Journal of the
Military Operations Research
Society of Korea, Vol. 9, No. 1
June, 1983

An Algorithm for Optimal Allocation of Spare Parts

Jee, Man Won*

ABSTRACT

The algorithm developed in this paper utilized kettelle's [1] idea of the undominated allocation sequence and his way of tableau computation to solve the more general spares allocation problem in the system availability optimization. The algorithm is to optimally allocate resources to the independent modules which are connected to be series/parallel/mixed system configurations. It has advantages over the standard dynamic programming algorithm by eliminating the need for backtracking and by solving the allocation problem for any budget size. By careful heuristic inspection the algorithm can be made very efficient for manual calculations because large blocks of cells can be eliminated from computation.

A numerical example is provided to illustrate the allocation algorithm.

1. Introduction

Various authors have considered resource allocation of sapres (parallel redundancy) in order to maximize system reliability with a budget constraint. Moskowitz and McLean [11] used a variational method to optimize redundancy. Burtion and Howard [4] and Bellman and Dreyfus [3] solved the problem using dynamic programming. Kettelle [8] developed a heuristic algorithm from dynamic programming and that was extended by Proschan and Bray [13] to allow for multiple constraints. Ghare and Taylor [6] used a branch-and-bound procedure; Mizukami [10] used convex integer programming, and Tillman and Liittschwager [17] maximized reliability subject to several constraints using an integer programming formulation. Everett [5] used the generalized Lagrange multiplier method to solve the same problem discussed by Kettelle. Sharma and Venkateswaran [14] and Nakagawa and Nakashima [12] developed intuitive algorithms that provide approximate solutions. In all of those cases the system consists of components in series with parallel redundancy.

* Korea Institute of Defense Analyses

Aggarwal [1], and Kuo, Hwang, and Tillman [9] develop heuristic methods for optimal system reliability for more complex systems. Other papers [15] and [16] have attempted to determine stock levels for components in order to maximize equipment operational availability subject to a budget constraint. For their purposes operational availability for an equipment is calculated using the definition

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR} + \text{MSRT}}$$

where MSRT is the mean supply response time. Since MTBF and MTTR are unaffected by the number of spares, the models really minimize mean supply response time. The method they use is a Lagrange multiplier approach with an embedded dynamic programming technique.

In the research summarized above there have been many approaches to solving resource allocation problems using various mathematical programming techniques and considering different types of resource constraints. Our spares allocation problem could also be solved several different ways. The system availability objective function does not possess the required characteristics (linear, convex, concave) that allow the use of some of the specialized programming techniques. It is, however, separable and monotone in the decision variables (the numbers of spares) and the constraints are linear. Therefore, solution techniques like dynamic programming or heuristic methods can be applied.

None of the algorithms described above were developed for the availability allocation problem that we address. However, there are similarities that we can exploit. In the next section we present an allocation algorithm for the availability problem that extends the results obtained by Burton and Howard [4] and Kettelle [8].

2. The Allocation Algorithm

Consider a weapon system with k components, and let n_i be the number of spares allocated to component i for $i = 1, 2, ..., k$. Let L_i and U_i be lower and upper bounds on the number of spares for component i and let $n = (n_1, n_2, ..., n_k)$ be the vector of spares allocation for the k components. Finally, let c(n) be the total cost for the allocation n and B the upper bound on the dollars available for allocation. The mathematical problem we address is

$$\begin{aligned} &\max \quad A^{(n)}(t) \\ &\text{subject to} \quad c(n) \leq B \\ &\qquad L_i \leq n_i \leq U_i \\ &\qquad i = 1, 2, ..., k \end{aligned} \tag{2-1}$$

where $A^{(n)}(t)$ is the system point availability with allocation n.

The mathematical programming problem (2-1) has little in the way of special structure in the objective function which allows the use of various efficient linear or nonlinear programming algorithms. However the system availability does possess a separability and monotonic structure that allows the use of dynamic programming methods. The solution technique that we derive is based on Kettelle's reliability redundancy allocation algorithm [8].

Kettelle provides an easily usable algorithm for obtaining an exact solution for maximizing the reliability of a parallel redundancy series system subject to a budget constraint. His algorithm generates undominated redundancy allocations (or dominating sequence of allocations) for successively larger subsystems from undominated allocations for small subsystems.

To understand the concept of "an undominated allocation" the following definition is introduced [2].

Definition : n° is undominated (or dominating) if

$$A^{(n)}(t) > A^{(n^\circ)}(t) \text{ implies } c(n) > c(n^\circ)$$

whereas

$$A^{(n)}(t) = A^{(n^\circ)}(t) \text{ implies either } c(n) > c(n^\circ)$$

$$\text{or } c(n) = c(n^\circ)$$

where

$$n = (n_1, n_2, \dots, n_N).$$

Kettelle confined his algorithm to series-type systems. Burton & Howard [4] showed that the allocation problem can be solved by dynamic programming for any system configuration composed of a mixture of series and parallel connections. They developed a computer algorithm for this problem and demonstrated that the dynamic programming method works very well for complex systems.

The Burton-Howard recursive dynamic programming algorithm can be improved computationally by introducing Kettelle's idea of undominated sequences and his tableau computation methods. The algorithm which we develop for optimizing system availability adapts features of both the Kettelle algorithm and the Burton-Howard algorithm and results in an algorithm which is computationally efficient. The result solves the allocation problem not only for the budget B, but also for all budgets $B' \leq B$. This is important since in real world applications the budget B itself is not always known precisely at the time of solution. We describe the algorithm in the following material. We assume that the computational formulae derived in [7] are used to evaluate component availability and that the system configuration is known. A numerical example is provided in Section 3.

The steps of the availability allocation algorithm are described below. Since the algorithm was generated out of a dynamic programming solution, it is not necessary to prove optimality. (Proofs are available from the Kettelle and Burton-Howard references.) Where we have made modifications of the previous results to reduce the number of decision alternatives at a given point, we prove that the modifications cannot result in inferior solutions.

