LEAST-COST ALLOCATIONS OF RELIABILITY INVESTMENT

John D. Kettelle, Jr.

Kettelle & Wagner, Paoli, Pennsylvania

(Received February 24, 1961)

This paper presents two complementary techniques for determining optimal allocation of reliability investment in a multi-stage system. The first is a step-wise dynamic programming algorithm, which has proved to be a simple and direct means for obtaining the exact solution to the basic problem of least-cost allocation of redundancy. The second technique is an explicit solution to the investment allocation problem if the unreliability of each stage decreases exponentially (and continuously) as its cost increases. This solution is based on an inequality, which for reasonably reliable systems justifies minimizing the sum of the stage unreliabilities instead of the system unreliability.

DYNAMIC PROGRAMMING ALGORITHM

THIS SECTION presents a simple algorithm which has provided an exact solution to a problem of redundancy allocation in a multi-stage system. It is a direct application of dynamic programming, with a few additional aspects to organize and speed up the calculations. The problems to which it has so far been applied have been nonrepetitive analyses of the major components of large radar, communications, and data-processing systems, and have been conveniently worked out by hand as described below. However, a flow chart is included as an alternative description of the algorithm. The last part of the section proves the validity of the algorithm under rather general conditions, and discusses other applications.

Statement of the Redundancy Allocation Problem

An M-stage system is specified, which is operational only if each (series) stage has at least one operational component from among possibly more than one in parallel. A component of stage i costs c_i and has availability $a_i (i=1, \cdots, M)$. Availability is defined here as the probability (constant over time) a component or system is operational at a point in time. Failure behavior, and repair behavior, among all components are assumed independent, so that system availability is

$$A = \prod_{i=1}^{i=M} [1-(1-a_i)^{n_i}],$$

where n_i is the number of parallel components provided at stage i. The cost of this configuration is

$$C = \sum_{i=1}^{i=M} c_i n_i.$$

Given a system availability requirement R, the problem is to determine a least-cost configuration [i.e., $(n_1, \cdots, n_M)$] that yields $A \geqq R$.

As an example, consider the following four-stage system, with a system availability goal of 0.99:

Stage	1	2	3	4
Component cost	1.2	2.3	3.4	4.5
Component availability	0.8	0.7	0.75	0.85

This example will be carried forward through this section, with extra indentions to illustrate the steps of the algorithm.

Dominating Sequences

For a given group of redundancy configurations, defined over a given set of stages, an availability-cost sequence $[(A_0,C_0), (A_1,C_1), \cdots]$ is defined to be a *dominating sequence* if (A_i,C_i) for $i=1, 2, \cdots$ is a cheapest entry in the group with reliability exceeding A_{i-1}. In a similar spirit, one configuration is said to *dominate* another if it has either (a) more availability and no more cost, or (b) no less availability and less cost. Note that a dominating sequence contains only configurations that are undominated. The dominating sequence is essentially the *optimal policy* of BELLMAN[1] and the (complete) optimal n^* curve of BLACK AND PROSCHAN.[2] The dominating sequence for the entire system, for all possible redundancy configurations, is simply the system reliability-cost curve (assuming optimal allocations).

The given reliability-cost sequence for each individual stage is (trivially) dominating for the stage. The algorithm described below constructs dominating sequences for successively larger groups of stages until the entire system is included. The following three sections present in order:

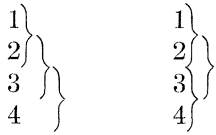
(1) A version of the algorithm suitable for hand computation, (2) A computer flow chart, (3) General conditions under which the algorithm is valid, and further applications.

Steps for Hand Computation

The successive steps of the procedure are presented below, with application to the particular example set off by an extra indention.

(1) Plan successive pairings of stages or groups of stages, each new combination consisting of two previous stages or combinations. Continue until the entire system has been combined.

The following indicate two alternate pairings of the four-stage system —the one on the right will be used in this example:



There will always be $M-1$ such pairings. Although it is not clear precisely how much can be gained by ingenious pairings, simplifications will arise if both members in the pair have equal steps in reliability or cost.

(2) For each stage in the system, determine the minimum number of components required to attain the availability R at that stage alone. These minimum requirements for each stage are called 'base' requirements. The rest of the computation will concentrate on *additional* costs and *additional* requirements above these base requirements. In particular, only configurations with availability greater than R will be considered in the dominating sequences.

If there is any interest in the possibility of saving money by investigating *lower* requirements (than the R originally specified), those should be specified *at this time*—if they are not introduced until later, the whole problem may have to be reworked.

The total allowable unavailability for the sample system is only 0.01, so the base requirement and resulting costs and unavailabilities for the four stages are as follows:

Stage	1	2	3	4
Basic number of components required	3	4	4	3
Basic cost	3.6	9.2	13.6	13.5
Basic stage unavailability	0.0080	0.0081	0.0039	0.0034

(3) For the first pair of stages, prepare a table similar to Fig. 1. Across the top, post the costs c_{1j} and the unavailabilities $b_{1j}(b_{ij}=1-a_{ij})$ corresponding to successive additions of components above the base requirement—i.e., the first b_{1j} posted should be less than $Q(Q=1-R)$.

