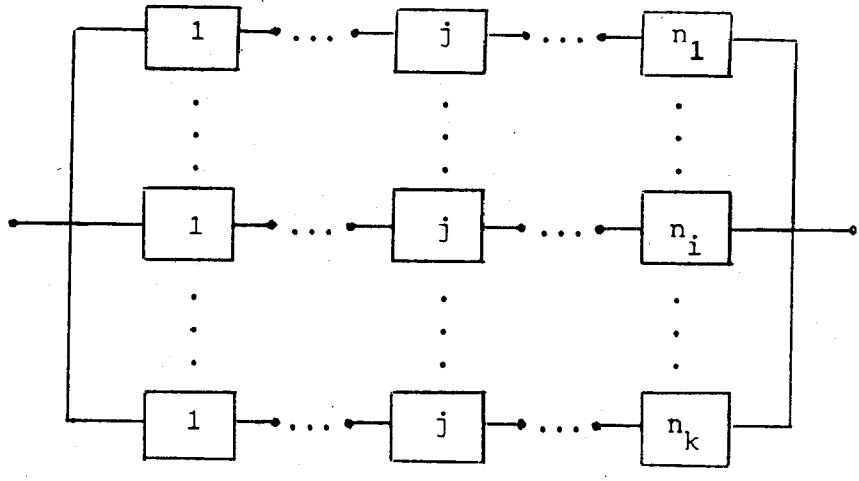


Figure 1.6

Having on hand the structure function of a certain system we can achieve its reliability in terms of the component reliabilities following the steps of the procedure below:

PROCEDURE 1.1

a) Reduce the structure function in its polynomial form performing the operations denoted in it.

b) Set $x^{\alpha}_i = x_i$ (since $x_i = 0$ or 1) for every $\alpha \geqq 1$

c) Replace the component state variables x_i by the component reliabilities p_i. The resulted expression of p_i gives the desirable system reliability in terms of the component reliabilities.

Kaufman (1972) gives a number of numerical examples of this procedure. In the general nature examples presented in subsections 1.4.1 and 1.4.3 the system reliabilities have been derived on the basis of the probability laws. Alternately, we obtain the relevant formulas 1.15, 1.17 and 1.20 from the structure functions 1.24, 1.25 and 1.26 respectively setting directly p_i in the place of x_i since no power x^{α}_i is going to appear performing the operations. Similarly, we obtain the reliability of a parallel-series system from the formula 1.27 and it is:

$$R = 1 - \prod_{i=1}^{k} (1 - \prod_{j=1}^{n_i} p_{ij}) = 1 - \prod_{i=1}^{k} (1 - \prod_{j=1}^{n_i} \{1 - q_{ij}\}) \qquad (1.28)$$

CHAPTER 2

RELIABILITY OPTIMIZATION

2.1 *Introduction to the Redundancy*

A certain system consisting of a given number of components possesses a reliability level depending on the reliabilities of its particular components.

In Chapter 1 the general methodology of calculating the system reliability in terms of the component reliabilities has been discussed. In the present Chapter a theoretical review is going to be presented giving an answer to the following problem:

PROBLEM 2.1

"If the system reliability is regarded too low for our aims under consideration what can we do in order to improve it? What are the effects on the system reliability and on certain cost functions related with it if a change of the system configuration is made in order to obtain higher reliability level?".

An obvious answer to the first part of the problem is to use components of greater reliability. However, there are particular situations in which the effects of such a technique on the over-all reliability may be regarded very poor. In some other cases the component reliabilities are not under our control. Gordon (1956) gives an idea of the poorness of the effects on the system reliability when we increase the component reliability. He considers an original series system consisting of k components whose the reliabilities are taken to be equal to each other, say p. Thus the over-all reliability is given by the formula 1.15 which in this case is reduced to

$$R = p^k \tag{2.1}$$

The function p^k (since $0<p<1$) tends to zero when k becomes large. Thus taking k to be large the component reliability must be very close to unity if the system reliability is not to be regarded so low that the system is worthless.

In addition to poor effects on the over-all reliability, higher component reliability could require a higher component production cost or increase the component weight, volume, e.t.c. and the balance between higher over-all reliability and higher cost functions may be regarded unsatisfactory.

According to what has been up to now discussed, the system reliability improvement via higher component reliability is not the best way in many cases. A more convenient method that is not related with the component reliabilities but with the number of components in the system is what we call "Redundancy". With this technique spare components are added to each basic component of the system in such a number that increases the over-all reliability to a desirable level.

Redundancy can be applied in two ways:

PARALLEL REDUNDANCY

To each basic component a number of n similar spare components is added and all $n+1$ components af the system operate at the same time.

STAND-BY REDUNDANCY

To each basic component a number of an similar spare components is added. Only one component is in function at a time while a spare component is put into operation as soon as the functioning one fails.

This work on hand deals with the application of the multiobjective programming techniques on the problem of parallel redundancy optimization. Therefore, the parallel redundancy effects on the over-all reliability is the main subject of the subsequent sections. However, most of which is discussed in the next pages is applicable to stand-by redundancy as well, if some changes on the calculation of the over-all reliabity are made.

2.2 *Parallel Redundancy Effects on the Over-all Reliability*

According to the definition of the parallel redundancy, given in the section 2.1, the set of one basic component i and the spare ones n_i could

be regarded as a parallel type system whose all $n_i + 1$ components are identical possessing equal to each other reliabilities, say p. Therefore the reliability of this set is calculated by the formula 1.18, that is:

$$R_i = 1 - (1 - p)^{n_i+1} = 1 - q^{n_i+1} \tag{2.2}$$

Assumming n_i to be independent variable, the reliability function $R_i(p,n_i)$ is represented by a single—parameter family of curves on the (R_i,n_i)—plane and its bahaviour is illustrated in the figure 2.1. Clearly, as the component reliability increases, the number n_i of the redundant components in parallel decreases when the over-all reliability remains fixed. Solving the equation 2.2 with respect to n_i we express the number of redundant components as a function of the given component reliability and a specified over-all reliability R_i, that is:

$$n_i = \left[1 - \frac{\log(1 - R_i)}{\log(1 - p)}\right] \tag{2.3}$$

where [x] denotes the biggest integer less than x.

Consider now the entire system consisting of k basic components in series. Let $p_1,p_2,\ldots,p_k$ be the reliabilities of these basic components. Thus the system reliability is:

$$R' = \prod_{i=1}^{k} p_i \tag{2.4}$$

Parallel redundancy can be applied to this system in two different ways. Firstly, we can duplicate the entire system in a sense illustrated by the figure 2.2. A new system is then obtained and its overall reliability is calculated using the formula 1.28, that is:

$$R'' = 1 - (1 - R')^2 = 1 - (1 - \prod_{i=1}^{k} p_i)^2 \tag{2.5}$$

It is clear that

$$R'' > R' \tag{2.6}$$

and in this way we reach a higher over-all reliability duplicating the entire system.

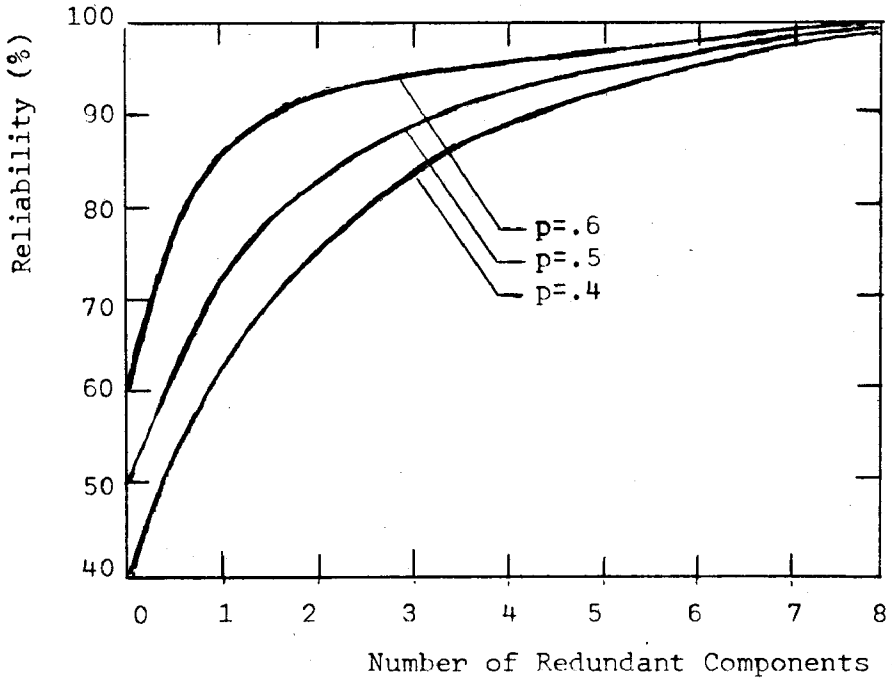


Figure 2.1

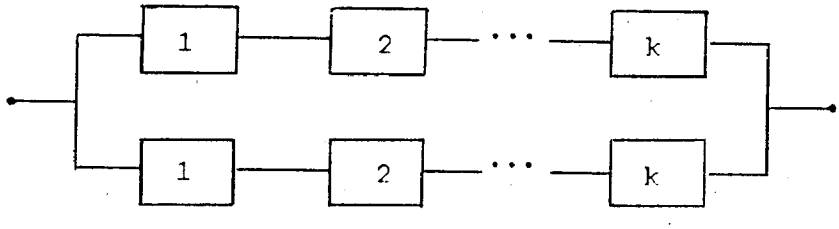


Figure 2.2

Secondly, we can duplicate the particular system components obtaining herefore a parellel-series new system of which the configuration is illustrated in figure 1.5. Thus the over-all reliability of this system is given if we set in the formula 1.22 $n_i = 2$, that is;

$$R''' = \prod_{i=1}^{k} \{1 - (1 - p_i)^2\} \tag{2.7}$$

Comparing now R''' and R', as it has been dor first case, we see that:

$$p_i > p_i^2 \quad \text{or}$$
$$p_i < 2p_i - p^2{}_i \quad \text{or}$$
$$p_i < 1 - (1 - p_i)^2 \quad \text{or}$$

$$\prod_{i=1}^{k} p_i < \prod_{i=1}^{k} \{1 - (1 - p_i)^2\}$$

which means that:

$$R''' > R' \tag{2.8}$$

that is, the original system reliability has been again improved. Finally, between the two methods of applying the parallel redundancy, being described above, the second one is the most effective since from 2.4 and 2.6 we can find:

$$R''' > R'' \tag{2.9}$$

while all cost functions (ie system cost, weight, e.t.c.) related to the system remain unchanged.

The result from the previous comparison, that is duplicating each system component is more effective than duplicating the entire system, is of a more general application valid for any type of original system even though it is not a series one, as it was before. This is what Kaufmann, Cryon and Grouchko (1969) and Kaufmann (1972) point out.

2.3 *The Problem of Parallel Redundancy Optimum Allocation*

Let the original system be one consisting of k basic components. Let

$$\mathbf{n} = (n_1, n_2, \ldots, n_k) \tag{2.10}$$

be the vector whose the entries $n_i = 0,1,2,\ldots$ represent the number of

the redundant components added to each basic component i = 1,2,...,k. Thus the over-all reliability is a function

$$z = R(\mathbf{n}) = R(n_1,n_2,...,n_k) \tag{2.11}$$

which is increasing with respect each n_i assumming the basic component reliabilities p_i to be fixed. This means it is possible to make the system reliability as high as it is desirable adding redundant components.

It is clear that any redundant component added to the basic components of the system increases the system cost and/or weight and/or volume e.t.c. together with the over-all reliability. Generally speaking, a system reliability increasing causes changes to some cost functions related to. Let these cost function be:

$$h_j(\mathbf{n}) = h_j(n_1,n_2,...,n_k) \tag{2.12}$$

for j = 1,2,...,r. Since these resources are normally limitted by upper bounds the over-all reliability cannot be as high as we would like. An of the erliest and mostly important problem in the reliability theory is to find the optimum redundancy allocation, that is the allocation giving the maximum value of the over-all reliability subject to limitations on the cost functions.

In terms of mathematical programming this problem is stated as follows:

$$\begin{aligned} &\text{maximize: } z = R(\mathbf{n}) \\ &\text{subject to: } h_j(\mathbf{n}) \leqq b_j\,, \quad j = 1,2,...,r \\ &\qquad\qquad n_i = 0,1,2,... \end{aligned} \tag{2.13}$$

A wide number of various observations of this problem are made and a similar number of solving algorithms has been developed according to particular assumptions, definitions and the methodology which is followed.

The crucual points making differences between the particular cases of treating the problem 2.13 could be listed as follows:

i) The type of the original system which may be a series one or of some other configuration.
ii) The number of cost functions being 1 or 2,... or r.

iii) The type of the cost functions h_j which could be linear or not.
iv) The particular area of mathematical programming in which the problem of redundancy optimization is embeded, say Dynamic Programming, Integer Programming, Geometric Programming, e.t.c.