The Algorithm

Define the N stages ($N \leq k$) to consist of the independent system entities. These entities can be the individual components if they operate independently or modules composed of dependently operating components. Let $a_i^{(n_i)}$ be the availability (for convenience we drop reference to time t even though there is still a specific time t in which we are interested) for the independent entity i, i = 1,2, ..., N.

1) Compute the stage return, $a_i^{(n_i)}$ for stage i for $L_i \leq n_i \leq U_i$. This calculation utilizes computational formulae in [7].

2) Formulate the N-stage return function using the availability calculus for independent series/ parallel configurations. Burton and Howard show that the problem can always be formulated so that separability and monotonicity are satisfied for any series/parallel mixed system configuration provided the entities are independent.

3) Set up a tableau such as that shown in Tables 2.7 for the two-stage problem. The entries in the row headings correspond to the triple, $(n_1, n_1c_1$, and $a_1^{(n_1)})$; the entries in the column heading correspond to $(n_2, n_2c_2, a_2^{(n_2)})$. The entries in the body of the table give the allocation (n_1, n_2), the cost $c(n_1, n_2)$ and the two-stage return (availability).

4) Start with (1,1) as the first element in the sequence of undominated allocations for the subsystem consisting of stages one and two.

5) Select as the next member of the sequence of dominating allocations the cheapest cost entry with availability higher than the previous element of the sequence. (We discuss later some methods of eliminating entire blocks of possibilities from consideration.)

6) After proceeding through a given tableau in the above manner, increase the problem to a 3-stage return using the undominated allocations from the previous 2-stage problem as the row entries and $(n_3, c_3(n_3), a_3^{(n_3)})$ as the column entries. Obtain the dominating allocations for this subsystem consisting of stages 1, 2, and 3 as before.

7) In general, for the N-stage problem, use as the row elements $((n_1, n_2, ..., n_{N-1}), c(n_1, n_2, ..., n_{N-1}), A^{(n_1, n_2, ..., n_{N-1})})$ and as the column elements $(n_N, c(n_N), a^{n_N})$ and step through selecting a sequence of dominating allocations by taking the cheapest cost entry with availability higher than the previous value.

Note that if c_{ij} and a_{ij} represent the cost and availability of cell (i,j), and if cell (i,j) is a member of the optimal sequence, then the next member of the optimal sequence must be located in the region represented as quadrant I or III in Figure 2.1 below.

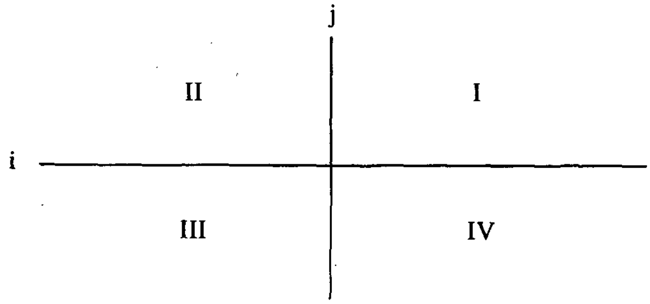


Fig. 2.1 Definition of quadrant I, II, III, IV

The tableau method used in this algorithm has the following advantages:

1) It does not need backtracking procedures which are used in the standard dynamic programming algorithm.

2) It does not need extra calculations for the changes of budget level. Since we can read off Figure 2.3, the allocation and availability for each level of the budget ranges from 0 to 50.

3) It is efficient especially for manual calculations. As we could see in the tables (Tables 2.7 ~ 2.11) we do not need to fill out every cell of the tables. Heuristic inspection of some of the cells allows the user to eliminate large blocks of cells. As an example, suppose, in Table 2.9, we have just computed elements of the cell (9,7) of the table 2.9 which shows allocation (3,4,0,0,8), cost 17.4 and availability 0.6336, and we found out that this allocation should currently be included in the undominated sequence according to the 5th criterion of this algorithm, then we know that we have already considered all the elements of the cells above the solid line in Table 2.9, all of which have costs less than 17.4. So in the next calculations, we may try the entries in several cells in quadrant I or III for the possible elimination of entire blocks. Suppose for example we compute the availability in the cell (19, 5) of Table 2.9 which we found to be 0.5966 (lower than 0.6336) with cost 24.6 (higher than 17.4), then we do not need to consider cells which are contained in the shaded area ("North-West Corner" elimination rule). This is because cost and availability of the cell (19,5) of Table 2.9 are maximal among those of the cells contained in the shaded area (cost and availability in the cell (i,j) of the table are increasing in i and/or j), and thus every cell in the shaded area has lower availability with higher cost than those of cell (9,7) which is the current element in the dominating sequence.

In those stages where cumulative stage returns are computed from series connections, the remainder of an entire row or column can sometimes be rejected as dominated. Let c_{io} and a_{io} represent the cost and availability of the entry heading row i. If $a_{ij} > a_{i'o}$ where $i' < i$, all entires in row i' which cost more than c_{ij} are dominated because $a_{ij} > a_{i'o} > a_{i'k}$ $k \geq 1$. The second inequality follows because $a_{i'k} = a_{i'o} \cdot a_{ok}$ where $a_{ok} \leq 1$, $k = 1,2, \ldots$. The same is true for columns. As an example, note that in Table 2.7 cell (2,2) (cost 5.4, availability 0.330135) dominates every cell in the first row starting with (1,4) since $a_{22} = 0.330135 > a_{1o} = 0.2865 > a_{1k}$ $(k \geq 1)$, and $c_{22} = 5.4 < c_{1\ell}$, $\ell \geq 3$.

In those stages where cumulative stage returns are computed from parallel connections there seems to be noclear cut rule of elimination other than the "north-west corner elimination rule", i.e., try several cells heuristically in quadrant I and III, if $a_{i'j'} < a_{ij}$ then we eliminate from further consideration cell (ℓ, m) such that $\ell \leq i'$ and $m \leq j'$.