In our example (Fig. 1 combines the first two stages) the first stage has a component unavailability of 0.2, a unit cost of 1.2, and a base requirement of 3 components, so

$$c_{1j} = (3+j)1.2,$$

and

$$b_{1j} = (0.2)^{3+j}. \qquad (j=0, 1, 2, \cdots)$$

Post successive steps along the top until the stage unavailability promises to be low enough to support the system requirement (Q/M might

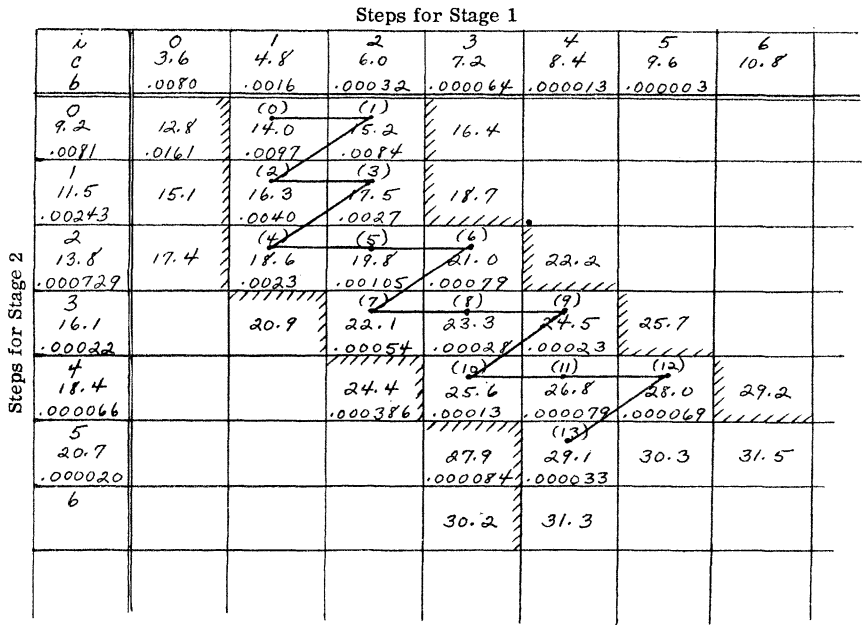


Fig. 1. Combinations of stages 1 and 2; i number of step, c cost above basic cost, b unavailablity; ( ) indicate step for *next* level.

be a first guess). If subsequent calculations require it, the present table can be conveniently extended later.

Post stage 2 in a similar fashion down the left-hand side of the figure.

(4) To construct the dominating sequence for the first two stages, begin with the entry in the upper left corner. This will be the first term (except its unavailability may not necessarily be less than the system allowance Q), since it is clearly the cheapest entry. Compute and post the costs and the unavailabilities of other entries only as needed below.

For the present example, the cost C_{ij} of the combination and the

unavailability B_{ij} achieved by the combination at row i and column j are given by

$$C_{ij}=c_{2i}+c_{1j},$$

and

$$B_{ij}=b_{2i}+b_{1j}-b_{2i}\cdot b_{1j}.$$

If b_{2i} and b_{1j} are both small, the product term can usually be disregarded. An estimate of the possible cumulative effect of repeating this approximation all through the algorithm is provided by the inequality reported in a later section.

If an element E_{ij}, at the intersection of row i and column j, is an element of the dominating sequence, then search for the next element may be restricted to the rows $r \leqq i$ and the columns $s \leqq j$. From each such row or column the candidate for the next member of the sequence is the cheapest element (in that row or column) with less unavailability. The cheapest of all these candidates becomes the next element.

Often the remainder of an entire row or column can be rejected. In particular, if the unavailability B_{ij} is less than the unavailability b_{2k} posted at the left of a previous row k, then all entries in row k which cost more than C_{ij} can be rejected. Note that C_{kn} is monotone increasing in n, and that $B_{kn} \geqq b_{2k}$ for $n=0, 1, 2, \cdots$. A symmetric remark holds for eliminating the lower portions of columns.

In Fig. 1, the boundary of the rejected combinations has been indicated by shading. Each rejected combination is dominated by a combination that has not been rejected.

(5) Number in order of cost, starting with 0, the entries that cannot be dominated and whose unavailability is less than the system allowance Q. Continue until it appears that the dominating sequence has progressed to an availability that will eventually support the system requirement. (A first guess for such a limit may be Q/M. In subsequent figures, a better guess may be Q/S, where S is the number of subsystems that will be combined with the subsystem developed by the figure.) If in developing subsequent figures it appears that the present figure has not been carried out far enough, it can still be extended with no wasted effort.

In Fig. 1, the sequence has been extended (more than actually necessary) to 13 terms, for the last of which the unavailability is 0.000033, and the cost is 29.1. Figure 2 displays the calculations for stages 3 and 4.