In the subsequent sections of this Chapter a theoretical review of the methodology followed in solving the problem 2.13 is discussed. This discussion will be restricted in the presentation of the various solving algorithms where they are not of particular interest through the pages of this dissertation. However, a detailed description will be neccessary in presenting the algorithms that are going to be applied in Chapter 3.

2.4 *Reliability Maximization under Constraints*

2.4.1 *Series Systems; Single Constraint*

A first treatment of the problem 2.13 with a single constraint, $r = 1$, for series systems has been developed by Moskovitz and McLean (1956) and Mine (1956) using Lagrange multiplier method. In applying this technique the variables n_i are assumed to be continuous, but they actually take only integer values. The obtained solutions are therefore approximated.

Bellman and Dreyfus (1962) give a Dynamic Programming approach to the same problem where the single constraint represents a linear cost function. In mathematical terms the particular problem solved by the authors is the following:

$$\text{maximize: } R(\mathbf{n}) = \prod_{i=1}^{k} (1 - q_i^{n_i+1})$$

$$\text{Subject to: } \sum_{i=1}^{k} c_i n_i \leqq b$$

$$n_i = 0,1,2,\ldots \text{ for all } i = 1,2,\ldots,k$$

where c_i is the cost of one component of type i, n_i is the number of redundant components added to the basic component i. The set of all $n_i + 1$ identical components is called "stage i". Finally, b is the budget that the total redundancy cost cannot exceed. The relevant reccurence equation is:

$$f_1(b) = R([b/c_1]) \tag{2.15}$$

$$f_N(b) = \max_{n_N} \{R_N(n_N) \cdot f(b - n_N c_N)\}, \quad N = 2,3,...,k \tag{2.16}$$

where $f_N(b)$ represents the value of the objective function $R(\mathbf{n})$ obtained using an optimal policy while

$$n_N c_N \leqq b \tag{2.17}$$

$R_N(n_N)$ is the reliability of the stage N if n_N redundant components are added and [x] denotes the biggest integer less than x.

Barlow and Proschan (1965) describe an algorithm solving the same problem based on the concept of the undominated redundancy allocation. This procedure (refered as "procedure 1" by the authors) generates a family of "undominated allocations", that is each member has the property that any other allocation giving higher reliability than a member under consideration must need higher cost. In the strict mathematical notation an undominated redundancy allocation is defined as follows:

$$\mathbf{n} \text{ is undominated} \underset{\text{def}}{\longleftrightarrow} \text{for every } \mathbf{n}' \neq \mathbf{n} \text{ then,} \tag{2.18}$$

$$R(\mathbf{n}') > R(\mathbf{n}) \rightarrow \sum_{i=1}^{k} c_i n_i' > \sum_{i=1}^{k} c_i n_i \quad \text{and,}$$

$$R(\mathbf{n}') = R(\mathbf{n}) \rightarrow \sum_{i=1}^{k} c_i n_i' \geqq \sum_{i=1}^{k} c_i n_i$$

The relevant algorithm generating undominated allocations is the following:

ALGORITHM 2.1 *(Procedure 1)*

a) Start with the cheapest cost allocation $\mathbf{0} = (0,0,...,0)$ which is always undominated.

b) If the present allocation is $\mathbf{n}$ obtain a next allocation $\mathbf{n}'$ by adding one component to the stage i for which the quantity:

$$q_i = \frac{1}{c_i} \cdot \{\log R_i(n_i + 1) - \log R_i(n_i)\} \tag{2.19}$$

becomes maximal. This means that we add one component to that stage i for which the over-all reliability increasing per unit of money spent is maximum.

c) Repeat the step b) until a specified cost bound is exceeded.

The main disadvantage of this procedure is that it derives an incomplete family of redundancy allocations, as it is emplasized by the authors, and this family contains undominated members under the additional condition that $\log R(\mathbf{n})$ must be concave. In the particular problem 2.14 in which we are interested in the present subsection this requirement is valid since

$$\log R(\mathbf{n}) = \sum_{i=1}^{k} \log R_i(n_i)$$

and since the considered system is a series one we have

$$\log R_i(\mathbf{n}_i) = \log(1 - q_i^{n_i+1})$$

for all $i = 1,2,...,k$, which means that $\log R_i(n_i)$ are concave and, therefore, $\log R(\mathbf{n})$ is also concave.

In order to obtain a complete family of undominated redundancy allocations for the same problem 2.14, Kettelle (1962) has developed an algorithm based on the Dynamic Programming formulation. This procedure derives undominated redundancy allocations for larger subsystems from undominated allocations for smaller ones. A step-by-step description of this algorithm could be stated as follows:

ALGORITHM 2.2

a) Consider the subsystem consisting of the stages 1 and 2 only.

b) Start with the first undominated allocation (0,0) for this subsystem (1,2).

c) Obtain the next undominated redundancy allocation choosing the cheapest cost allocation with reliability no lower than the one just achieved. If there exist two allocations of identical cost choose the one

with the highest reliability. If there exist two allocations with identical cost and highest reliability choose the one with the lowest component index. If the present allocation is at the intersection of the row i and column j search for the next allocation may be confined to the union of rows 1,2,...,i and columns 1,2,...,j, where j is the number of redundant components in stage 1 and i the number in stage 2.

d) Repeat the steps a) to c) for the next subsystem (2,3) consisting of the stages 3 and 4 only.

e) Repeat the steps a) to c) combining the two subsystems (1,2) and (3,4). Thus, a complete family of undominated redundancy allocations for the subsystem (1,2,3,4) has been achieved.

f) In this way, obtaining a complete family of undominated redundancy allocations for the next two-stage subsystem and combining it with the subsystem achieved just before, we can derive a complete family of allocations for the entire subsystem.

A schematical representation of such a subsystem combination for a six-stage original system is illustrated in figure 2.3.

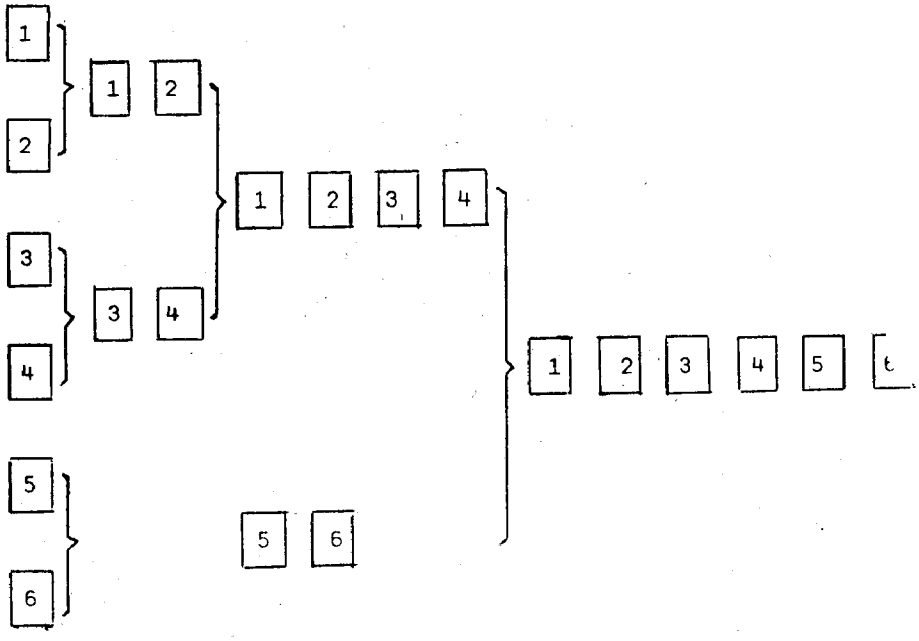


Figure 2.3

2.4.2 *Series Systems; Multiple Constraints*

In this case, with the additional assumption that there are only two constraints, say linear cost and linear weight, Bellman and Dreyfus (1958) have developed a Dynamic Programming procedure combined with Lagrange multiplier method in order to solve the following problem in mathematical terms:

$$\text{maximize: } R(\mathbf{n}) = \prod_{i=1}^{k} (1 - q_i^{n_i+1}) \tag{2.20}$$

$$\text{Subject to: } \sum_{i=1}^{k} c_{ij} n_i = c_j(\mathbf{n}) \leqq b_j, \quad j = 1,2,...,r$$

$$n_i = 0,1,2,... \quad , \quad i = 1,2,...,k$$

for $r = 2$, where c_{i1} and c_{i2} are positive numbers representing the cost and the weight of the i^{th} component in the stage 1 and 2 respectively. Setting now:

$$R_i(n_i) = 1 - q_i^{n_i+1}$$

$$f_N(b_j) = \max_{S_N} \prod_{i=1}^{N} \{R_i(n_i) \cdot \exp(-\lambda c_{i2} n_i)\}$$

where S_N is the set of solutions defined by the constraints in 2.20 and λ is a Lagrange multiplier such that the weight constraint is satisfied, an easy in calculations forward reccurence equation is obtained:

$$f_1(b_1) = \max_{n_1} R_1(n_1) \cdot \exp(-\lambda c_{12} n_1)$$

$$f_N(b_1) = \max_{n_N} \{R_N(n_N) \cdot \exp(-\lambda c_{N2} n_N) \cdot f_{N-1}(b_1 - c_{N1} n_N)\}, \quad N \geqq 2$$

The maximum in the later formula is taken over all n_N such that:

$$0 \leqq n_N \leqq [b_1/c_{N1}]$$

For various values of λ we calculate solutions to the problem 2.20 choosing that gives the highest weight less than b_2.

Another approach to the problem 2.20, given in its general form with r linear constraints, is refered by Barlow and Proschan (1965). Two different procedures (called "procedure 1 for multiple cost factors" and "procedure 2" respectively) are discussed both generating undominated allocations for the problem 2.20. In a similar way to the section 2.4.1 the definition of an undominated redundancy allocation when multiple linear cost factors are involved is stated as follows:

$$\mathbf{n} \text{ is undominated} \underset{\text{def}}{\longleftrightarrow} \text{for every } \mathbf{n}' \neq \mathbf{n} \quad \text{then,} \tag{2.21}$$

$$R(\mathbf{n}') > R(\mathbf{n}) \rightarrow c_j(\mathbf{n}') > c_j(\mathbf{n}) \text{ for at least one } j$$
$$R(\mathbf{n}') = R(\mathbf{n}) \rightarrow c_j(\mathbf{n}') > c_j(\mathbf{n}) \text{ for at least one } j \text{ or}$$
$$c_j(\mathbf{n}') = c_j(\mathbf{n}) \text{ for all } j = 1,2,\ldots,r$$

where the expression of the cost factors $c_j(\mathbf{n})$ is given in the problem 2.20. On the basis of this definition the two algorithms described below derive families (generally incoplete) of undominated redundancy allocations for the problem 2.20.

ALGORITHM 2.3 *(Procedure 1 for multiple cost factors)*

a) Start with the cheapest allocation (0,0,...,0) which is always undominated.

b) Obtain a next allocation by adding one component to the stage i for which the quantity:

$$\eta_i = \{\sum_{j=1}^{r} \alpha_j c_{ij}\}^{-1} \cdot \{\log R(n_i + 1) - \log R(n_i)\} \tag{2.22}$$

is maximum, where α_j are non-negative real numbers summing to unity.

c) Repeat the step b).

The meaning of the step b) is that we add one component to that stage i for which the rate of the over-all reliability increasing and a linear convex combination of the corresponding cost factors to be maximum. The quantities α_j are chosen in the interval (0,1) and to each vector $\boldsymbol{\alpha} = (\alpha_1,\alpha_2,\ldots,\alpha_r)$ one family of undominated allocations is corresponded. If we repeat the algorithm 2.3 for various vectors $\boldsymbol{\alpha}$, say starting from the vector (1,0,...,0) and proceeding until the final vector (0,0,...,0,1) varying the α_j by some fixed increament Δ, we obtain a wider family.

Even if all the possible vectors $\boldsymbol{\alpha}$ have been taken the family resulted in this way is still incomplete.

ALGORITHM 2.4

Let $\boldsymbol{\lambda} = (\lambda_1,...,\lambda_r)$ be a vector of which each $\lambda_j \geqq 0$ and at least one $\lambda_j > 0$. For $i = 1,2,...,k$ obtain $n_i(\boldsymbol{\lambda})$ as the smalest integer m satisfying the inequality:

$$\log R_i(m+1) - \log R_i(m) < \sum_{j=1}^{r} \lambda_j c_{ij} \tag{2.23}$$

This procedure, and the foregoing one, generates an incomplete family of undominated allocations (since $\log R(\mathbf{n})$ is concave in the problem 2.20).

Proschan and Bray (1965) have developed a generalized Kettelle algorithm in order to treat the problem 2.20 in its general case, ie when r cost factors are involved.