3. An Example

As an illustration consider the system configuration shown in Figure 2.2 and the data in Table 2.1. Solve the following problem.

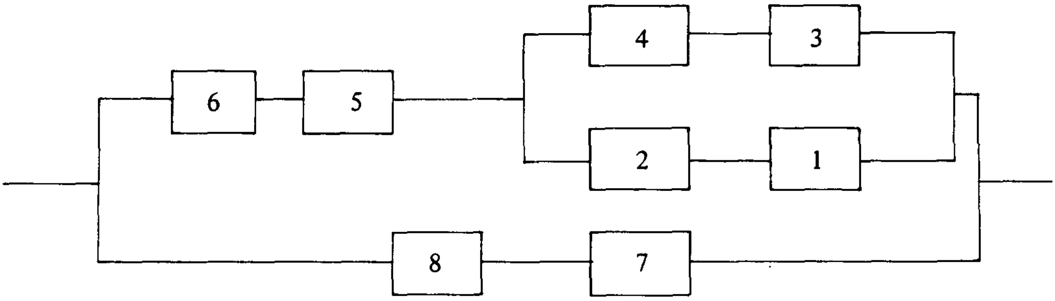


Fig. 2.2 System configuration for example problem

Table 2.1 System Data

Component		Failure Rate (λ)	Recovery Rate (η)	Cost (c)	Upper Bound (u)	Number of Spares (n)	Availability (a)	Operating Rule
① ②	①	$\frac{1}{80}$	$\frac{1}{4}$	1.4	7	n_1	a_1	scenario 1 spares not shared
	②	$\frac{1}{60}$	$\frac{1}{3}$	1.3	7	n_2	a_2	
③ ④	③	$\frac{1}{50}$	$\frac{1}{2}$	1.5	4	n_3	a_{34}	scenario 2 spares not shared
	④	$\frac{1}{40}$	$\frac{1}{2}$	1.2	6	n_4		
⑤ ⑥		$\frac{1}{50}$	$\frac{1}{4}$	1.0	8	n_{56}	a_{56}	scenario 2 spares shared
⑦ ⑧		$\frac{1}{40}$	$\frac{1}{2}$	1.4	9	n_{78}	a_{78}	scenario 1 spares shared

Scenario 1 : The surviving component continues in service, and if it fails, its replacement begins immediately and proceeds independently and concurrently with the replacement of the other failed component. All failed components resume operation as soon as they are replaced.

Scenario 2 : The surviving component is shut down until replacement of the failed component is accomplished.

The problem is:

$$\begin{aligned} &\max \ A^{(n)}(t) \\ &\text{s.t. } n \cdot c \leq B \\ &\quad 0 \leq n_i \leq U_i \end{aligned}$$

for t = 90, B is flexible between 40 and 50.

Solution: For this problem the availability is given by

$$\begin{aligned} A^{(n_1,n_2,n_3,n_4,n_{56},n_{78})} = 1 - [1-a_{78}{}^{(n_{78})}]\,\{1-a_{56}{}^{(n_{56})} \\ \cdot [1-(1-a_{34}{}^{(n_3,n_4)})(1-a^{(n_1)}a_2{}^{(n_2)})]\} \end{aligned} \tag{2-2}$$

which is separable since it can be written as

$$A^{(n_1,n_2,n_3,n_4,n_{56},n_{78})} = a_{78} \circ a_{56} \circ a_{34} \circ a_2 \circ a_1$$

where "o" represents the composition operator.

We seek

$$\begin{aligned} A(B) = &\max_{n} \ A^{(n_1,n_2,n_3,n_4,n_{56},n_{78})} \\ &\quad \text{s.t. } n \cdot c \leq B \\ &\quad\quad 0 \leq n_i \leq U_i \end{aligned}$$

Let $a_i{}^{(n_i)}$ be the stage return and $A_i(X_i)$ be the maximum i-stage return.

For stage 1,

$$\begin{aligned} A_1(X_1) = &\max_{n_1} a_1{}^{(n_1)} \\ &\quad \text{s.t. } 0 \leq n_1 \leq 6 \\ &\quad\quad 0 \leq c_1 n_1 \leq X_1 \end{aligned} \tag{2-3}$$

For stages 2, 3, 4 and 5

$$\begin{aligned} A_2(X_2) = &\max_{n_2} a_2{}^{(n_2)} \cdot A_1(X_2 - n_2 c_2) \\ &\quad \text{s.t. } 0 \leq n_2 \leq 6 \\ &\quad\quad 0 \leq n_2 c_2 \leq X_2 \end{aligned} \tag{2-4}$$

$$A_{34}(X_{34}) = \max_{n_3,n_4} \quad 1-[1-a_{34}{}^{(n_3,n_4)}]\ [1-A_2(X_{34}-n_3c_3-n_4c_4)]$$

$$\text{s.t.}\quad 0 \leq n_3 \leq 3$$

$$0 \leq n_4 \leq n_5 \qquad (2\text{-}5)$$

$$n_3 c_3 + n_4 c_4 \leq X_{34}$$

$$A_{56}(X_{56}) = \max_{n_{56}} a_{56}^{(n_{56})} \cdot A_{34}(X_{56} - c_{56} n_{56})$$

$$\text{s.t.}\quad 0 \leq n_{56} \leq 6 \qquad (2\text{-}6)$$

$$0 \leq c_{56} n_{56} \leq X_{56}$$

$$A_{78}(X_{78}) = 1 - \min_{n_{78}} [1 - a_{78}^{(n_{78})}]\ [1 - A_{56}(X_{78} - c_{78} n_{78})]$$

$$\text{s.t.}\quad 0 \leq n_{78} \leq 7 \qquad (2\text{-}7)$$

$$0 \leq c_{78} n_{78} \leq X_{78} = B$$

The stage returns ($a_i^{(n_i)}$) are computed from the formulas in (7) and the computation results are listed in Tables (2.2 - 2.6). Here n_i is the total number of parts (original plus spares for component i).