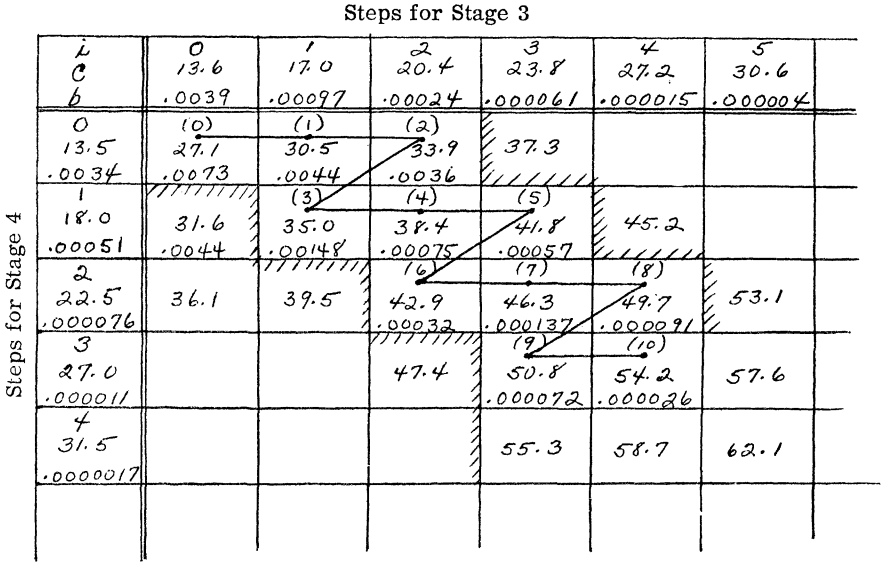


Fig. 2. Combinations of stages 3 and 4; i number of step, c cost above basic cost, b unavailability; ( ) indicate step for *next* level.

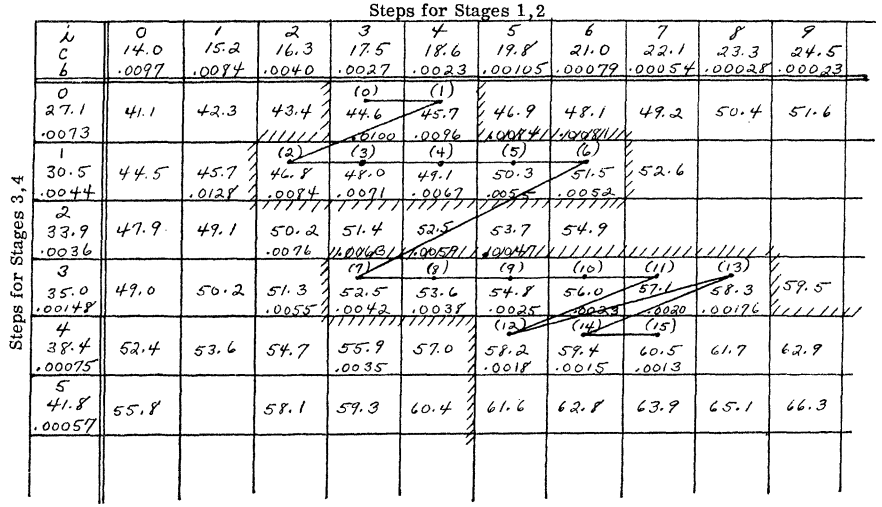


Fig. 3. Combinations of stages 1, 2, 3, and 4; i number of step, c cost above basic cost, b unavailability; ( ) indicate step for *next* level.

(6) Proceed to the next level of combinations, using the availability and cost step functions represented by the previous dominating sequences as the inputs.

In the example, Fig. 3 gives the calculations for the second (and last) level of combinations. Note that in this level the increases in cost and the gains in availability are both quite irregular. If, at the *first* level, namely at the individual stages, the cost increases had been

TABLE I

DOMINATING SEQUENCE FOR STAGES 1, 2, 3, 4

	Cost	Unavailability	Components			
			Stage 1	Stage 2	Stage 3	Stage 4
0	44.6	0.0100	5	5	4	3
1	45.7	0.0096	4	6	4	3
2	46.8	0.0084	4	5	5	3
3	48.0	0.0071	5	5	5	3
4	49.1	0.0067	4	6	5	3
5	50.3	0.0055	5	6	5	3
6	51.5	0.0052	6	6	5	3
7	52.5	0.0042	5	5	5	4
8	53.6	0.0038	4	6	5	4
9	54.8	0.0025	5	6	5	4
10	56.0	0.0023	6	6	5	4
11	57.1	0.0020	5	7	5	4
12	58.2	0.0018	5	6	6	4
13	58.3	0.00176	6	7	5	4
14	59.4	0.0015	6	6	6	4
15	60.5	0.0013	5	7	6	4

irregular, the method would have worked equally well. In these figures, c_i and b_i are used to denote the cost and unavailability of the ith member of the dominating sequence for the group of stages.

Continue through successive levels until all stages have been combined. The resulting dominating sequence represents the progressively more expensive (but more reliable) optimal configurations—its first element that meets the availability goal is the solution. The availability-cost function corresponding to optimal allocations for the system, and the corresponding configurations themselves (not necessarily unique) may also be tabulated.