Fedorowitz and Mazumdar (1968) use Geometric Programming Techniques in order to solve the problem 2.20. The corresponding Geometric Program is obtained setting:

$$x_i = e^{n_i}$$

and

$$1 - x_i^{\ln q_i} \geqq z_i$$

for $i = 1,2,...,k$, and it is:

$$\text{minimize } f(x_i,z_i) = \prod_{i=1}^{k} z^{-1} \tag{2.24}$$

$$\text{subject to } z_i + x_i^{\ln q_i} \leqq 1, \quad i = 1,2,...,k$$

$$e^{-b_j} \cdot \prod_{i=1}^{k} x_i^{c_{ij}} \leqq 1, \quad j = 1,2,...,r$$

$$x_i, z_i \geqq 0 \text{ for all } i$$

Mizukami (1968) considers the same problem setting:

$$z(\mathbf{n}) = \log R(\mathbf{n}) = \sum_{i=1}^{k} \log R_i(n_i) = \sum_{i=1}^{k} \log(1 - q_i^{n_i+1})$$

Since the functions $R_i(n_i)$ are concave the function $z(\mathbf{n})$ is concave separable. Maximizing $R(\mathbf{n})$, as the problem 2.20 requires, is equivalent to maximize $z(\mathbf{n})$. Therefore, the following concave programming formulation is equivalent to the original one:

$$\text{maximize}: \; z(\mathbf{n}) = \sum_{i=1}^{k} (1 - q_i^{n_i+1}) \tag{2.25}$$

$$\text{subject to}: \; \sum_{i=1}^{k} c_{ij} n_j \leqq b_j, \quad j = 1,2,...,r$$

$$n_i = 0,1,2,... \text{ for all } i$$

This concave program is equivalent now to the following Linear Program:

$$\text{maximize}: \; z = \sum_{i=1}^{k} y_i \tag{2.26}$$

$$\text{subject to}: \; y_i \leqq \lambda_{si} n_i + \mu_{si}, \quad i = 1,...,k, \quad s = 1,...,l$$

$$\sum_{i=1}^{k} c_{ij} n_i \leqq b_j, \quad j = 1,2,...,r$$

$$n_i = 0,1,2,... \text{ for all } i$$

where,

$$\lambda_{si} = \{R(n_{is}) - R_i(n_{i,s-1})\} / (n_{is} - n_{i,s-1})$$
$$\mu_{si} = \{n_{is} R_i(n_{i,s-1}) - n_{i,s-1} R(n_{is})\} / (n_{is} - n_{i,s-1})$$

and

$$n_{i1}, n_{i2}, ..., n_{il}$$

are given values of n_i.

Ghare and Taylor (1969) convert the problem 2.20 into a multidi-

mentional Knapsack problem. The authors prove that the problem 2.20 is equivalent to the program:

$$\text{maximize}: \; z = \sum_{i=1}^{k} \sum_{l=1}^{\infty} \alpha_{il} x_{il} \tag{2.27}$$

$$\text{subject to}: \; \sum_{i=1}^{k} \sum_{l=1}^{\infty} c_{ij} x_{il} \leqq d_j, \quad j = 1,2,...,r$$

$$x_{il} = 0 \text{ or } 1$$

where $x_{il} = 0$ implies $x_{im} = 0$ for all $m > 1$, $\alpha_{il} = \ln(1 - q_i^{l+1}) - \ln(1 - q_i^{l})$

nd

$$d_j = b_j - \sum_{i=1}^{k} c_{ij}$$

A modified branch and bound algorithm developed by Ghare and Walters is used in order to solve the program 2.27.

2.4.3 *More complicated Situations*

Jensen (1970) considers the problem of optimum redundancy design for a series original system duplicating not each system component but duplicating the series subsystems in which the system can be subdivided. If this is made a "series-parallel-series" redundant reliability network is resulted, an example of which is illustrated in figure 2.4. In this example the original system (1,2,3,4,5,6) is divided into three subsystems (1,2), 3, (4,5,6) and the first and third ones have been duplicated while the second one has been tripled. In general, a redundant design is defined by a set af triplets:

$$D = \{(i_1,j_1;n_1), (i_2,j_2;n_2),...,(i_k,j_k;n_k)\}$$

In the example of the figure 2.4 we have:

$$D = \{(1,2;2), (3,3;3), (4,6;2)\}$$

A dynamic programming formulation is developed in order to solve the problem which is in mathematical terms:

$$\text{maximize } R(D) = \prod_{(i,j;n)} R(i,j;n) \tag{2.28}$$

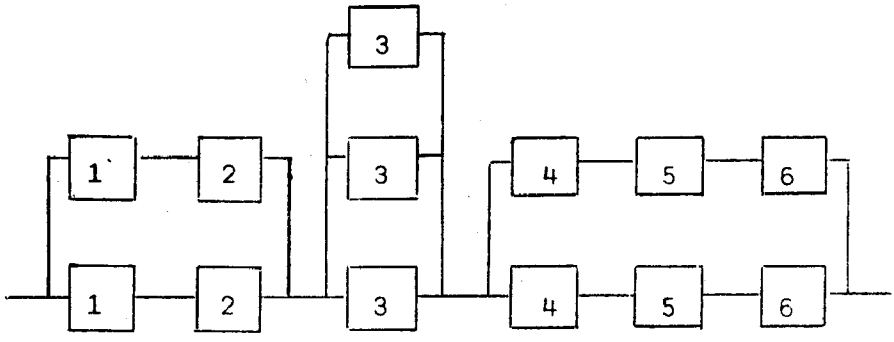


Figure 2.4

Finally, Hendrix and Stedry (1974) work on the following problem: "Given a basic component system design with r goals and k components and given a wealth constrain w, it is desirable to determine $\mathbf{n} = (n_1,..., n_k)$, where n_i is the number of identical components of type i, in order to maximize:

$$R = \sum_{j=1}^{r} r_j Q_j$$

where R is the expectation of total reward, r_j is the reward associated with the goal G_j and Q_j is the probability that the goal G_j will be met, subject to

$$\sum_{i=1}^{k} c_i n_i \leqq w$$

where c_i is the cost of one component of type i".

2.5 *Cost Functions minimization under a Reliability Constraint*

So far, in the section 2.4 we were occupied with the problem of achieving maximum reliability via parallel redundancy given cost-function constraints associated with the number of redundant components in the

system. This problem means that some upper bounds for the system cost, weight, volume, e.t.c., has been already specified and we are seeking for the redundancy vector **n** maximizing the over-all reliability. But, in applying this methodology to particular real life problems, the resulted maximum over-all reliability could be regarded very poor for our aims. An obvious solution to this later problem is to increase the bounds in the cost-function constraints and then to solve the maximum reliability problem from the beginning or to find a new solution of the basis of the past one by sensitivity analysis.

An alternative method is to consider the "inverse of the maximum reliability problem", that is the problem of simultaneously minimizing the cost functions related with the redundant vectors while maintaining the over-all reliability at some desirable level.

In the case of one cost function only Kettelle's algorithm as well as Barlow and Proschan's one, both refered in section 2.4.1 as algorithm 2.1 and 2.2 respectively, could be applied for solving the inverse of the maximum reliability problem in this case, which is in mathematical terms:

$$\text{Minimize}: \; c(\mathbf{n}) = \sum_{i=1}^{k} c_i n_i \tag{2.29}$$

$$\text{subject to}: \; R(\mathbf{n}) = \prod_{i=1}^{k} (1 - q_i^{n_i+1}) \geqq L$$

$$n_i = 0,1,2,\ldots \text{ for all } i$$

It is sufficient to seek for that member of the redundancy allocation family, generated by these algorithms, corresponding to a reliability level higher than L but, it cannot be said for the case of multiple cost functions.

Tillman and Liitswager observe a certain objective function of redundancy allocations to be optimized while a set of constraints (linear or not) must be satisfied and the over-all reliability must exceed a desirable level. Speaking in mathematical terms the solved problem is stated as follows:

$$\text{Optimize}: \; z(\mathbf{n}) = \sum_{i=1}^{k} f_i(n_i) \tag{2.30}$$

$$\text{Subject to}: \sum_{i=1}^{k} g_{ij}(n_i) \leqq b_j \quad , \quad j = 1,2,...,r$$

$$\prod_{i=1}^{k} (1 - q_i^{n_i+1}) \geqq L$$

$$n_i = 0,1,2,... \text{ for all } i$$

Where $z(\mathbf{n})$ is the objective function of the system to be optimized, $f_i(n_i)$ is the objective function at stage i, $g_{ij}(n_i)$ is the amount of the j^{th} resource consumed at stage i. An integer programming formulation of the problem 2.30 is made giving the optimum redundancy allocation:

$$\text{Optimize}: z = \sum_{i=1}^{k} \sum_{m=0}^{n_i'} \Delta f_{im} n_{im} \tag{2.31}$$

$$\text{Subject to}: \sum_{i=1}^{k} \sum_{m=0}^{n_i'} \Delta g_{ijm} n_{im} \leqq b_j \,, \quad j = 1,2,...,r$$

$$\sum_{i=1}^{k} \sum_{m=0}^{n_i'} \Delta \ln(1 - q_{im}^{n_{im}+1}) \geqq \ln L$$

$$n_{io} = 1$$
$$n_{im} - n_{i,m-1} \leqq 0$$
$$n_{im} \geqq 0 \text{ for all i and m}$$

where m is an index used to denote a particular redundant component at the stage i, n_{im} is the m^{th} redundancy at stage i with $n_{im} = 1$ for $m \leqq n_i$ and $n_{im} = 0$ for $n_i < m \leqq n_i'$ (n_i' is the maximum number of redundant units used at stage i),

$$n_i = \sum_{m=1}^{n_i'} n_{im}$$

is the number of redundant components at stage i, $\Delta f_{im} = f_{im} - f_{i,m-1}$ with $\Delta f_{io} = f_{io}$ and Δg_{ijm} and

$$\Delta \ln(1 - q_{im}^{n_{im}+1})$$

are similarly defined. The particular problems solved by this method are a) Maximize Reliability subject to multiple linear cost function constraints, b) Minimize cost subject to multiple non-linear and separable restraint functions while maintaining an acceptable reliability level and c) Optimal choise of design for parallel redundancy system.

The general situation of minimizing several cost function simultaneously while maintaining the over-all reliability over a desirable level is of our interest in the next Chapter 3. It seems that Multiobjective Programming Techniques could be applied in order to give an answer to this problem.

CHAPTER 3

MULTIOBJECTIVE TECHNIQUES IN PARALLEL REDUNDANCY ALLOCATION

3.1 *Multiobjective Programming Methods*

Practitioners and Theorists enganged on real world problems for decision making feel that in a large number of situations several objectives should be simultaneously considered.

In such problems what it is desirable is a vector

$$\mathbf{x} = (x_1, x_2, \ldots, x_k) \tag{3.1}$$

where x_i are non-negative real numbers and the vector $\mathbf{x}$ must simultaneously optimize (maximize or minimize) r real-valued functions:

$$f_j(\mathbf{x}) = f_i(x_1, \ldots, x_k) \tag{3.2}$$

Without loss the generality we assume the r functions f_j to be minimized A set of restraints:

$$h_m(\mathbf{x}) = h_m(x_1, x_2, \ldots, x_k) \leqq b_m, \qquad m = 1,2,\ldots,l \tag{3.3}$$

define a set

$$C^k \leqq R^k \tag{3.4}$$

of all feasible vectors $\mathbf{x}$ that are regarded to be candidate solutions.

We note that the requirement of this problem, namely the simultaneous minimizing of all objective functions is hardly met in practice. For example in a transportation problem we would like maximize the transportation velocity and minimize the transportation cost at the same

time. Clearly, it is very difficult to obtain both,since any increasing in the velocity implies higher cost. How can we overcome this difficulty?

Eilon (1972) suggests a variety of methods that could be applied giving efficient solutions that balance the optimum values of the objectives around a tolerable level.

To do this it is neccessary to compare the values of the objective functions not separately but simultaneously. In other words, we need compare the various vectors

$$(f_1(\mathbf{x}),\ f_2(\mathbf{x}),...,f\ (\mathbf{x})) \tag{3.5}$$

with each other. This is essentially the basic idea of the following mathematical formulation of the multiobjective problem stated in the beginning of this section {Bod (1963) and Rudeanu (1969)}.

$$\begin{aligned} &\text{Minimize}: \ \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}),f_2(\mathbf{x}),...,f\ (\mathbf{x})) \\ &\text{Subject to}: \ h_m(\mathbf{x}) \leqq b_m\,, \quad m = 1,2,...,l \\ &\qquad\qquad x_1,x_2,...,x_k \geqq 0 \end{aligned} \tag{3.6}$$

where $\mathbf{F}(\mathbf{x})$ is a vector-valued function taking values in R^r, that is $\mathbf{F}(\mathbf{x}) \leqq R^r$
and its entries $f_j(\mathbf{x})$ are r real-valued functions.