At stage 1 and 2, from Eq. (2-3) and Table 2.2 the maximum return from stage 1 is

$$A_1(n_1 c_1) = a_1^{(n_1)}$$

To obtain a complete sequence of undominated allocations for the subsystem consisting of stages 1 and 2 according to recursive Equation (2-4), we set up Table 2.7. The entires in the body of the Table (2.5) give the spares, (vector of two elements), cost, and availability for the subsystem consisting of stages 1 and 2. Thus, the entry (2,3) corresponds to $n_1 = 2$ and $n_2 = 3$ with cost 6.7 achieving subsystem availability 0.495725. The chosen elements connected by arrows form an undominated

Table 2.2 Stage return from stage 1

n_1	$n_1 c_1$	$a_1^{(n_1)}$
1	1.4	0.2865
2	2.8	0.6476
3	4.2	0.8565
4	5.6	0.9305
5	7.0	0.9379
6	8.4	0.9412
7	9.8	0.9417
∞	∞	0.9524

Table 2.3 Stage return from stage 2

n_2	$n_2 c_2$	$a_2^{(n_2)}$
1	1.3	0.1889
2	2.6	0.5098
3	3.9	0.7655
4	5.2	0.8929
5	6.5	0.9377
6	7.8	0.9495
7	9.1	0.9519
∞	∞	0.9524

sequence of allocations.

The elements of the sequence are chosen in the following way.

1) Start with (1.1), the first undominated allocation.

2) The next undominated allocation is the cheapest cost entry with availability higher than that of previous allocations.

Table 2.4 Stage return from stage 3

$\begin{bmatrix} n_4 \\ n_4c_4 \end{bmatrix}$ $\begin{bmatrix} n_3 \\ n_3c_3 \end{bmatrix}$	1 1.2	2 2.4	3 3.6	4 4.8	5 6.0	6 7.2
1 1.5	(1,1) 2.7 0.0111	(1,2) 3.9 0.0418	(1,3) 5.1 0.0802	(1,4) 6.3 0.1116		
2 3.0	(2,1) 4.2 0.0350	(2,2) 5.4 0.1271	(2,3) 6.6 0.2409	(2,4) 7.8 0.3315		
3 4.5	(3,1) 5.7 0.0595	(3,2) 6.9 0.2120	(3,3) 8.1 0.3968	(3,4) 9.3 0.5407	(3,5) 10.5 0.6213	
4 6.0				(4,4) 10.8 0.7160	(4,5) 12.0 0.8124	(4,6) 13.2 0.8533

Table 2.5 Stage return from stage 4

n_{56}	$n_{56}c_{56}$	$a_{56}(n_{56})$
2	2.0	0.01832
3	3.0	0.1034
4	4.0	0.2723
5	5.0	0.4854
6	6.0	0.6617
7	7.0	0.7747
8	8.0	0.8285

Table 2.6 Stage return from stage 5

n_{78}	$n_{78}c_{78}$	$a_{78}(n_{78})$
2	2.8	0.0079
3	4.2	0.0426
4	5.6	0.1300
5	7.0	0.2741
6	8.4	0.4483
7	9.8	0.6133
8	11.2	0.7410
9	12.6	0.8238

Table 2.7 Sequence of max return from stage 1,2

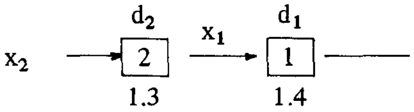
a_2 / a_1	1 1.3 0.1889	2 2.6 0.5098	3 3.9 0.7655	4 5.2 0.8929	5 6.5 0.9377	6 7.8 0.9495	7 9.1 0.9519
1 1.4 0.2865	(1,1) 2.7 0.0541	(1,2) 4.0 0.1461	(1,3) 5.3 0.2193	6.6 0.2558			
2 2.8 0.6476	4.1 0.1223	(2,2) 5.4 0.3301	(2,3) 6.7 0.4957	(2,4) 8.0 0.5783	(2,5) 9.3 0.6072		
3 4.2 0.8565			(3,3) 8.1 0.6556	(3,4) 9.4 0.7648	(3,5) 10.7 0.8031		
4 5.6 0.9305				(4,4) 10.8 0.8309	(4,5) 12.1 0.8725	(4,6) 13.4 0.8835	(4,7) 14.7 0.8858
5 7.0 0.9379						(5,6) 14.8 0.8906	(5,7) 16.1 0.8928
6 8.4 0.9412						(6,6) 16.2 0.8937	(6,7) 17.5 0.8960
7 9.8 0.9417	11.1 0.1779	12.4 0.4800	13.7 0.7208	15.7 0.8408	16.3 0.8830	17.6 0.8941	(7,7) 19.9 0.8964

These procedures are consistent with the Burton-Howard algorithm (Table 2.8). The sequence of undominated allocation in Table 2.7 shows that if we have 15 as our budget for this two-component subsystem then we allocate $n_1 = 5$ and $n_2 = 6$ with cost 14.8 achieving availability 0.89055, if B = 20 then n = (7,7) with cost 19.9 achieving 0.8964 and so forth.

At stage 3, component #3 and #4 form a two-component series subsystem which operates under scenario 2 with spares not shared. The stage returns are computed from the formula (4-22) and (4-23) in [7] and listed in Table 2.4. The sequence of undominated allocations is marked by arrows also in Table 2.4. From recursive equation (2-5) we obtain a sequence of undominated allocations for the subsystem consisting of component #1, #2, #3 and #4 by using the undominated allocations of stage 1,2 and the undominated allocations of stage 3 which is shown in Table 2.9.