It will be noted that often a dominating sequence contains a relatively inefficient step (*see*, for example, step 2 in Fig. 2, in which the unavailability decreases only from 0.0044 to 0.0036 at an increase in cost from 30.5 to 33.9). These inefficient steps are often completely by-passed at the next level of computation (as was the example mentioned in the previous parentheses). A criterion by which individual steps may be

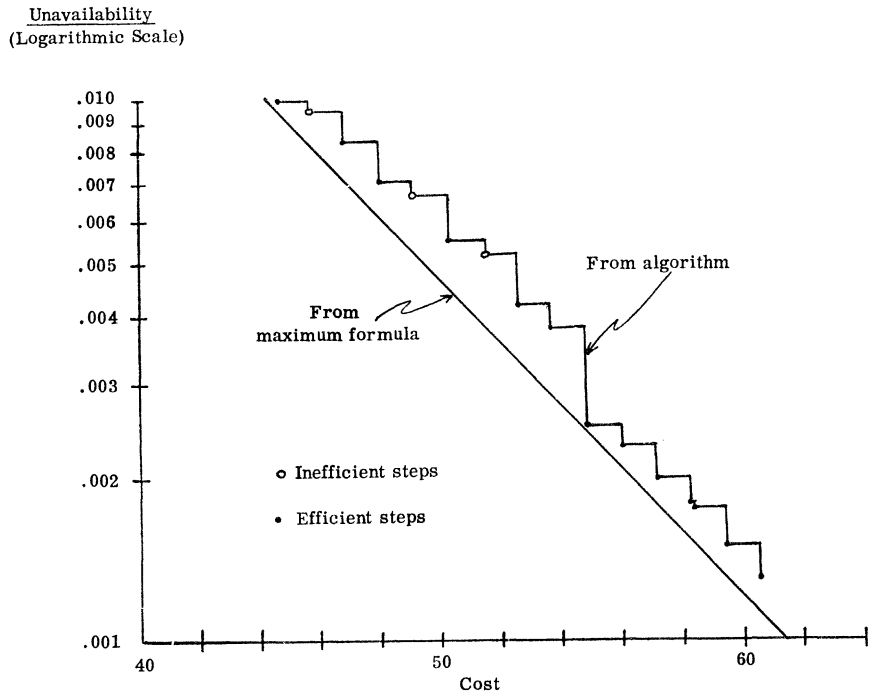


Fig. 4. Unavailability versus system investment for sample system.

judged, and inefficient ones immediately discarded, is presented in the next section.

The dominating sequence for the entire system in the sample problem (as derived in Fig. 3) is posted, together with the corresponding configuration, in Table I. The resulting availability-cost (step) function is plotted in Fig. 4.

Computer Flow Chart

Figure 5 presents a general flow chart implementing essentially the same computations as are performed in the hand computation. It utilizes the following notation:

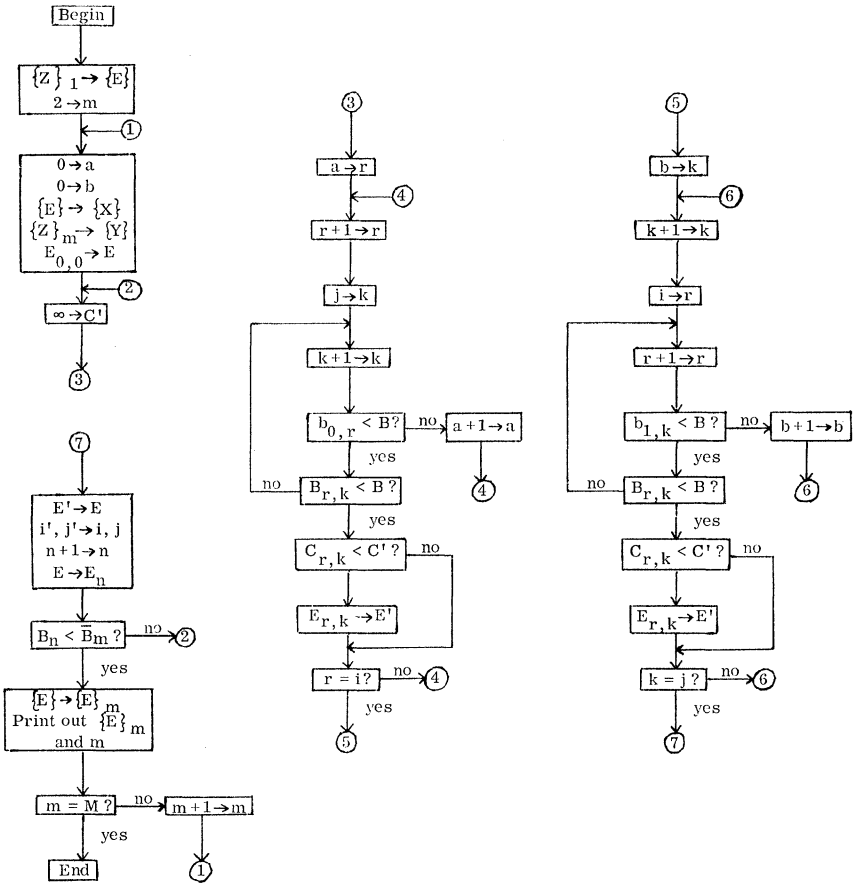


Fig. 5. Computer flow chart for allocation algorithm.