Saying "minimum of $\mathbf{F}(\mathbf{x})$" means the existence of some relation defined on the field fo values R^r of the function $\mathbf{F}(\mathbf{x})$, denoted by the symbol $\leqq$, on the basis of which two any values $\mathbf{F}$ and $\mathbf{F}'$ in the set R^r will be comparable to each other.

One may define in R^r any relation in order to compare its elements with each other, but the most common in use ones, related to the real-life situations, are the following:

RELATION VIA A UTILITY FUNCTION

It is defined as follows:

$$(f_1,...,f_r) \leqq (f_1',...,f_r') \quad \text{iff} \quad U(f_1,...,f_r) \leqq U(f_1',...,f_r') \tag{3.7}$$

where the function

$$U : R^r \to R \tag{3.8}$$

is a real-valued "utility" function, non-decreasing with respect to each function f_j, $j = 1, 2,...,r$. For example, in the case of two objectives only, the function U could be chosen as the fractional form:

$$U(f_1,f_2) = \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}, \qquad f_1(\mathbf{x}) < 0, \qquad f_2(\mathbf{x}) > 0$$

or it could be chosen as the additive utility function:

$$U(f_1,f_2) = u_1\{f_1(\mathbf{x})\}^{b_1} + u_2\{f_2(\mathbf{x})\}^{b_2}$$

commonly used in the theory of consumer's choice, where u_1, u_2, b_1 and b_2 are positive and f_1 and $f_2 > 0$ for all $\mathbf{x}$.

Thus, considering U as afunction of the vector $\mathbf{x}$ in C^k, we may write:

$$U(f_1(\mathbf{x}),...,f_r(\mathbf{x})) = u(\mathbf{x}) \in R \tag{3.9}$$

Therefore, any solution of the single-objective program:

$$\begin{aligned} &\text{Minimize: } u(\mathbf{x}) \\ &\text{Subject to: } h_m(\mathbf{x}) \leqq b_m, \quad m = 1, 2,...,l \\ &\qquad\qquad x_1,x_2,...,x_k > 0 \end{aligned} \tag{3.10}$$

is also a solution of the problem 3.6, the set R^r being equipped with the relation 3.7. To prove it, let $\mathbf{x}^*$ be an optimal solution of the problem 3.10. Therefore,

$$u(\mathbf{x}^*) \leqq u(\mathbf{x}) \text{ for every } \mathbf{x} \in C^k$$

and so,

$U(f_1(\mathbf{x}^*),...,f_r(\mathbf{x}^*)) \leqq U(f_1(\mathbf{x}),...,f_r(\mathbf{x}))$, for every $\mathbf{x} \in C^k$. Finally, by 3.7

$$(f_1(\mathbf{x}^*),...,f_r(\mathbf{x}^*)) \leqq (f_1(\mathbf{x}),...,f_r(\mathbf{x})), \text{ for every } \mathbf{x} \in C^k$$

which means that $\mathbf{x}^*$ is also an optimal solution of the problem 3.6.

The "trade-off" method refered by Eilon (1972) can be considered particular case of the relation 3.7 if we adopt the U to be the weighted average of the objective functions f_j, that is:

$$U(f_1(\mathbf{x}),...,f_r(\mathbf{x})) = \sum_{j=1}^{r} w_j f_j(\mathbf{x}) = u(\mathbf{x}) \tag{3.11}$$

where the weights are chosen between 0 and 1 and they sum to unity. The meaning of the weights is that they measure the contribution of each objective function f_j to the "trade-off" function U according to its importance. The choice therefore of the numbers w_j is a particular problem depending on the real-life problem goals.

LEXICOGRAPHIC RELATION

It is defined as follows:

$$\begin{aligned}
&(f_1,\ldots,f_r) \leqq (f_1',\ldots,f_r') \quad \text{iff} \\
&f_1 < f_1' \quad \text{or} \\
&(f_1 = f_1' \text{ and } f_2 < f_2') \quad \text{or} \\
&\ldots\ldots\ldots\ldots \\
&(f_j = f_j', \quad j = 1,2,..,r-1 \text{ and } f_r < f_r') \text{ or} \\
&(f_j = f_j', \quad \text{for all } j)
\end{aligned} \tag{3.12}$$

One can easily verify that the set R^r equipped with the relation 3.12 is a totally ordered set and so, if there exists a minimal element $\mathbf{F}^*$ of the set $\mathbf{F}(C^k)$ it is unique.

"Optimization in tandem" refered by Eilon (1972) is embeded into this case. The relevant procedure in order to obtain a vector $\mathbf{x}$ in C^k minimizing the function $\mathbf{F}(\mathbf{x})$ could be described as follows:

*Adopt some preference ordering between the objectives f_j according of their importance in the particular problem under consideration. Let a decreasing priority be defined by the sequence $f_1, f_2,\ldots,f_r$.

*Minimize f_1 subject to the original restraints. Let the set of its solutions be C_1^k; minimize f_2 over this set. Let C_2^k be the set of its solutions; minimize f_3 over the new set, e.t.c. In general, minimize the next objective function f_j over the set C^k_{j-1} of the optimal solutions of the immediately previous function f_{j-1} minimized over the set C^k_{j-2}.

*The procedure will be terminated either when a set of optimal solutions is met containing only one element, or after having minimized the last objective function f_r. In the primer case the solution of the problem 3.6 is unique while in the later the solutions are all the elements contained in the last set.

More precisely, the described procedure leads to a sequence of single-objective function programs as follows:

$$\begin{aligned}
&\text{for } j = 1: \text{ Minimize}: f_1(\mathbf{x}) \\
&\qquad \text{Subject to}: h_m(\mathbf{x}) \leqq b_m, \quad m = 1,2,\ldots,l \\
&\qquad\qquad x_1,\ldots,x_k \geqq 0
\end{aligned} \tag{3.13}$$

$$\text{for } j > 1: \text{ Minimize}: f_j(\mathbf{x}) \tag{3.14}$$
$$\text{Subject to}: h_m(\mathbf{x}) \leqq b_m, \quad m = 1,2,...,l$$
$$f_\mu(\mathbf{x}) = f^*_\mu, \quad \mu = 1,2,...,j-1$$
$$x_1,...,x_k \geqq 0$$

where $f_1^*, f_2^*, ..., f^*_{j-1}$ denote the minimum values of the functions $f_1(\mathbf{x})$, $f_2(\mathbf{x}),...,f_{j-1}(\mathbf{x})$ obtained by the solution of the previous $j-1$ programs 3.14.

One easily verify that any solution $\mathbf{x}^*$ achieved from the sequence 3.14 is also a solution of the problem 3.6, the set R^r being provided with the lexicographic relation.

Finally, we note that this method is better than the trade-off one in that no measure of the objective importance is needed; it is sufficient simply to say which objective is the most important between the remainder ones. In the other hand, the disadvantage of a larger volume of calculations should be emphasized.

COMPONENT-TO-COMPONENT RELATION

It is defined as follows:

$$(f_1,...,f_r) \leqq (f_1',...,f_r') \quad \text{iff} \quad f_j \leqq f_j', \text{ for all } j \tag{3.15}$$

The set R^r equipped with the relation 3.15 is a partially ordered set and therefore the set $\mathbf{F}(C^k)$, which is a subset of R^r, adopts several minimal elements (if they exist) being incomparable with each other with respect the component-to-component relation.

It is no difficult one to see that any optimal element in $\mathbf{F}(C^k)$ according to lexicographic relation in R^r is also an optimal element of the same set if we consider the set R^r to be provided with the component to-component relation, but the inverse is not valid. Therefore, applying successively the procedure described for the lexicographic relation and cancelling all the vectors in $\mathbf{F}(C^k)$ that are comparable to the optimal vector obtained each time we can find all the incomparable minimal vectors in $\mathbf{F}(C^k)$.

Surely, this last relation and its associated procedure is the most realistic of the three. But the difficulty in the practical applications becomes clear looking at the very large volume of calculations.

Rudeanu (1969) has applied the lexicographic and component-to

component relations in order to obtain the solution of the problem 3.6 in the case of "pseudoboolean" objective functions, that is when the variables x_i are $0 - 1$ valued.

The main purpose of the present work is to point out the possibility of applying the multiobjective methodology in the problem of simultaneously minimizing several cost functions while maintaining the over-all reliability over a desirable level and, even though we introduce several objectives, any method generating a family of undominated redundancy allocations may be applied at least in the case of adopting the first of the three relations described before.

In the subsequent sections of this Chapter we are going to prove it and to show the way of applying, in the trade-off treatment, the Kettelle's and Barlow-Proschan's algorithms described in the previous Chapter 2.

3.2 *Formulation of the Parallel Redundancy Allocation Problem*

The Parallel Redundancy Allocation Problem which is going to be traeted is the following:

"Obtain the optimum number of redundant components in parallel at each stage i of a system consisting of k stages in series so that r objective functions — say system cost, weight, e.t.c. — $f_j(n_1,n_2,...,n_k)$ to be minimized while the over-all reliability $R(n_1,n_2,...,n_k)$ must be maintained over a given level".

According to what has been discussed, the following explicit formulas are availlable:

$$f_j(\mathbf{n}) = \sum_{i=1}^{k} c_{ij}(n_i + 1)\,, \quad j = 1,2,...,r \tag{3.16}$$

and

$$R(\mathbf{n}) = \prod_{i=1}^{k} R_i(n_i) = \prod_{i=1}^{k} (1 - q_i^{\,n_i+1}) \tag{3.17}$$

where $c_{ij} > 0$ is the linear cost coefficient of the i^{th} component in the j^{th} linear cost function f_j, q_i is the unreliability of a component of type i, n_i is the number of redundant components in parallel at the stage i and $R_i(n_i)$ is the reliability of the stage i.

The foregoing sections of this Chapter lead to the following multiobjective programming formulation:

$$\text{minimize: } \mathbf{F(n)} = \left(\sum_{i=1}^{k} c_{i1}(n_i + 1),\ldots, \sum_{i=1}^{k} c_{ir}(n_i + 1)\right) \tag{3.18}$$

$$\text{Subject to: } R(\mathbf{n}) = \prod_{i=1}^{k} (1 - q_i^{n_i+1}) \geqq L$$

$$n_i = 0,1,2,\ldots \text{ for all } i$$

This last program can be simplified reformating the entries $f_j(\mathbf{n})$ of the vector-valued objective function $\mathbf{F(n)}$ to the form:

$$f_j(\mathbf{n}) = \sum_{i=1}^{k} c_{ij} n_j + \sum_{i=1}^{k} c_{ij}$$

Since the last term is independent of the n_i can be ignored. Thus, any optimal solution of the problem 3.18 is also an optimal solution of the program 3.19 described bellow and visa versa.

$$\text{minimize: } \mathbf{F(n)} = \left(\sum_{i=1}^{k} c_{i1} n_1,\ldots, \sum_{i=1}^{k} c_{ir} n_i\right) \tag{3.19}$$

$$\text{Subjec to: } R(\mathbf{n}) = \prod_{i=1}^{k} (1 - q_i^{n_i+1}) \geqq L$$

$$n_i = 0,1,2,\ldots \text{ for all } i$$

In this program only one constraint is involved, the reliability one.

3.3 *Trade-off Application*

Choosing some proper weight-vector $\underline{w} = (w_1, w_2, \ldots, w_r)$ with entries between 0 and 1 and summing to unity, we may calculate the trade-off function for the program 3.19 as follows:

$$U(f_1(\mathbf{x}),\ldots,f_r(\mathbf{x})) = \sum_{i=1}^{k} \left(\sum_{j=1}^{r} w_j c_{ij}\right) n_i \tag{3.20}$$

In applying the trade-off technique to real world multiobjective problems we have to find the values of the weights w_j. Unfortunately, there is no some general rule to do this since the choise of these values depends on the particular situation. For example, in a transportation problem with two goals, namely the transportation cost and the over-all transportation time, the relative importance of the cost and the time will depend on the conditions of the particular realistic situation. Since the choise of w_j is strongly related to the priorities of the objectives this choise depends on the particular conditions of the problem.

B. Roy (1969) in his article "Classement et choix en presence de points de vue multiples" (that is "Classification and choise with multiple objectives") gives some examples of the choise of the weights w_j. If, for example, all the objectives are characterized with an equivalent priority we may choose $w_j = 1/r$. In the other hand, if the objectives must be strictly ordered we may set $w_j = 2^{r-j}/(2^{r+1} - 1)$. An intermediate situation is the choise of the weights for the examination papers where some papers could be of equal priorities and some other of greater or less importance.

The function 3.20 is non-decreasing with respect to each f_j and being considered as a function of the redundancy vector $\mathbf{n}$, ie $u(\mathbf{n})$, is a non-decreasing function of each n_i. Therefore, the problem 3.19 is reduced to the following single-objective function program:

$$\text{Minimize: } u(\mathbf{n}) = \sum_{i=1}^{k} \left(\sum_{j=1}^{r} w_j c_{ij} \right) . n_i \tag{3.21}$$

$$\text{Subject to: } R(\mathbf{n}) = \prod_{i=1}^{k} (1 - q_i^{n_i+1}) \geqq L$$

$$n_i = 0,1,2,\ldots \text{ for all } i$$

In this way, the problem of finding the optimum parallel redundancy allocation simultaneously minimizing several cost functions while keeping the over-all reliability over some level has been reduced to a problem of finding the optimum allocation minimizing a single trade-off function under the same reliability restriction.