Table 2.8 Sequence of max return from stage 1,2 using standard dynamic programming method

$c_1 = 1.4$
$c_2 = 1.3$



x_2	d_2	$c(d_2)$	a_2	$x_2 - c(d_2)$ x_1	d_1	A_1	a_2A_1	A_2
1	0	0	0	1	0	0	0	0
2	0	0	0	2	1	0	0	0
	1	1.3	.1889	0.7	0	0	0	
3	0	0	0	3	2	.6476	0	
	①	1.3	.1889	1.7	①	.2865	0.0541	0.0541
	2	2.6	.5098	0.4	0	0	0	with cost 2.7
4	0	0	0	4	2	.6476	0	
	1	1.3	.1889	2.7	1	.2865	0.0541	
	②	2.6	.5098	1.4	①	.2865	.1461	0.1461
	3	3.9	.7655	0.1	0	0	0	with cost 4.0
5	0	0	0	5	3	.8565	0	
	1	1.3	.1889	3.7	2	.6476	.1223	
	②	2.6	.5098	2.4	①	.2865	.1461	0.1461
	3	3.9	.7655	1.1	0	0	0	with cost 4.0
6	0	0	0	6	4	.9305	0	
	1	1.3	.1889	4.7	3	.8565	1618	
	②	2.6	.5098	3.4	②	.6476	.3301	0.3301 with cost 5.4
	③	3.9	.7655	2.1	①	.2865	.2193	0.2193 with cost 5.3
	4	5.2	.8929	0.8	0	0	0	

At stage 4, since component #5 and #6 operate under scenario 2 with spares shared by both components, the payoff function $a_{56}^{(n_{56})}$ which is computed from formula (4-27) and (4-28) in [7] and listed in Table 2.5 is the availability of the module which is an independent entity.

According to the recursive equations (2-6) we obtain a sequence of undominated allocations for the subsystem consisting of components #1, #2, #3, #4, #5, and #6 by using the undominated allocations of stages 1, 2, 3 and the sequence of payoffs from stage 4 (component #5, #6). Table 2.10 shows this sequence, and Table 2.11 shows final allocations according to the recursive equations (2-7) in a similar fashion.

From the final table (2.11) we can construct the following graph which shows allocation and availability for each level of budget which is flexible ranging from 40 to 50.

If we would like to have a system availability of 0.900 then we would need only a budget of 26.7 consumed by allocating $(n_1, n_2, n_3, n_4, n_{56}, n_{78})$ = (3,3,0,0,6,9). Figure 2.4 is the graphical representation of Table 2.11.

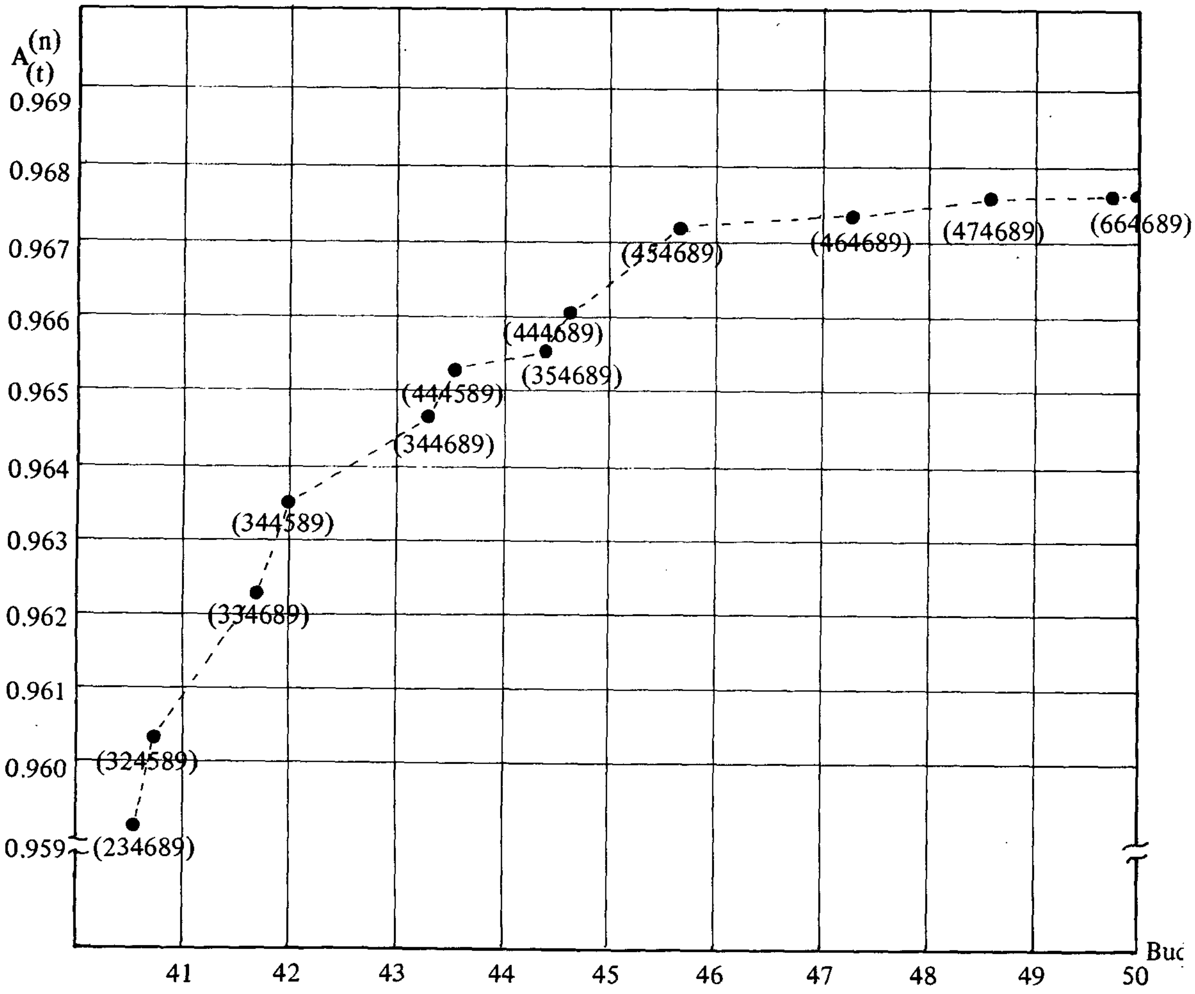


Fig. 2.3 Availability-cost curve for undominated allocations (budget level 40 ~ 50).