Inputs:

$\{Z\}_m$ = Sequence of unavailability-cost pairs for stage m.
M = Total number of stages.
$\overline{B}_m$ = Lower limit for computing combined unavailability for first m stages.

Outputs:

$\{E\}_m$ = Dominating sequence of unavailability-cost pairs for combination of first m stages.

Internal:

$\{X\}$ = Dominating sequence $\{(b_{0,0}\, c_{0,0}), (b_{0,1}\, c_{0,1}), \cdots\}$ of previous stages (column headings).

$\{Y\}$ = Unavailability-cost pairs $\{(b_{1,0}\, c_{1,0}),\ (b_{1,1}\, c_{1,1}),\ \cdots\}$ of next stage to be added (row headings).

$\{E\}$ = Dominating sequence generated from $\{X\}$ and $\{Y\}$.

a = Number of rejected rows.

b = Number of rejected columns.

$(B_{r,k},C_{r,k})$ = Unavailability-cost pair from combining X_r with Y_k.

E = Last element $(B,C) = (B_{i,j},C_{i,j})$ of dominating sequence.

E' = Candidate $(B',C') = (B_{i',j'},C_{i',j'})$ for next element of dominating sequence.

Generalization, Validity, and Further Applications

This section proves the basic validity of the algorithm under somewhat general conditions, and describes briefly some additional applications.

Validity. Basic to the validity of this algorithm is the guarantee that one may restrict attention to dominating sequences for matrices—that as later stages are added, there is no previously-rejected combination of earlier stages that might somehow fit better with new ones.

As a generalization from the reliability application, assume that

1. if a and a' are the payoffs for two disjoint combinations of stages, then they determine a unique combined payoff $f(a,a')$.

2. f is monotone in the sense that

$$a_1 \geqq a_2 \text{ and } a_1' \geqq a_2' \text{ implies } f(a_1,a_1') \geqq f(a_2,a_2')$$

for all payoffs a_1,a_2,a_1',a_2'. Then the following theorem furnishes the desired guarantee:

THEOREM: *Suppose that (A,C) is the payoff-cost pair of a configuration of the combined stages of the disjoint subsystems F and F'. Then (A,C) either equals or is dominated by some term of the dominating sequence of the matrix formed from the dominating sequences of F and F'.*

Proof. For an inductive proof, assume that the theorem has already been proved for the subsystems (and associated matrices) denoted by F and F'. If F (or F') is a single stage, then its dominating sequence is simply its *entire* sequence of payoff-cost pairs, so that the first step of the induction is trivial.

The configuration represented by (A,C) is the series combination of a configuration of F and a configuration of F'; let the payoff-cost pairs of the latter configurations be (a,c) and (a',c') respectively. Let (a_j,c_j) and $(a_i',\ c_i')$ be terms, whose existence is guaranteed by inductive hypothesis, in the respective dominating sequences for F and F' such that (a,c) and (a',c') respectively equal or are dominated by (a_j,c_j) and (a_i',c_i'). Then by condition **2** on f

$$A_{ij} = f(a_i',a_j) \geqq f(a',a) = A.$$

But (a,c) is dominated by or equals (a_j,c_j) and (a',c') is dominated by or equals (a_i',c_i'), so that

$$c_j \leqq c \quad \text{and} \quad c_i' \leqq c'.$$

Finally,

$$C_{ij} = c_j + c_i' \leqq c + c' = C.$$

Thus (A,C) is dominated by or equals (A_{ij},C_{ij}), which is an entry in the matrix.

It remains to prove that (A_{ij},C_{ij}) is dominated by or equals a term in the dominating sequence of the matrix. Let $E_m = (A_m,C_m)$ be the first term of the dominating sequence whose cost exceeds C_{ij}; so that $C_{m-1} \leqq C_{ij}$. Then it follows from the definition of the dominating sequence that $A_{m-1} \geqq A_{ij}$. Therefore (A_{ij},C_{ij}) is dominated by or equals E_{m-1}, and the proof is completed.

There remains the question of feasibility—whether the algorithm can indeed generate a dominating sequence leading to a payoff-cost pair of interest in a finite (hopefully rather small) number of steps. Here the basic requirement is that

3. for each individual stage, the sequence of improvements has unbounded cost.

This prevents the algorithm from getting trapped in a single row or column with a never-ending sequence of trivial improvements. It is also useful if reasonably low upper bounds for the payoff function in each row (or column) of the matrix are available. One simple such bound derives from the condition

4. $f(a,a') \leqq a$.

Conditions **1** through **4** are indeed used directly in both the hand and the machine versions of the algorithm, and are sufficient to guarantee their termination in a finite number of steps. Practical success and efficiency depend naturally on the size of the proposed single-stage improvements —for example, carrying a long string of very small improvements through various successive combinations can be annoying. However, the reliability problems so far submitted (involving up to eight stages) to the algorithm have typically been solvable by hand in a few hours.

Discussion of applications. This section describes a few additional applications that have been tried for the algorithm.

Reliability. Other reliability problems to which the algorithm has been applied include multi-channel availability and switching.