Any method, included the ones discussed in Chapter 2, solving the problem of achieving the optimum redundancy allocation that minimize

a single cost function while maintaining the over-all reliability at some level (specified or not) can be applied to the program 3.21. Barlow and Proschan's procedure for generating an incomplete family of undominated allocations,as well as Kettelle's algorithm deriving a complete family of undominated allocations can be applied to the problem 3.21, and both generate a solution to the original problem 3.19.

A question arises at this point: "Given some undominated redundancy allocation in the problem 3.21 is it also an undominated one in the problem 3.19? Is the inverse valid?".

In order to have an answer to the first part of the question, let $\mathbf{n}$ be some undominated allocation in the problem 3.21. According to the defition of the undominated allocation (see formula 2.21), for every allocation $\mathbf{n}' \neq \mathbf{n}$ the following relations are valid.

$$R(\mathbf{n}') > R(\mathbf{n}) \rightarrow u(\mathbf{n}') > u(\mathbf{n})$$
$$R(\mathbf{n}') = R(\mathbf{n}) \rightarrow u(\mathbf{n}') \geqq u(\mathbf{n})$$

Since the function U is non-decreasing with respect each f_j the statement

$$u(\mathbf{n}') > u(\mathbf{n}) \rightarrow U(f_1(\mathbf{n}'),\ldots,f_r(\mathbf{n}')) > U(f_1(\mathbf{n}),\ldots,f_r(\mathbf{n}))$$

implies that there exists at least one j for which $f_j(\mathbf{n}') > f_j(\mathbf{n})$. In the other hand the statement:

$$u(\mathbf{n}) = u(\mathbf{n}') \rightarrow U(f_1(\mathbf{n}),\ldots,f_r(\mathbf{n})) = U(f_1(\mathbf{n}'),\ldots,f_r(\mathbf{n}'))$$

implies either $f_j(\mathbf{n}') = f_j(\mathbf{n})$ for all j, or there exists at least one j for which $f_j(\mathbf{n}') > f_j(\mathbf{n})$ since the relation $f_j(\mathbf{n}') < f_j(\mathbf{n})$ for all j implies:

$$U(f_1(\mathbf{n}'),\ldots,f_r(\mathbf{n}')) < U(f_1(\mathbf{n}),\ldots,f_r(\mathbf{n}))$$

which is not true on the basis of our assumption. Summarizing the results we have find that:

$$\begin{aligned} &R(\mathbf{n}') > R(\mathbf{n}) \rightarrow f_j(\mathbf{n}') > f_j(\mathbf{n}) \text{ for at least one } j \\ &R(\mathbf{n}') = R(\mathbf{n}) \rightarrow f_j(\mathbf{n}') = f_j(\mathbf{n}) \text{ for all } j \text{ or} \\ &\qquad\qquad\quad f_j(\mathbf{n}) > f_j(\mathbf{n}) \text{ for at least one } j \end{aligned} \tag{3.22}$$

which means that the allocation $\mathbf{n}$ is also an undominated allocation in the problem 3.19.

For every $\mathbf{n}' \neq \mathbf{n}$ the definition 3.22 is valid but it is easy to see that:

$$f_j(\mathbf{n}') > f_j(\mathbf{n}) \text{ for at least one } j \leftrightarrow$$
$$U(f_1(\mathbf{n}'),\ldots,f_r(\mathbf{n}')) > U(f_1(\mathbf{n}),\ldots,f_r(\mathbf{n}))$$

and therefore $R(\mathbf{n}') > R(\mathbf{n})$ does not imply $u(\mathbf{n}') > u(\mathbf{n})$ which means that $\mathbf{n}$ is not an undominated allocation for the problem 3.21 even though it is an undominated allocation in the problem 3.19.

Thus, we have proved the following property:

PROPERTY 3.1

Every undominated parallel redundancy allocation in the problem 3.21 is also an undominated one in the problem 3.19. The inverse is not valid.

3.4 *Barlow and Proschan's Procedure Application*

In applying the algorithm 2.1 (Barlow and Proschan's procedure 1 for a single cost factor) to the problem 3.21 we do not expect a complete family of undominated allocations for the problem 3.19, since this algorithm derives an incomplete family for the problem 3.21.

Thus we create a new undominated redundancy allocation for the problem 3.19 by adding one redundant component at the stage i for which the increasing of the over-all reliability per unit of average "cost" spent is the maximum. Mathematically speaking, the stage i is that maximizing the quantity:

$$Q_i = \{\sum_{j=1}^{r} w_j c_{ij}\}^{-1} \cdot \{\log(1 - q_i^{n_i+2}) - \log(1 - q_i^{n_i+1})\} \qquad (3.23)$$

It is worth to note that, from a computational point of view, we need the same calculations for solving the problem of obtaining the optimum allocation with minimum several cost functions and reliability over a given level with those in applying the algorithm 2.3 for solving the problem of achieving the optimum allocation maximizing the reliability under several cost constraints.

As a numerical example let the table 3.1 contains the data of a redundancy allocation problem with two cost functions, say the system cost and the system weight, for a four-stages original system and a reliability level $L = .90$. We want to find that parallel redundancy allocation with minimum system cost and system weight keeping the system reliability greater than .90.

Desiring to treat this problem by the trade-off method we have to

choose the weights w_1 for the system cost and w_2 for the system weight. Let:

$$\mathbf{w} = (w_1, w_2) = (.25, .75)$$

The numbers .25 and .75 are hypothetical being arbritrarily chosen for numerical purposes only. If a real world problem were to be solved we would remind the remark on the weight choice of the section 3.3.

i	c_{i1}	c_{i2}	q_i
1	1.20	5.00	.20
2	2.30	4.00	.30
3	3.40	8.00	.25
4	4.50	7.00	.15

Table 3.1

Appendix-I contains a Computer Program creating the family of undominated parallel redundancy allocations for this problem. The input of the program is a table of the form 3.1. The output includesthe family of allocations extending up to the given reliability level (in the present example up to .99955) and the corresponding trade-off, cost and weight function values.

Figure 3.1 represents graphically the results with the over-all reliability against the trade-off function.

3.5 *Kettelle's Algorithm Application*

The Algorithm 2.2 (Kettelle's algorithm) generates a complete family of undominated parallel redundancy allocations for the problem 3.21 but not for the original multiobjective program 3.19, as it has been justified by the property 3.1.

Solving the example of the section 3.4 by this method the calculations are made by two computer programs developed in Appendix-II. The first program generates a complete family of undominated allocations for the subsystem consisting of the stages 1 and 2 and, by repetition, for the subsystem consisting of the stages 3 and 4. The second program derives a complete family of undominated parallel redundancy allocations for the problem 3.21 (being incomplete for the original 3.19)

combining the two subsystems in order to build the entire system consisting of the stages 1, 2, 3, 4.

Figure 3.2 is a diagrammatical representation of the final family with the trade-off function against the over-all reliability.

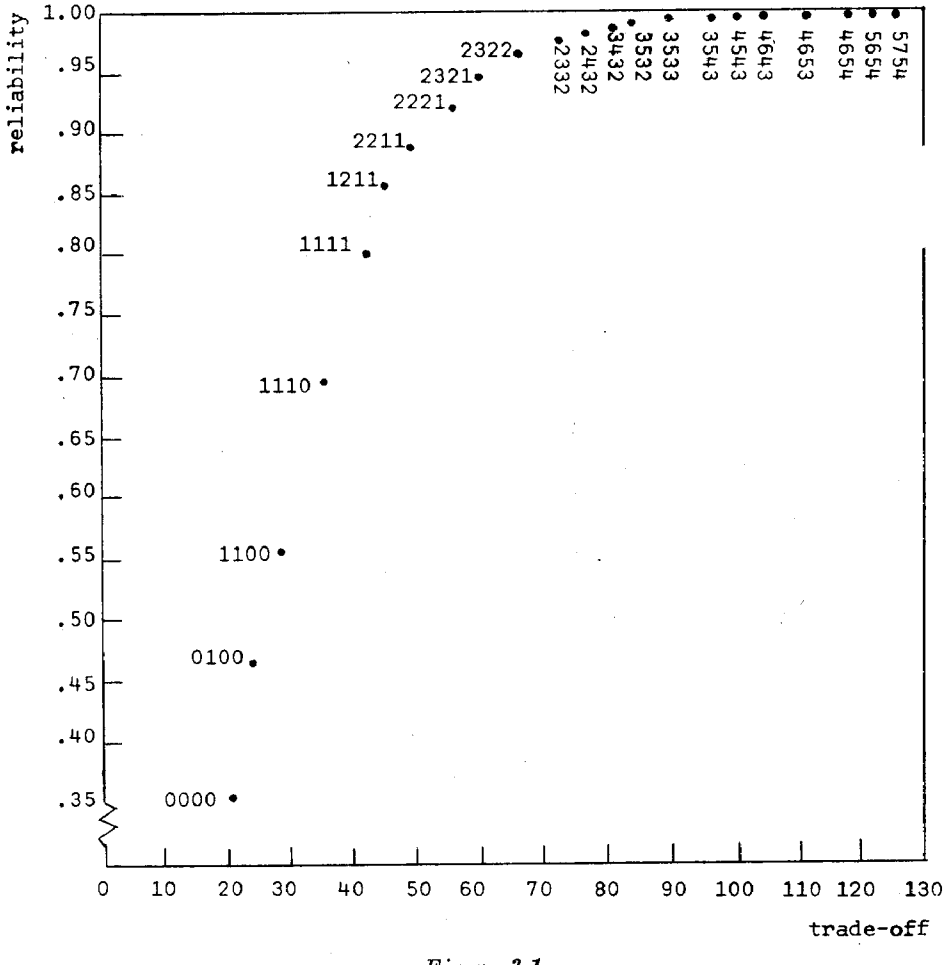


Figure 3.1

3.6 *Discussion of the Results and Conclusions*

Before proceeding to a comparison of the results we note that in applying Algorithm 2.1 to the example of the table 3.1 the cost and weight values as well as the trade-off function values have been calculated in

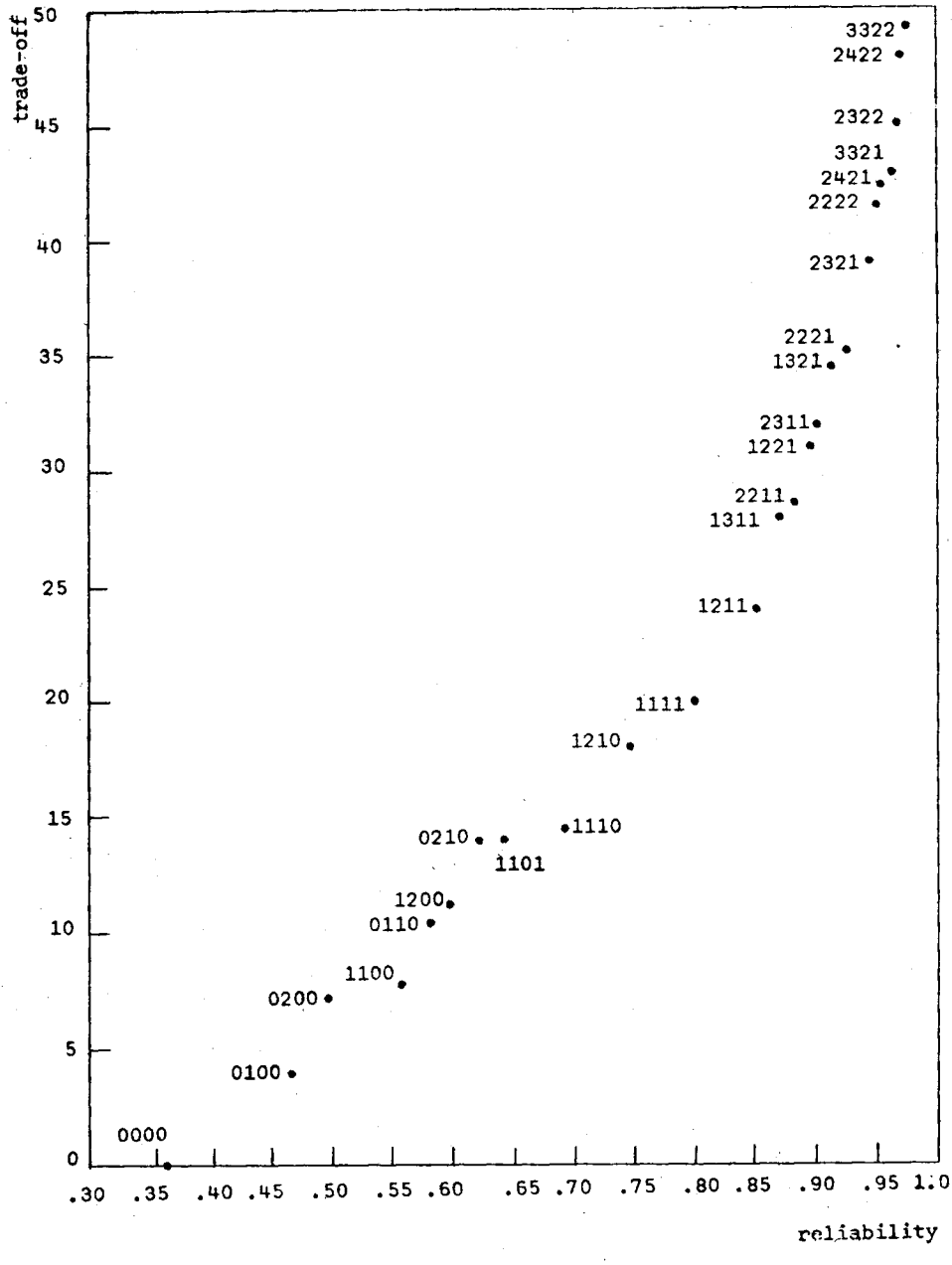

trade-off
50
45
40
35
30
25
20
15
10
5
0
3322
2422
2322
3321
2421
2222
2321
2221
1321
2311
1221
2211
1311
1211
1111
1210
1110
0210
1101
1200
0110
1100
0200
0100
0000
.30 .35 .40 .45 .50 .55 .60 .65 .70 .75 .80 .85 .90 .95 1.0
reliability

Figure 3.2

order to include their constant initial values, while the corresponding values in applying Algorithm 2.2 to the same problem do not. Therefore, in order to compare the results obtained by these algorithms we have to subtract the initial values from the cost functions and trade-off ones achieved by Algorithm 2.1.