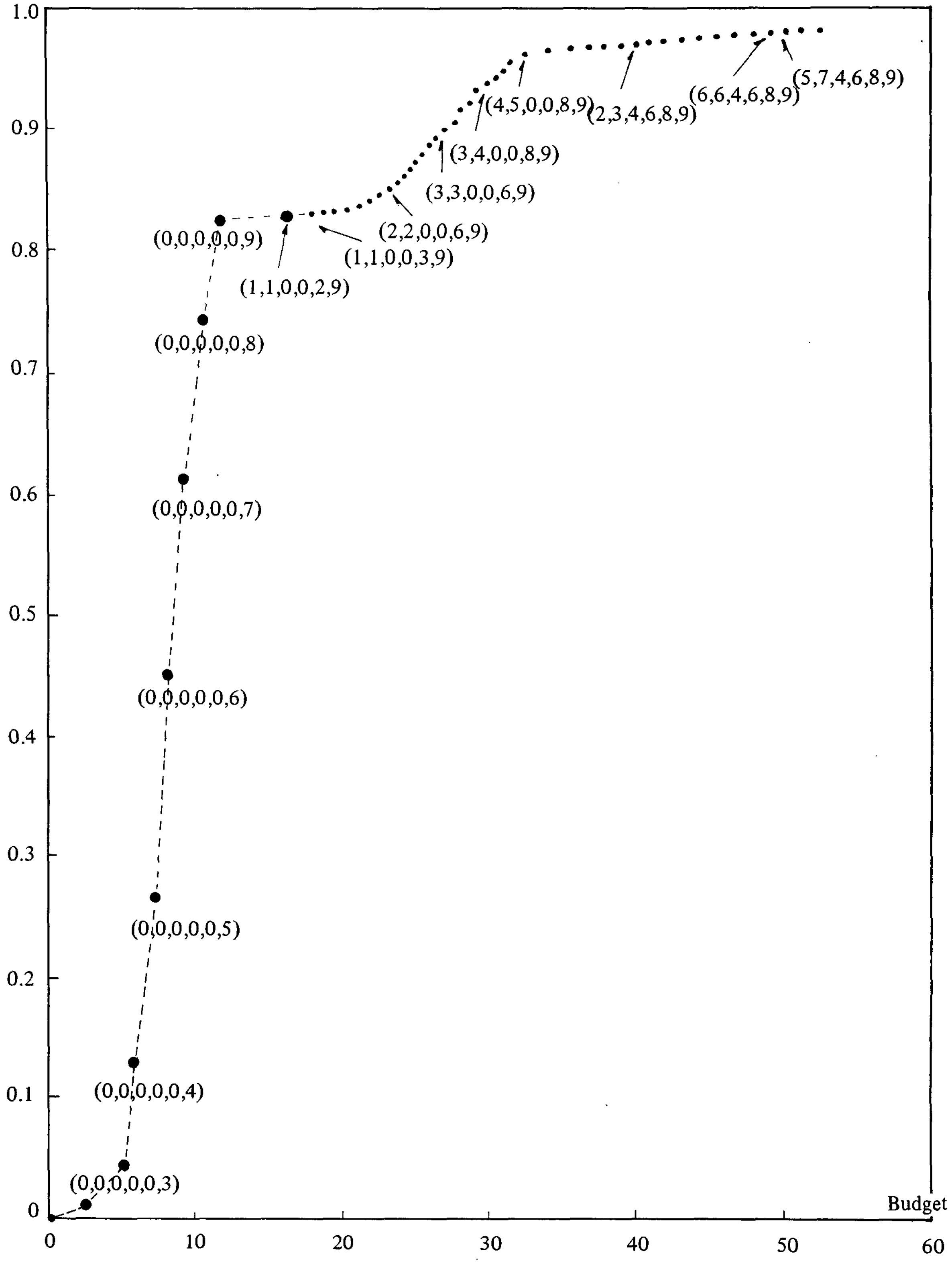


Fig. 2.4 Availability-cost curve for undominated allocations

Table 2.9 Sequence of Max Return from Stage 1, 2, 3

a_{34} \ A_{12}	(0.0) 0 0.0	(1.1) 2.7 0.0111	(1.2) 3.9 0.0418	(1,3) 5.1 0.08016	(2.2) 5.4 0.1271	(2.3) 6.6 0.2409	(2.4) 7.8 0.3315	(3,3) 8.1 0.3968	(3.4) 9.3 0.5407	(3.5) 10.5 0.6213	(4.4) 10.8 0.7160	(4.5) 12.0 0.8124	(4.6) 13.2 0.8533
(0.0) 0 0.0	(0,0,0,0) 0 0.0	(0,0,1,1) 2.7 0.0111	(0,0,1,2) 3.9 0.0418	(0,0,1,3) 5.1 0.0802	(0,0,2,2) 5.4 0.1271	(0,0,2,3) 6.6 0.2409	(0,0,2,4) 7.8 0.3315	(0,0,3,3) 8.1 0.3968	(0,0,3,4) 9.3 0.5407	(0,0,3,5) 10.5 0.6213	(0,0,4,4) 10.8 0.7160	(0,0,4,5) 12.0 0.8124	(0,0,4,6) 13.2 0.8533
(1.1) 2.7 0.0541	(1,1,0,0) 2.7 0.0541	(1,1,1,1) 5.4 0.0646	(1,1,1,2) 6.6 0.0937	(1,1,1,3) 7.8 0.1299	(1,1,2,2) 8.1 0.1743	9.3 0.2824	10.5	10.8 0.429	12.0 0.5456	13.2	13.5 0.7313	14.7	15.9
(1.2) 4.0 0.1461	(1,2,0,0) 4.0 0.1461	(1,2,1,1) 6.7 0.1555	(1,2,1,2) 7.9 0.1817	9.1	9.4					14.5 0.6766	14.8		17.2 0.8747
(1.3) 5.3 0.2193	(1,3,0,0) 5.3 0.2193	8.0 0.228	9.2 0.2530						14.6				18.5 0.8854
(2.2) 5.4 0.3301	(2,2,0,0) 5.4 0.3301	8.1 0.3376	9.3 0.358						14.7			17.4 0.8743	(2,2,4,6) 18.6 0.9017
(2,3) 6.7 0.4957	(2,3,0,0) 6.7 0.4957	9.4 0.5013	10.6 0.5168				14.5	14.8				(2,3,4,5) 18.7 0.9054	(2,3,4,6) 19.9 0.9260
(2,4) 8.0 0.5783	(2,4,0,0) 8.0 0.5783	10.7 0.5829	11.9 0.5959			14.6						(2,4,4,5) 20.0 0.9210	(2,4,4,6) 21.2 0.9381
(3.3) 8.1 0.6556	(3,3,0,0) 8.1 0.6556	10.8 0.6574			13.5	14.7						(3,3,4,5) 20.1 0.9354	(3,3,4,6) 21.3 0.9495
(3,4) 9.4 0.7648	(3,4,0,0) 9.4 0.7648	12.1 0.7674	13.3	14.5								(3,4,4,5) 21.4 0.9559	(3,4,4,6) 22.6 0.9655
(3,5) 10.7 0.8031	(3,5,0,0) 10.7 0.8031	13.4 0.8053	14.6									22.7 0.9630	(3,5,4,6) 23.9 0.9710
(4,4) 10.8 0.8309	(4,4,0,0) 10.8 0.8309	13.5 0.8327	14.7									(4,4,4,5) 23.8 0.9683	(4,4,4,6) 24.0 0.9732
(4,5) 12.1 0.8725	(4,5,0,0) 12.1 0.8725	14.8 0.8739										(4,5,4,5) 24.1 0.9761	(4,5,4,6) 25.3 0.9813
(4,6) 13.4 0.8835	(4,6,0,0) 13.4 0.8835	16.1 0.8843	17.3 0.8884	18.5 0.8928								25.4 0.9781	(4,6,4,6) 26.6 0.9829
(4,7) 14.7 0.8858	(4,7,0,0) 14.7 0.8858	17.4 0.8870										26.7 0.9786	(4,7,4,6) 27.9 0.9832