(1) *Multi-channel availability.* If a system requires at least m parallel channels (instead of just one as considered in the text), to be operable, the generalized

algorithm may be applied directly. On the other hand, if there is some payoff for one channel and merely *additional* payoff for more than one, then conditions **1**, **2**, and **3**, are, in general, not met.

(2) *Switching.* A three-dimensional version of the algorithm matrix may be used to account for switching between adjacent stages. The joint availability of two stages depends, in general, on the availability of each stage plus certain characteristics (e.g., availability and reaction time) of the switching system that is provided between them. Also, the cost of the switching depends in part on the number of elements in each of the adjacent stages. Thus one would consider investments in stage 1 parallel to one axis, investments in stage 2 parallel to the second, and investments in switches parallel to the third. This three-dimensional algorithm may then be iterated for combinations of subsystems, with the third axis always representing the investment in the switches joining the two subsystems under consideration.

Sprinkler system design. This illustration is intended to suggest the versatility of the algorithm.

A simplified sprinkler system may be regarded as composed of a series of three prefabricated pipes (and identical nozzles) of progressively smaller cross-sections. In the event of a fire, the underwriters assume all the nozzles will be triggered, and require that the gallonage (gallons per minute) delivered by each be at least a specified amount. The gallonage delivered at a nozzle depends only on the dynamic pressure at the nozzle; the pressure drop in the next (inner) section of pipe depends on its dimensions and on the gallonage. The problem is to design a least-cost sprinkler system, including the costly water tower to provide the necessary pressure, that meets underwriter requirements.

The generalized version of the algorithm may be applied, where the payoff function at each stage is the pressure. (As a minor complication, the resultant gallonage totals must also be kept track of.) Complications associated with branches in the pipe are handled with no essential difficulties.

ALLOCATION WITH (CONTINUOUS) EXPONENTIAL STAGE AVAILABILITY

This section proves that if for each stage of a system the unavailability decreases exponentially (and continuously) with investment then so does the unavailability for a properly allocated system. In particular, a formula for maximum system availability (as a function of cost) is derived, together with a rule for attaining it. Since this exponential behavior may be regarded as a continuous approximation to the step-wise stage response to redundancy, this formula affords a check on the calculations of the preceding algorithm. In support of the formula, a rather powerful inequality is presented to justify minimizing the *sum* of the stage unavailabilities.

Maximum Availability Formula

The theorem below establishes a formula for the minimum *sum* of stage unavailabilities.

THEOREM: *For an n-stage system, make the following assumptions, for* $i=1, \cdots, n$:

(*1*) *The unavailability* B_i *of stage* i, *if* y *dollars are invested in it, is*

$$B_i(y)=b_i^{y/c_i}.$$

(*2*) *For a given system investment* x, $\beta(x)$ *is the minimum sum of stage unavailabilities attainable by distributing* x *among the stages, i.e.,*

$$\beta(x)=\min_{\Sigma x_i=x}\sum_{i=1}^{i=n} b_i^{x_i/c_i}.$$

Then there exist numbers X, D, *and* γ, *such that*

$$\beta(x)=D\,e^{x/\gamma} \quad \textit{for } x>X.$$

Moreover, X, D, *and* γ *may be calculated by the following formulas*:

$$X=\max\{\max_{k=1,\cdots,n-1}[\omega_{k+1}\log t_k],\max_{k=1,\cdots,n-1}[-\gamma_k\log t_k]\}, \textit{ for } n>1,$$

$X=0$ *for* $n=1$,

$$D=D_n,$$

and

$$\gamma=\gamma_n,$$

where

$$\omega_i=c_i/\log b_i, \qquad (i=1, \cdots, n)$$

$$\gamma_i=\sum_{j=1}^{j=i}\omega_j, \qquad (i=1, \cdots, n)$$

$$D_i=-\gamma_i/\prod_{j=1}^{j=i}(-\omega_j)^{\omega_j/\gamma_i}, \qquad (i=1, \cdots, n)$$

$$t_i=D_i\omega_{i+1}/\gamma_i. \qquad (i=1, \cdots, n-1)$$

Proof. The proof is by induction. First define $\beta_k(x)$ to be the corresponding minimal sum for the first k stages; if the investment x is restricted to these stages:

$$\beta_k(x)=\min_{\Sigma_{j=1}^{j=k} x_j=x}\sum_{i=1}^{i=k} b_i^{x_i/c_i}. \qquad (k=1, \cdots, n)$$

Then

$$\beta_1(x)=b_1^{x/c_1}=e^{x(\log b_1)/c_1}=e^{x/\gamma_1}, \qquad (x\geqq 0)$$

which verifies the theorem for $n=1$. Assume now that the theorem holds for $n=1, 2, \cdots, k$. Then (as an example of the dynamic programming principle of optimality),

$$\beta_{k+1}(x)=\min_{0\leqq y\leqq x}[\beta_k(x-y)+e^{y/\omega_{k+1}}],$$

$$=\min_{0\leqq y\leqq x}[D_k e^{(x-y)/\gamma_k}+e^{y/\omega_{k+1}}]. \qquad (x>X)$$

Here y is the cost allocated to the $k+1$th stage.