The figures 3.1 and 3.2 may be used in order to compare the families of undominated parallel redundancy allocations derived by both procedures. Inspecting these figures we see that Algorithm 2.2 (kettelle) generates a wider family than Algorithm 2.1 (Barlow and Proschan) containing a larger number of allocations.

Taking the reliability lewer level to be .90, the Algorithm 2.1 provides the optimal redundancy allocation (2,2,2,1) with over-all relibility equal to .92876, trade-off value 34.67, redundancy cost 18.30 and redundancy weight 41.0. Algorithm 2.2 derives the optimal solution (2,3,11), over-all reliability .901711, trade-off value 32.050, redundancy cost 17.20 and redundancy weight 37.00. Thus the later algorithm generates a cheaper trade-off solution.

Through the pages of this work, the problem of obtaining the optimal parallel redundancy allocation with minimum several cost functions under a reliability constraint has been treated using multiobjective techniques.

Considering the set of vectors with entries the values of the cost functions provided with the relation via some utility function, we have applied the trade-off technique replasing the objectives by their weighted average. Surely, we may use some other utility function of non-linear form. In this case Algorithms 2.1 and 2.2 cannot be applied and so we have to seeking for some other method depending on the utility function form.

In the other hand, in the "optimization in tandem", that is if we adopt the lexicographic relation, a sequence of single-objective function programs should be solved with the additional restraint to maintain the optimal solutions of the predecessor program. Tillman and Liittschwager (1967) method seems to be convenient for solving these programs.

Finally, in the most realistic situation of adopting the component to component relation, the same methodology may be followed but the difficulty of a large volume of calculations as well as some technical problems in cancelling the path leading to a particular minimal vector into the set of the vectors with entries the values of the objective functions arise.

```
0009          C     THIS PROGRAM GENERATES AN UNCOMPLETE FAMILLY OF UNDOMINATED ALLO-
0010          C     CATIONS FOR THE PROBLEM OF ACHIEVING A GIVEN RELIABILITY LEVEL
0011          C     WITH MINIMUM SEVERAL LINEAR COST FACTORS USING BARLOW-PROSCHAN
0012          C     METHOD
0013          C
0014          C     PROGRAMER: M,K,KONTESSIS
0015          C     DATE: 05/08/1975
0016          C     COMPUTER: ICL
0017          C     INSTALATION: UNIVERSITY OF SOUTHAMPTON COMPUTER CENTER
0018          C
0019                MASTER KONT
0020                DIMENSION C(15,5),W(5),Q(15),N(15),COST(5),REL(15),S(15),QUANT(15)
0021                INTEGER R
0022          C
0023                READ(5,100) K,R,BR
0024            100 FORMAT(2I5,F7.5)
0025                READ(5,105) (Q(I),I=1,K)
0026            105 FORMAT(6F7.5)
0027                READ(5,110) (W(J),J=1,R)
0028            110 FORMAT(2F7.5)
0029                READ(5,115) ((C(I,J),I=1,K),J=1,R)
0030            115 FORMAT(6F8.2)
0031                WRITE(6,116)
0032            116 FORMAT(1H1)
0033          C
0034                DO 5 I=1,K
0035              5 N(I)=0
0036             10 DO 15 J=1,R
```

```
0037                COST(J)=0,
0038                DO 15 I=1,K
0039             15 COST(J)=COST(J)+C(I,J)*(N(I)+1)
0040                DO 20 I=1,K
0041             20 REL(I)=1-Q(I)**(N(I)+1)
0042                SYSREL=1,
0043                DO 25 I=1,K
0044             25 SYSREL=SYSREL*REL(I)
0045                TOTCOS=0,
0046                DO 30 J=1,R
0047             30 TOTCOS=TOTCOS+W(J)*COST(J)
0048           C
0049                WRITE(6,120) (N(I),I=1,K)
0050            120 FORMAT(1H0,'ALLOCATION : ',4(I2,','))
0051                WRITE(6,130) SYSREL,TOTCOS,(J,COST(J),J=1,R)
0052            130 FORMAT(1H ,'RELIABILITY : ',F7.5/1H ,'TRADE-OFF FUNCTION : ',F8.2
0053               + /1H ,'COST-FACTORS : ',2('   C(',I2,') = ',F8.2))
0054           C
0055                IF(SYSREL.GE.BR) GO TO 50
0056                DO 40 I=1,K
0057                S(I)=0,
0058                DO 35 J=1,R
0059             35 S(I)=S(I)+W(J)*C(I,J)
0060             40 QUANT(I)=(1/S(I))*(ALOG(1-Q(I)**(N(I)+2))-ALOG(1-Q(I)**(N(I)+1)))
0061                XMAX=QUANT(1)
0062                IMAX=1
0063                DO 45 I=1,K
0064                IF(QUANT(I).LE.XMAX) GO TO 45
0065                XMAX =QUANT(I)
0066                IMAX=I
0067             45 CONTINUE
0068                N(IMAX)=N(IMAX)+1
0069                GO TO 10
0070             50 STOP
0071                END
```

```
ALLOCATION :  0, 0, 0, 0,
RELIABILITY : 0,35700
TRADE-OFF FUNCTION :     20,85
COST-FACTORS :     C( 1) =     11,40    C( 2) =     24,00

ALLOCATION :  0, 1, 0, 0,
RELIABILITY : 0,46410
TRADE-OFF FUNCTION :     24,43
COST-FACTORS :     C( 1) =     13,70    C( 2) =     28,00

ALLOCATION :  1, 1, 0, 0,
RELIABILITY : 0,55592
TRADE-OFF FUNCTION :     28,48
COST-FACTORS :     C( 1) =     14,90    C( 2) =     33,00

ALLOCATION :  1, 1, 1, 0,
RELIABILITY : 0,69615
TRADE-OFF FUNCTION :     35,32
COST-FACTORS :     C( 1) =     18,50    C( 2) =     41,00

ALLOCATION :  1, 1, 1, 1,
RELIABILITY : 0,80057
TRADE-OFF FUNCTION :     41,70
COST-FACTORS :     C( 1) =     22,80    C( 2) =     48,00

ALLOCATION :  1, 2, 1, 1,
RELIABILITY : 0,85600
TRADE-OFF FUNCTION :     45,27
COST-FACTORS :     C( 1) =     25,10    C( 2) =     52,00

ALLOCATION :  2, 2, 1, 1,
RELIABILITY : 0,88453
TRADE-OFF FUNCTION :     49,33
COST-FACTORS :     C( 1) =     26,30    C( 2) =     57,00

ALLOCATION :  2, 2, 2, 1,
RELIABILITY : 0,92876
TRADE-OFF FUNCTION :     56,17
COST-FACTORS :     C( 1) =     29,70    C( 2) =     65,00

ALLOCATION :  2, 3, 2, 1,
RELIABILITY : 0,94680
TRADE-OFF FUNCTION :     59,75
COST-FACTORS :     C( 1) =     32,00    C( 2) =     69,00
```

```
ALLOCATION :  2, 3, 2, 2,

RELIABILITY : 0,96532
TRADE-OFF FUNCTION :     66,12
COST-FACTORS :     C( 1) =     36,50   C( 2) =     76,00

ALLOCATION :  2, 3, 3, 2,
RELIABILITY : 0,97681
TRADE-OFF FUNCTION :     72,97
COST-FACTORS :     C( 1) =     39,90   C( 2) =     84,00

ALLOCATION :  2, 4, 3, 2,
RELIABILITY : 0,98240
TRADE-OFF FUNCTION :     76,55
COST-FACTORS :     C( 1) =     42,20   C( 2) =     88,00

ALLOCATION :  3, 4, 3, 2,
RELIABILITY : 0,98874
TRADE-OFF FUNCTION :     80,60
COST-FACTORS :     C( 1) =     43,40   C( 2) =     93,00

ALLOCATION :  3, 5, 3, 2,
RELIABILITY : 0,99042
TRADE-OFF FUNCTION :     84,17
COST-FACTORS :     C( 1) =     45,70   C( 2) =     97,00

ALLOCATION :  3, 5, 3, 3,
RELIABILITY : 0,99327
TRADE-OFF FUNCTION :     90,55
COST-FACTORS :     C( 1) =     50,20   C( 2) =    104,00

ALLOCATION :  3, 5, 4, 3,
RELIABILITY : 0,99619
TRADE-OFF FUNCTION :     97,40
COST-FACTORS :     C( 1) =     53,60   C( 2) =    112,00

ALLOCATION :  4, 5, 4, 3,
RELIABILITY : 0,99747
TRADE-OFF FUNCTION :    101,45
COST-FACTORS :     C( 1) =     54,80   C( 2) =    117,00

ALLOCATION :  4, 6, 4, 3,
RELIABILITY : 0,99798
TRADE-OFF FUNCTION :    105,03
COST-FACTORS :     C( 1) =     57,10   C( 2) =    121,00

ALLOCATION :  4, 6, 5, 3,
RELIABILITY : 0,99871
TRADE-OFF FUNCTION :    111,87
COST-FACTORS :     C( 1) =     60,50   C( 2) =    129,00
```

```
ALLOCATION :  4, 6, 5, 4,
RELIABILITY : 0,99914
TRADE-OFF FUNCTION :    118,25
COST-FACTORS :     C( 1) =     65,00    C( 2) =    136,00

ALLOCATION :  5, 6, 5, 4,
RELIABILITY : 0,99940
TRADE-OFF FUNCTION :    122,30
COST-FACTORS :     C( 1) =     66,20    C( 2) =    141,00

ALLOCATION :  5, 7, 5, 4,
RELIABILITY : 0,99955
TRADE-OFF FUNCTION :    125,87
COST-FACTORS :     C( 1) =     68,50    C( 2) =    145,00
```

```
0008          C     THIS PROGRAM GENERATES TWO FAMILLIES OF UNDOMINATED ALLOCATIONS
0009          C     ONE FOR EACH TWO-STAGE SUBSYSTEM 1-2 AND 3-4, USING KETTELLE S
0010          C     ALGORITHM BEING APPLIED TO A MULTIOBJECTIVE REDUNDANCY ALLOCATION,
0011          C     PROBLEM WITH A FOUR-STAGE SYSTEM.
0012          C
0013          C     PROGRAMER: M,K,KONTESSIS
0014          C     DATE 10/8/75
0015          C     COMPUTER: ICL
0016          C     INSTALATION: SOUTHAMPTON UNIVERSTITY COMPUTING CENTER
0017          C
0018                MASTER KONT1
0019                DIMENSION Q(4),W(2),C(4,2),CNEW(4),C1(8,8),C2(8,8),F(8,8),RS(8,8),
0020               +  NN1(8,8)
0021                INTEGER R
0022                READ(5,100) K,R
0023          100   FORMAT(2I5)
0024                READ(5,105) (Q(I),I=1,K)
0025          105   FORMAT(4F5.6)
0026                READ(5,110) (W(J),J=1,R)
0027          110   FORMAT(2F5.6)
0028                READ(5,115) ((C(I,J),I=1,K),J=1,R)
0029          115   FORMAT(4F5.3)
0030                INDEX=0
0031                I1=1
0032                I2=2
0033                DO 5 I=1,K
0034                CNEW(I)=0.
0035                DO 5 J=1,R
0036              5 CNEW(I)=CNEW(I)+C(I,J)*W(J)
```

```
0037              10 WRITE(6,120) I1,I2
0038             120 FORMAT(///1H ,'REDUNDANCY ALLOCATIONS FOR SUBSYSTEM ',I1,'=',I1)
0039                 DO 15 N1=1,8
0040                 DO 15 N2=1,8
0041                 C1(N1,N2)=(N1-1)*C(1,1)*(N2-1)*C(2,1)
0042                 C2(N1,N2)=(N1-1)*C(1,2)*(N2-1)*C(2,2)
0043                 F(N1,N2)=(N1-1)*CNEW(1)*(N2-1)*CNEW(2)
0044                 RS(N1,N2)=(1-Q(1)**N1)*(1-Q(2)**N2)
0045              15 NN1(N1,N2)=N1-1
0046                 DO 20 N1=1,8
0047                 WRITE(6,125) (NN1(N1,N2),N2=1,8)
0048             125 FORMAT(1HU,8(I1,8X))
0049                 WRITE(6,130)  (NN1(N2,N1),N2=1,8)
0050             130 FORMAT(1H+,8(2X,I1,6X))
0051                 WRITE(6,135)  (C1(N1,N2),N2=1,8)
0052                 WRITE(6,135)  (C2(N1,N2),N2=1,8)
0053                 WRITE(6,135)  (F(N1,N2),N2=1,8)
0054             135 FORMAT(1H ,8(F8.3,1X))
0055                 WRITE(6,140)  (RS(N1,N2),N2=1,8)
0056             140 FORMAT(1H ,8(F8.6,1X))
0057              20 CONTINUE
0058                 IF(INDEX.EQ.1) GO TO 30
0059                 INDEX=1
0060                 Q(1)=Q(3)
0061                 Q(2)=Q(4)
0062                 C(1,1)=C(3,1)
0063                 C(2,1)=C(4,1)
0064                 C(1,2)=C(3,2)
0065                 C(2,2)=C(4,2)
0066                 CNEW(1)=CNEW(3)
0067                 CNEW(2)=CNEW(4)
0068                 I1=3
0069                 I2=4
0070                 GO TO 10
0071              30 WRITE(6,145)
0072             145 FORMAT(1H1)
0073                 STOP
0074                 END
```