(5.6) 14.8 0.8906	(5,6,0,0) 14.8 0.8906	17.5 0.8918											(5,6,4,6) 28.0 0.9839
(5,7) 15.1 0.8928	(5,7,0,0) 15.1 0.8928												(5,7,4,6) 29.3 0.9843
(6,6) 15.2 0.8937	(6,6,0,0) 15.2 0.8937												(6,6,4,6) 29.4 0.9844
(6.7) 17.5 0.8960	(6,7,0,0) 17.5 0.8960												(6,7,4,6) 30.7 0.9847
(7.7) 19.9 0.8964	(7,7,0,0) 19.9 0.8964												(7,7,4,6) 32.1 0.9848

Table 2.10 Sequence of Max Return from Stage 1, 2, 3, 4

a_{56} / A_{1234}	2 2.0 0.01832	3 3.0 0.1014	4 4.0 0.2723	5 5.0 0.4854	6 6.0 0.6617	7 7.0 0.7747	8 8.0 0.8285
(0,0,0,0) 0 0.0							
(1,1,0,0) 2.7 0.0541	(1,1,0,0,2) 4.7 0.00099	(1,1,0,0,3) 5.7 0.0055	(1,1,0,0,4) 6.7 0.0147	(11,0,0,5) 7.7 0.0263	8.7 0.0358		
(1,2,0,0) 4.0 0.1460	(1,2,0,0,2) 6.0 0.00267	(1,2,0,0,3) 7.0 0.0148	(1,2,0,0,4) 8.0 0.0398	(1,2,0,0,5) 9.0 0.0709	(1,2,0,0,6) 10.0 0.0966	11.0 0.113	
(1,3,0,0) 5.3 0.2193	7.3 0.00402	8.3 0.0222	9.3 0.0597	(1,3,0,0,5) 10.3 0.1064	11.3 0.1451		
(2,2,0,0) 5.4 0.3301			(2,2,0,0,4) 9.4 0.0899	(2,2,0,0,5) 10.4 0.1603	(2,2,0,0,6) 11.4 0.2184	(2,2,0,0,7) 12.4 0.2558	13.4 0.2735
(2,3,0,0) 6.7 0.4957				(2,3,0,0,5) 11.7 0.2410	(2,3,0,0,6) 12.7 0.328	(2,3,0,0,7) 13.7 0.3840	14.7 0.4107
(2,4,0,0) 8.0 0.5783				13.0 0.2810	14.0 0.3826	(2,4,0,0,7) 15.0 0.4480	
(3,3,0,0) 8.1 0.6556				13.1 0.3182	(3,3,0,0,6) 14.1 0.4338	(3,3,0,0,7) 15.1 0.5079	(3,3,0,0,8) 16.1 0.5432
(3 4 0 0) 9.4 0.7648					15.4 0.5060	(3,4,0,0,7) 16.4 0.5725	(3,4,0,0,8) 17.4 0.6233
(3,5,0,0) 10.7 0.8031							(3,5,0,0,8) 18.7 0.6654
(4,4,0,0) 10.8 0.8309					16.8 0.5497	(4,4,0,0,7) 17.8 0.6437	(4,4,0,0,8) 18.8 0.6884
(4,5,0,0) 12.1 0.8725				17.1 0.4235	18.1	19.1 0.6759	(4,5,0,0,8) 20.1 0.7229
(4,6,0,0) 13.4 0.8835		16.4	17.4	18.4			(4,6,0,0,8) 21.4 0.7320
(4,7,0,0) 14.7 0.8358	16.7	17.7	18.7				(4,7,0,0,8) 22.7 0.7338
(5,6,0,0) 14.8 0.8906	16.8						(5,6,0,0,8) 23.8 0.7378
(5,7,0,0) 16.1 0.8928	18.1						(5,7,0,0,8) 24.1 0.7397
(6,6,0,0) 16.2 0.8937							(6,6,0,0,8) 24.2 0.7404
(6,7,0,0) 17.5 0.8960							(6,7,0,0,8) 25.5 0.7423
(2,2,4,6) 18.6 0.9017	20.6 0.0165	21.6 0.0914	22.6 0.2455	23.6 0.4377	(2,2,4,6,6) 24.6 0.5966	25.6 0.6935	(2,2,4,6,8) 26.6 0.747