To calculate $\beta_{k+1}(x)$, denote the expression in the brackets as $f(y)$,

then solve $f'(y)=0$ for y. The solution is

$$y=y_0=[(x/\gamma_k)+\log t_k]/[(1/\gamma_k)+(1/\omega_{k+1})].$$

Since f'' is always positive, and $f'(y_0)=0$, f has a minimum at y_0. That y_0 is actually inside the interval $0\leqq y\leqq x$, follows from the assumption $x>X$. Therefore

$$\beta_{k+1}(x)=f(y_0)=[D_k t_k^{-\omega_{k+1}/\gamma_{k+1}}+t_k^{\gamma_k/\gamma_{k+1}}]$$
$$\exp\{x(1/\gamma_k)(1/\omega_{k+1})/[(1/\gamma_k)+(1/\omega_{k+1})]\}.$$

Remembering that $\gamma_k=\sum\omega_i$, the right-hand bracket becomes simply $1/\gamma_{k+1}$. The left-hand bracket establishes a recursive formula for D_k. This can be converted to the direct expression in the theorem as follows:*

$$D_i\, t_i^{-\omega_{i+1}/\gamma_{i+1}}+t_i^{\gamma_i/\gamma_{i+1}}=t_i^{\gamma_i/\gamma_{i+1}}[1+(D_i/t_i)]=(\gamma_{i+1}/\omega_{i+1})t_i^{\gamma_i/\gamma_{i+1}}.$$

Thus $$t_i=D_i\omega_{i+1}/\gamma_i=(\omega_{i+1}/\omega_i)t_{i-1}^{\gamma_i/\gamma_i}.$$

Now $$t_1=\omega_2/\omega_1,$$

and, by induction,

$$t_i=-\omega_{i+1}/\prod_{j=1}^{j=i}(-\omega_j)^{\omega_j/\gamma_i},$$

so that $$D_i=-\gamma_i/\prod_{j=1}^{j=i}(-\omega_j)^{\omega_j/\gamma_i}.$$

This completes the proof.

Error Estimate

The pertinence of the foregoing result depends on how well system unavailability is approximated by the sum of the stage unavailabilities. This section outlines a proof that if

$$\beta(x)=\min_{\Sigma x_i=x}\sum_{i=1}^{i=n}B_i(x_i),$$
$$\bar{\beta}(x)=\min_{\Sigma x_i=x}\left\{1-\prod_{i=1}^{i=n}[\{1-B_i(x_i)\}]\right\},$$

and $$\beta(x)<\tfrac{1}{4};$$

then $$0\leqq\beta(x)-\bar{\beta}(x)\leqq\beta^2(x).$$

To prove the above error estimate, one may first show that such a sum B of stage unavailabilities is an approximation for system unavailability $\bar{B}$—that in particular, if $\bar{B}<\tfrac{1}{2}$ and $B<1$, then

$$0\leqq B(x_1,\cdots,x_n)-\bar{B}(x_1,\cdots,x_n)\leqq\tfrac{1}{2}B^2(x_1,\cdots,x_n)$$
$$\leqq\tfrac{1}{2}\bar{B}^2(x_1,\cdots,x_n)/[1-2\bar{B}(x_1,\cdots,x_n)], \qquad (1)$$

* The author is grateful to Mrs. Betty J. Flehinger for discovering this direct formula for D_i.

where, for an n-stage system with stage unavailabilities $\bar{B}_i (i=1, \cdots, n)$,

$$B(x_1, \cdots, x_n) = \sum_{i=1}^{i=n} \bar{B}_i(x_i),$$

$$\bar{B}(x_1, \cdots, x_n) = 1 - \prod_{i=1}^{i=n} [1-\bar{B}_i(x_i)].$$

The inequalities (1) can be derived from the equation

$$1-\bar{B}=1-B+\sum_{\substack{i,j=1 \\ i\neq j}}^{n} \bar{B}_i \bar{B}_j - \sum_{\substack{i,j,k=1 \\ i\neq j\neq k\neq i}}^{n} \bar{B}_i \bar{B}_j \bar{B}_k + \cdots + (-1)^n \prod_{i=1}^{n} \bar{B}_i.$$

The terms following B alternate in sign and can be shown to decrease in absolute value. Therefore,

$$0<B-\bar{B}<\sum_{\substack{i,j=1 \\ i\neq j}}^{n} \bar{B}_i \bar{B}_j = \tfrac{1}{2}\left(B^2 - \sum_{i=1}^{n} \bar{B}_i^2\right) < \tfrac{1}{2} B^2.$$