REDUNDANCY ALLOCATIONS FOR SUBSYSTEM 1=2

0 0	0 1	0 2	0 3	0 4	0 5	0 6	0 7
0,000	2,300	4,600	6,900	9,200	11,500	13,800	16,100
0,000	4,000	8,000	12,000	16,000	20,000	24,000	28,000
0,000	3,575	7,150	10,725	14,300	17,875	21,450	25,025
0,550000	0,726000	0,778400	0,793520	0,798056	0,799417	0,799825	0,799948

1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7
1,200	3,500	5,800	8,100	10,400	12,700	15,000	17,300
5,000	9,000	13,000	17,000	21,000	25,000	29,000	33,000
4,050	7,625	11,200	14,775	18,350	21,925	25,500	29,075
0,672000	0,873600	0,934080	0,956224	0,957667	0,959300	0,959790	0,959937

2 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7
2,400	4,700	7,000	9,300	11,600	13,900	16,200	18,500
10,000	14,000	18,000	22,000	26,000	30,000	34,000	38,000
8,100	11,675	15,250	18,825	22,400	25,975	29,550	33,125
0,694400	0,902720	0,965216	0,985965	0,989589	0,991277	0,991783	0,991935

3 0	3 1	3 2	3 3	3 4	3 5	3 6	3 7
3,600	5,900	8,200	10,500	12,800	15,100	17,400	19,700
15,000	19,000	23,000	27,000	31,000	35,000	39,000	43,000
12,150	15,725	19,300	22,875	26,450	30,025	33,600	37,175
0,698880	0,908544	0,971443	0,990313	0,995974	0,997672	0,998182	0,998334

4 0	4 1	4 2	4 3	4 4	4 5	4 6	4 7
4,800	7,100	9,400	11,700	14,000	16,300	18,600	20,900
20,000	24,000	28,000	32,000	36,000	40,000	44,000	48,000
16,200	19,775	23,350	26,925	30,500	34,075	37,650	41,225
0,699776	0,909709	0,972689	0,991583	0,997251	0,998951	0,999401	0,999614

5 0	5 1	5 2	5 3	5 4	5 5	5 6	5 7
6,000	8,300	10,600	12,900	15,200	17,500	19,800	22,100
25,000	29,000	33,000	37,000	41,000	45,000	49,000	53,000
20,250	23,825	27,400	30,975	34,550	38,125	41,700	45,275
0,699955	0,909942	0,972938	0,991837	0,997506	0,999207	0,999717	0,999870

6 0	6 1	6 2	6 3	6 4	6 5	6 6	6 7
7,200	9,500	11,800	14,100	16,400	18,700	21,000	23,300
30,000	34,000	38,000	42,000	46,000	50,000	54,000	58,000
24,300	27,875	31,450	35,025	38,600	42,175	45,750	49,325
0,699991	0,909988	0,972988	0,991887	0,997557	0,999258	0,999769	0,999922

7 0	7 1	7 2	7 3	7 4	7 5	7 6	7 7
8,400	10,700	13,000	15,300	17,600	19,900	22,200	24,500
35,000	39,000	43,000	47,000	51,000	55,000	59,000	63,000
28,350	31,925	35,500	39,075	42,650	46,225	49,800	53,375
0,699998	0,909998	0,972998	0,991897	0,997567	0,999268	0,999779	0,999932

REDUNDANCY ALLOCATIONS FOR SUBSYSTEM 3=4

0 0	0 1	0 2	0 3	0 4	0 5	0 6	0 7
0,000	4,500	9,000	13,500	18,000	22,500	27,000	31,500
0,000	7,000	14,000	21,000	28,000	35,000	42,000	49,000
0,000	6,375	12,750	19,125	25,500	31,875	38,250	44,625
0,637500	0,753125	0,747469	0,749620	0,749943	0,749991	0,749999	0,750000

1 0	1 1	1 2	1 3	1 4	1 5	1 6	1 7
3,400	7,900	12,400	16,900	21,400	25,900	30,400	34,900
8,000	15,000	22,000	29,000	36,000	43,000	50,000	57,000
6,850	13,225	19,600	25,975	32,350	38,725	45,100	51,475
0,796875	0,916406	0,934336	0,937025	0,937429	0,937489	0,937498	0,937500

2 0	2 1	2 2	2 3	2 4	2 5	2 6	2 7
6,800	11,300	15,800	20,300	24,800	29,300	33,800	38,300
16,000	23,000	30,000	37,000	44,000	51,000	58,000	65,000
13,700	20,075	26,450	32,825	39,200	45,575	51,950	58,325
0,836719	0,952227	0,981053	0,983877	0,984300	0,984364	0,984373	0,984375

3 0	3 1	3 2	3 3	3 4	3 5	3 6	3 7
10,200	14,700	19,200	23,700	28,200	32,700	37,200	41,700
24,000	31,000	38,000	45,000	52,000	59,000	66,000	73,000
20,550	26,925	33,300	39,675	46,050	52,425	58,800	65,175
0,846680	0,973682	0,992732	0,995589	0,996018	0,996082	0,996092	0,996093

4 0	4 1	4 2	4 3	4 4	4 5	4 6	4 7
13,600	18,100	22,600	27,100	31,600	36,100	40,600	45,100
32,000	39,000	46,000	53,000	60,000	67,000	74,000	81,000
27,400	33,775	40,150	46,525	52,900	59,275	65,650	72,025
0,849170	0,976545	0,995652	0,998518	0,998948	0,999012	0,999022	0,999023

5 0	5 1	5 2	5 3	5 4	5 5	5 6	5 7
17,000	21,500	26,000	30,500	35,000	39,500	44,000	48,500
40,000	47,000	54,000	61,000	68,000	75,000	82,000	89,000
34,250	40,625	47,000	53,375	59,750	66,125	72,500	78,875
0,849792	0,977261	0,996382	0,997250	0,999680	0,999744	0,999754	0,999756

6 0	6 1	6 2	6 3	6 4	6 5	6 6	6 7
20,400	24,900	29,400	33,900	38,400	42,900	47,400	51,900
48,000	55,000	62,000	69,000	76,000	83,000	90,000	97,000
41,100	47,475	53,850	60,225	66,600	72,975	79,350	85,725
0,849948	0,977440	0,996564	0,997433	0,999863	0,999928	0,999937	0,999939

7 0	7 1	7 2	7 3	7 4	7 5	7 6	7 7
23,800	28,300	32,800	37,300	41,800	46,300	50,800	55,300
56,000	63,000	70,000	77,000	84,000	91,000	98,000	105,000
47,950	54,325	60,700	67,075	73,450	79,825	86,200	92,575
0,849987	0,977485	0,996613	0,997478	0,999909	0,999973	0,999983	0,999984

```
C     THIS PROGRAM COMBINES THE TWO TWO-STAGE SUBSYSTEMS AND PROVIDES
C     THE FINAL FAMILLY OF UNDOMINATED ALLOCATIONS PRODUCED BY KETTELLE
C     S ALGORITHM BEING APPLIED TO A MULTIOBJECTIVE REDUNDANCY ALLOCA-
C     TION PROBLEM WITH A FOUR-STAGE SYSTEM
C
C     PROGRAMER: M.K.KONTESSIS
C     DATE: 15/5/75
C     COMPUTER: ICL
C     INSTALATION: SOUTHAMPTON UNIVERSITY COMPUTING CENTER
C
      MASTER KONT2
      DIMENSION N1(12),N2(12),N3(12),N4(12),C121(12),C122(12),C341(12),
     +  C342(12),F12(12),F34(12),R12(12),R34(12),C12341(12,12),
     +  C12342(12,12),F1234(12,12),R1234(12,12),
     + NN1(12,12),NN2(12,12),NN3(12,12),NN4(12,12)
      READ(5,100) (N1(L),N2(L),C121(L),C122(L),F12(L),R12(L),
     + N3(L),N4(L),C341(L),C342(L),F34(L),R34(L),L=1,12)
  100 FORMAT(2(6(I1,1X),4(F0.0)))
      WRITE(6,105)
  105 FORMAT(/////1H ,'REDUNDANCY ALLOCATIONS FOR THE SYSTEM 1-2-3-4')
      DO 5 L1=1,12
      DO 5 L2=1,12
      C12341(L1,L2)=C121(L1)+C341(L2)
      C12342(L1,L2)=C122(L1)+C342(L2)
      F1234(L1,L2)=F12(L1)+F34(L2)
      R1234(L1,L2)=R12(L1)*R34(L2)
      NN1(L1,L2)=N1(L1)
      NN2(L1,L2)=N2(L1)
      NN3(L1,L2)=N3(L1)
    5 NN4(L1,L2)=N4(L1)
      DO 10 L1=1,12
      WRITE(6,110) (NN1(L1,L2),L2=1,12)
  110 FORMAT(1HU,12(I1,8X))
      WRITE(6,115) (NN2(L1,L2),L2=1,12)
  115 FORMAT(1H+,12(2X,I1,6X))
      WRITE(6,120)(NN3(L2,L1),L2=1,12)
  120 FORMAT(1H+,12(4X,I1,4X))
      WRITE(6,125) (NN4(L2,L1),L2=1,12)
  125 FORMAT(1H+,12(6X,I1,2X))
      WRITE(6,130) (C12341(L1,L2),L2=1,12)
      WRITE(6,130) (C12342(L1,L2),L2=1,12)
      WRITE(6,130) (F1234(L1,L2),L2=1,12)
  130 FORMAT(1H ,12(F8.3,1X))
      WRITE(6,135) (R1234(L1,L2),L2=1,12)
  135 FORMAT(1H ,12(F8.6,1X))
   10 CONTINUE
      STOP
      END
```

REDUNDANCY ALLOCATIONS FOR THE SYSTEM 1-2-3-4

0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 0 1 2	0 0 2 1	0 0 2 2	0 0 2 3	0 0 3 2	0 0 3 3	0 0 4 2	0 0 4 3
0,000	4,500	3,400	7,900	12,400	11,300	15,800	20,300	19,200	23,700	22,600	27,100
0,000	7,000	6,000	15,000	22,000	23,000	30,000	37,000	38,000	45,000	46,000	53,000
0,000	6,375	6,850	13,225	19,600	20,075	26,450	32,825	35,300	39,675	60,150	66,525
0,337000	0,410550	0,446250	0,513187	0,523228	0,538847	0,549390	0,550971	0,555950	0,557550	0,557565	0,559171

0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	0 1 1 2	0 1 2 1	0 1 2 2	0 1 2 3	0 1 3 2	0 1 3 3	0 1 4 2	0 1 4 3
2,300	6,800	5,700	10,200	14,700	13,600	18,100	22,600	21,500	26,000	24,900	29,400
4,000	11,000	12,000	19,000	26,000	27,000	34,000	41,000	42,000	49,000	50,000	57,000
3,575	9,950	10,425	16,800	23,175	23,650	30,025	36,400	35,875	43,250	63,725	70,100
0,464100	0,533715	0,580125	0,667144	0,680197	0,700501	0,714207	0,716262	0,722709	0,724789	0,724835	0,726895

0 2 0 0	0 2 0 1	0 2 1 0	0 2 1 1	0 2 1 2	0 2 2 1	0 2 2 2	0 2 2 3	0 2 3 2	0 2 3 3	0 2 4 2	0 2 4 3
4,600	9,100	8,000	12,500	17,000	15,900	20,400	24,900	23,800	28,300	27,200	31,700
8,000	15,000	16,000	23,000	30,000	31,000	38,000	45,000	46,000	53,000	54,000	61,000
7,150	13,525	14,000	20,375	26,750	27,225	33,600	39,975	40,450	46,825	67,300	53,675
0,695230	0,770665	0,820285	0,715330	0,727287	0,748997	0,763652	0,765850	0,772743	0,774966	0,775016	0,777214

1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 0 1 2	1 1 2 1	1 1 2 2	1 1 2 3	1 1 3 2	1 1 3 3	1 1 4 2	1 1 4 3
3,500	8,000	6,900	11,400	15,900	14,800	19,300	23,800	22,700	27,200	26,100	30,600
9,000	16,000	17,000	24,000	31,000	32,000	39,000	46,000	47,000	54,000	55,000	62,000
7,625	14,000	14,475	20,850	27,225	27,700	34,075	40,450	40,925	47,300	67,775	54,150
0,556920	0,640458	0,696150	0,600572	0,810236	0,840602	0,857048	0,859515	0,867251	0,869747	0,869802	0,872298

2 0 0 0	1 4 0 1	1 2 1 0	1 2 1 1	1 2 1 2	1 2 2 1	1 2 2 2	1 2 2 3	1 2 3 2	1 2 3 3	1 2 4 2	1 2 4 3
5,800	10,300	9,200	13,700	18,200	17,100	21,600	26,100	25,000	29,500	28,400	32,900
13,000	20,000	21,000	28,000	35,000	36,000	43,000	50,000	51,000	58,000	57,000	66,000
11,200	17,575	18,050	24,425	30,800	31,275	37,650	44,025	44,500	50,875	51,350	57,725
0,595476	0,684797	0,744345	0,853937	0,872745	0,898797	0,916382	0,919020	0,927291	0,929960	0,930019	0,932694

3 0 0 0	3 0 0 1	1 3 1 0	1 3 1 1	1 3 1 2	1 3 2 1	1 3 2 2	1 3 2 3	1 3 3 2	1 3 3 3	1 3 4 2	1 3 4 3
8,100	12,600	11,500	16,000	20,500	19,400	23,900	28,400	27,300	31,800	30,700	35,200
17,000	24,000	25,000	32,000	39,000	40,000	47,000	54,000	55,000	62,000	63,000	70,000
14,775	21,150	21,625	28,000	34,375	34,850	41,225	47,600	48,075	54,450	54,925	61,300
0,607045	0,698099	0,758804	0,876624	0,889697	0,916256	0,934182	0,936871	0,945303	0,948024	0,948084	0,950811

2 0 0 0	2 4 0 1	2 2 1 0	2 2 1 1	2 2 1 2	2 2 2 1	2 2 2 2	2 2 2 3	2 2 3 2	2 2 3 3	2 2 4 2	2 2 4 3
7,000	11,500	10,400	14,900	19,400	18,300	22,800	27,300	26,200	30,700	29,600	34,100
18,000	25,000	26,000	33,000	40,000	41,000	48,000	55,000	56,000	63,000	64,000	71,000
15,250	21,625	22,100	28,475	34,850	35,325	41,700	48,075	48,550	54,925	55,400	61,775
0,615329	0,707624	0,769156	0,884530	0,901836	0,928757	0,946928	0,949654	0,958201	0,960958	0,961017	0,963786

3 0 0 0	2 3 0 1	2 3 1 0	2 3 1 1	2 3 1 2	2 3 2 1	2 3 2 2	2 3 2 3	2 3 3 2	2 3 3 3	2 3 4 2	2 3 4 3
9,500	13,800	12,700	17,200	21,700	20,600	25,100	29,600	28,500	33,000	31,900	36,400
22,000	29,000	30,000	37,000	44,000	45,000	52,000	59,000	60,000	67,000	68,000	75,000
19,325	25,200	25,675	32,050	38,425	38,900	45,275	51,650	52,125	58,500	58,975	65,350
0,627276	0,721369	0,784097	0,901711	0,919354	0,946798	0,965322	0,968101	0,976816	0,979625	0,979687	0,982507

4 0 0 0	2 4 0 1	2 4 1 0	2 4 1 1	2 4 1 2	2 4 2 1	2 4 2 2	2 4 2 3	2 4 3 2	2 4 3 3	2 4 4 2	2 4 4 3
11,600	16,100	15,000	19,500	24,000	22,900	27,400	31,900	30,800	35,300	34,200	38,700
26,000	33,000	34,000	41,000	48,000	49,000	56,000	63,000	64,000	71,000	72,000	79,000
22,400	28,775	29,250	35,625	42,000	42,475	48,850	55,225	55,700	62,075	62,550	68,925
0,630865	0,725492	0,788579	0,905865	0,924609	0,952209	0,970859	0,973634	0,982397	0,985226	0,985286	0,988122

3 0 0 0	3 3 0 1	3 3 1 0	3 3 1 1	3 3 1 2	3 3 2 1	3 3 2 2	3 3 2 3	3 3 3 2	3 3 3 3	3 3 4 2	3 3 4 3
10,500	15,000	13,900	18,400	22,900	21,800	26,300	30,800	29,700	34,200	33,100	37,600
27,000	34,000	35,000	42,000	49,000	50,000	57,000	64,000	65,000	72,000	73,000	80,000
22,875	29,250	29,725	36,100	42,475	42,950	49,325	55,700	55,175	62,550	63,025	69,400
0,631325	0,726025	0,789156	0,907529	0,925285	0,952906	0,971550	0,974346	0,983115	0,985945	0,986007	0,988845

5 0 0 0	3 5 0 1	3 5 1 0	3 5 1 1	3 5 1 2	3 5 2 1	3 5 2 2	3 5 2 3	3 5 3 2	3 5 3 3	3 5 4 2	3 5 4 3
13,900	18,400	17,300	21,800	26,300	25,200	29,700	34,200	33,100	37,600	36,500	41,000
30,000	37,000	38,000	45,000	52,000	53,000	60,000	67,000	68,000	75,000	76,000	83,000
25,975	32,350	32,825	39,200	45,575	46,050	52,425	58,800	59,275	65,650	66,125	72,500
0,631959	0,726750	0,789924	0,908412	0,926186	0,953833	0,972495	0,975275	0,984072	0,986906	0,986967	0,989800

6 0 0 0	3 6 0 1	3 6 1 0	3 6 1 1	3 6 1 2	3 6 2 1	3 6 2 2	3 6 2 3	3 6 3 2	3 6 3 3	3 6 4 2	3 6 4 3
12,800	17,300	16,200	20,700	25,200	24,100	28,600	33,100	32,000	36,500	35,400	39,900
31,000	38,000	39,000	46,000	53,000	54,000	61,000	68,000	69,000	76,000	77,000	84,000
26,650	32,825	33,300	39,675	46,050	46,525	52,900	59,275	59,750	66,125	66,600	72,975
0,634933	0,730173	0,793607	0,911717	0,930574	0,958353	0,977103	0,979916	0,988735	0,991581	0,991644	0,994491

SUMMARY

The present work deals with the problem of allocating to te stages of a system spare components in parallel in order to improve the reliability of this system under several cost-functions.

This problem can be regarded directly, that is to maximize the over-all reliability under several cost-function constraints (say system cost, system weight, system volume e.t.c.). Several papers have been written on this problem using various mathematical programming methods. But also, of interest could be the inverse problem which is to maintain the over-all system reliability over some safety level and it must be done with the cheapest several cost-functions. This topic has been treated by Kettelle (1962) and Barlow and Proschan (1965) in the case of a single cost-function but multiobjective programming techniques are involved in order to describe and solve the same problem in the case of multiple cost-functions to be simultaneously minimized.

What the work on hand does is that it considers a systematical multiobjective formulation of this inverse redundancy allocation problem based on various relations introduced in the Cartesian space of the cost-functions, and then solves this problem using the "trade-off" technique.

REFERENCE

1. *Barlow* and *Proschan,* Mathematical Theory of Reliability, J. Wiley, 1965.
2. *Bazovsky, I.,* Reliability Theory and Practice, Prentice Hall, 1961.
3. *Bellman* and *Dreyfus,* Dynamic Programming an the Reliability of Multicomponent Devices, O.R. (USA), 1958.
4. *Bellman* and *Dreyfus,* Applied Dynamic Programming, Princeton Univ. Press, 1962.
5. *Black* and *Proschan,* On Optimal Redundancy, O.R. (USA), 1959.
6. *Bod, P.,* Programmation Lineaire dans le cas de plusieurs fonctionsobjectif donnes simultanement, Publ. Math. Inst. Hungary Acad. Sci. (Series B), 8, 1963.
7. *Eilon, S.,* Goals and Constraints in Decision Making, O.R. Quarterly, 1972.
8. *Fedorowitcz* and *Mazumdar,* Use of Geometric Programming to Maximize Reliability achieved by Redundancy, O.R. (USA), 1968.
9. *Fyffe* and *Hines* and *Lee,* System Reliability Allocation and Computational Algorithm, IEEE Trans. on Reliability, R-17,2, June 1968.
10. *Gates,* The Reliability of Redundant Systems, Memorandum No 20-76, Jet Propulsion Laboratory, CIT, August 27, 1952.
11. *Geisler* and *Karr,* The Design of Military Supply Tables for Spare Components, O.R. (USA), 1956.
12. *Ghare* and *Taylor,* Optimal Redundancy for Reliability in Series Systems, O.R. (USA), 1969.
13. *Gordon, R.,* Optimum Component Redundancy for Maximum System Reliability, O.R. (USA), 1957.
14. *Gourary, M.,* An Optimum Allowance List Model, Naval Research Logistics Quarterly, 1956.
15. *Gourary, M.,* A simple Rule for the consolidation of Allowance Lists, Naval Research Logistics Quarterly, 1958.
16. *Hees* and *Meerendonk,* Optimum Reliability of Parallel Multicomponent Systems, O.R. Quarterly, 1961.
17. *Hendrix* and *Stedry,* The Elementary Redundancy Optimization Problem: A case study in Probabilistic Multigoal Programming, O.R. (USA), 1973.
18. *Jensen, P.,* Optimization of Series-Parallel-Series Networks, O.R. (USA), 1970.
19. *Kaufmann,* Reliability; A Mathematical Approach, Transworld Student Library Series, 1972.
20. *Kaufmann, Cryon* and *Grouchko,* La Gestion Scientifique des equipments: Lafiabilite, Vol. I, Dunod 1969.
21. *Kettelle, J.,* Least-Cost Allocation of Reliability Investment, O.R. (USA), 1962.
22. *Liittschwager, J.,* Dynamic Programming in the solution of a Multistage Reliability Problem, Journal of Industrial Engineering, 1964.

23. *Lloyd,* Reliability: Management, Methods and Mathematics, Prentice Hall, 1962.
24. *Lusser,* Reliability of Guided Missiles, Redstone Arsenal, 1954.
25. *McLeavy,* Numerical Investigation of Optimal Parallel Redundancy, O.R. (USA), 1974.
26. *Mine,* Reliability of Physical System, IRE Trans. PGIT Circuit Theory Special Supplement, 1959.
27. *Mizukami,* Optimum Redundancy for maximum System Reliability by the method of Convex and Integer Programming, O.R. (USA), 1968.
28. *Morrison,* The Optimum Allocation of spares Components in Systems, Technometrics, 1961.
29. *Moskowitz* and *McLean,* Some Reliability Aspects of System Design, IRE Trans. of Rel. and Quality Control, 1956.
30. *Proschan* and *Bray,* Optimum Redundancy under Multiple Constraints, O.R. (USA), 1965.
31. *Rudeanu, S.,* Programmation Bivalent a plusieurs fonctions Economiques, Revue Francais Informatique et Research Operationelle, 1969.
32. *Selman* and *Grisamore,* Optimum System Analysis by Linear Programming, Proc. of Annual Symposium on Reliab. of American Soc. for Quality Control, 1966.
33. *Tillman* and *Liittschwager,* Integer Programming Formulation of Constrained Reliability Problems, Management Science, 1967.
34. *Von Alvin* (editor), Relibility Engineering, Prentice Hall, 1964.
35. *Zelen* (editor), Statistical Theory of Reliability, Madison, 1964.

CONTENTS