This proves the first two inequalities in (1). It also follows from $B-\bar{B}<\frac{1}{2} B^2$ that

$$\begin{aligned}
B &< 1-(1-2\bar{B})^{1/2}, \\
\tfrac{1}{2} B^2 &< 1-\bar{B}-(1-2\bar{B})^{1/2} \\
&= 1-\bar{B}-[1-\bar{B}-\tfrac{1}{2}\bar{B}^2-\tfrac{1}{2}\bar{B}^3-\textstyle\sum_{k=4}^{k=\infty}\{(2\bar{B})^k/k!2^k\}\prod_{j=1}^{j=k}(2j-3)] \\
&< \tfrac{1}{2}\bar{B}^2+\tfrac{1}{2}\bar{B}^3+\tfrac{1}{8}\textstyle\sum_{k=4}^{k=\infty}(2\bar{B})^k \\
&= \tfrac{1}{2}\bar{B}^2+\tfrac{1}{2}\bar{B}^3+2\bar{B}^4/(1-2\bar{B}) \\
&= \tfrac{1}{2}\bar{B}^2[(1-\bar{B}-\bar{B}^2)/(1-2\bar{B})] < \tfrac{1}{2}\bar{B}^2/(1-2\bar{B}).
\end{aligned}$$

Assume that the particular arrangement $(x_1', \cdots, x_n')$ yields the minimum unavailability $\bar{\beta}$ for a given total cost $x = \sum x_i'$. Since

$$\bar{\beta} \leqq B(x_1', \cdots, x_n')$$

we therefore have

$$0 \leqq \beta - \bar{\beta} \leqq B(x_1', \cdots, x_n') - \bar{B}(x_1', \cdots, x_n') \leqq \tfrac{1}{2}\bar{\beta}^2/(1-2\bar{\beta}).$$

But for $\bar{\beta}$ and β both less than $\frac{1}{2}$, $\bar{\beta} \leqq \beta$ implies

$$\tfrac{1}{2}\bar{\beta}^2/(1-2\bar{\beta}) \leqq \tfrac{1}{2}\bar{\beta}^2/(1-2\beta) \leqq \beta^2,$$

so that we have an estimate for the error in terms of the minimal sum itself.

Note that the error estimate does not depend on the number of stages. This gives an estimate of the error (arising from minimizing the sum of stage unavailabilities instead of minimizing the more accurate expression for the system unavailability) in terms of the minimal sum. Typically the unavailabilities are small enough so that this error is negligible compared with the system unavailability—for example, if β is 0.01, then the error is less than 0.0001.

Computation Rules Implied by Formula

It may be noted that the formula results from unavailability allocations to each stage in the ratio $c_i/\log b_i$—a quantity that has long been recognized as controlling (for example, by Moskowitz and McLean in reference 3). This expression may be used to calculate two points on the maximum availability curve, which may then be plotted as a straight (as indicated by the formula) line on semi-log coordinate paper.

Uses of the Maximum Formula

For the sample system described in the first section, the straight line of the maximum formula is plotted in Fig. 4, together with the step function derived from the algorithm. This figure suggests the following uses of the maximum formula (in addition to its possible direct use where applicable):

1. To check the step functions determined from the dynamic programming algorithm. The step functions are, of course, very irregular, and it is valuable to have a continuous approximation to check them against for reasonableness. Note that in the example the minimum unavailabilities actually attainable with whole numbers of components average about an order of magnitude larger than the 'ideal' attainable with fractional components.

2. To furnish a quick estimate of availability *improvement* that may be expected by additional investment in redundancy. The easily-computed slope of the line graph in Fig. 4 is all that is needed to estimate marginal costs of additional availability. It specifies the (fixed) percentage by which the unavailability may be decreased per additional unit redundancy investment in the system.

3. To speed up the dynamic programming algorithm. It was noted in the first section that the dominating sequence of a subsystem often contains a number of steps representing small improvements for relatively large additional costs. In the next level of calculations, when these subsystems are combined, such steps often are skipped entirely in favor of alternative investment in the other subsystem. The maximum formula provides an immediate criterion with which to judge the efficiency of individual steps. It is conjectured that inefficient steps, for example, those whose marginal decrease in unavailability per unit cost is less than half of the average determined by the maximum formula, could be rejected a fortiori. The solid points in Fig. 4 illustrate how the step function would change. The steps that would be omitted are small circles. Note that the efficient combinations are still preserved; that what is lost are some intermediate investments that force particularly awkward redundancy combinations. Although it appears likely that successive combinations of these simplified step functions would still contain the most efficient combinations (remember, for example, how the inefficient step 2 was skipped in Fig. 2), a systematic investigation of the consequences in later combinations of such a rejection procedure has not been made.

In the same way as **3** above, the formula may be used to check the efficiency of the dominating configurations determined by the method of Black and Proschan.[2]

ACKNOWLEDGMENTS

THIS PAPER is presented with the permission of the Radio Corporation of America; the work was done under RCA Purchase Order No. GX9-947791-0002-96-D82, a subcontract for the United States Air Force. The effective cooperation and direction received from MESSRS. LEON F. FABBIOLI and PETER R. GYLLENHAAL of RCA in the course of this study are gratefully acknowledged. A substantial part of the mathematics supporting the techniques presented in this paper was contributed by DANIEL H. WAGNER.

REFERENCES

1. RICHARD BELLMAN, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.
2. GUY BLACK AND FRANK PROSCHAN, "On Optimal Redundancy," *Opns. Res.* **7**, 581–588 (1959).
3. F. MOSKOWITZ AND J. B. MCLEAN, "Some Reliability Aspects of Systems Design," Rome Air Development Center Report RADC-TN-55-4, May, 1955.