a_{56} A_{1234}	2 2.0 0.0183	3 3.0 0.1014	4 4.0 0.2723	5 5.0 0.4854	6 6.0 0.6617	7 7.0 0.7747	8 8.0 0.8285
(2,3,4.5) 18.7 0.9054							(2,3,4,5,8) 26.7 0.750
(2,3.4.6) 19.9 0.9260							(2,3,4,6,8) 27.9 0.7672
(3,3.4,5) 20.1 0.9354							(3,3,4.5,8) 28.1 0.7750
(3,3,4,6) 21.3 0.9495							(3,3,4,6,8) 29.3 0.7866
(3,4,4,5) 21.4 0.9559							(3,4,4,5,8) 29.4 0.7920
(3,4.4,6) 22.6 0.9655							(3,4.4,6,8) 30.6 0.7999
(4,4,4,5) 22.8 0.9683							(4,4,4,5,8) 30.8 0.8032
(3,5,4,6) 23.9 0.9710							(3,5,4,6,8) 31.9 0.8045
(4,4,4,6) 24.0. 0.9752							(4,4,4,6,8) 32.0 0.8079
(4,5.4,6) 25.3 0.9813							(4,5,4,6,8) 33.3 0.813
(4,6.4,6) 26.6 0.9829							(4,6,4,6,8) 34.6 0.8143
(4,7.4,6) 27.9 0.9832							(4.7.4.6,8) 35.9 0.8146
(5,6,4,6) 28.0 0.9839							(5,6,4,6.8) 36.0 0.8152
(5.7.4,6) 29.3 0.9843							(5,7,4,6,8) 37.3 0.8155
(6,6.4,6) 29.4 0.9844							(6,6,4.6,8) 37.4 0.8156
(6,7.4,6) 30.7 0.9847							(6,7,4,6,8) 38.7 0.8158
(7,7,4,6) 32.1 0.9848	34.1 0.218	35.1 0.0993	36.1 0.268	37.1 0.478	38.1 0.5516	39.1 0.7629	(7,7,4,6,8) 40.1 0.8159

Table 2.11 Sequence of Max Return from Stage 1, 2, 3, 4, 5

a_{78} A_{123456}	0 0.0 0.0	2 2.8 0.0079	3 4.2 0.0426	4 5.6 0.130	5 7.0 0.2741	6 84 0.4483	7 9.8 0.6133	8 11.2 0.7410	9 12.6 0.8238
(0,0,0,0 p) 0 0.0	(0,0,0,0,0,0) 0.0 0.000	(0,0,0,0,0,2) → 2.8 0.0079	(000003) → 4.2 0.0426	(000004) → 5.6 0.1300	(0,0,0,0,0,5) → 7.0 0.2741	(0,0,0,0,0,6) → 8.4 0.4483	(0,0,0,0,0,7) → 9.8 0.6133	(0,0,0,0,0,8) → 11.2 0.7410	(0,0,0,0,0,9) → 12.6 0.8238
(1,1,0,0,2) 4.7 0.0010	4.7 0.0010							15.9 0.7412	(1,1,0,0,2,9) 17.3 0.8240
(1,1,0,0,3) 5.7 0.0055									(1,1,0,0,3,9) 18.3 0.8248
(1,1,0,0,4) 6.7 0.0147									(1,1,0,0,4,9) 19.3 0.82639
(1,2,0,0,3) 7.0 0.0148									(1,2,0,0,3,9) 19.6 0.82641
(1,1,0,0,5) 7.7 0.0263									(1,1,0,0,5,9) 20.3 0.8284
(1,2,0,0,4) 8.0 0.0398									(1,2,0,0,4,9) 20.6 0.8308
(1,2,0,0,5) 9.0 0.0709									(1,2,0,0,5,9) 21.6 0.8363
(2,2,0,0,4) 9.4 0.0899									(2,2,0,0,4,9) 22.0 0.8396
(1,2,0,0,6) 10.0 0.0966									(1,2,0,0,6,9) 22.6 0.8408
(1,3,0,0,5) 10.3 0.1064									(1,3,0,0,5,9) 22.9 0.8426
(2,2,0,0,5) 10.4 0.1603								21.6 0.7825	(2,2,0,0,5,9) 23.0 0.8520
(2,2,0,0,6) 11.4 0.2184									(2,2,0,0,6,9) 24.0 0.8623
(2,3,0,0,5) 11.7 0.2410									(2,3,0,0,5,9) 24.3 0.8663
(2,2,0,0,7) 12.4 0.2558									(2,2,0,0,7,9) 25.0 0.8689
(2,3,0,0,6) 12.7 0.3280									(2,3,0,0,6,9) 25.3 0.8816
(2,3,0,0,7) 13.7 0.3840									(2,3,0,0,7,9) 26.3 0.8915

Q_{78} A_{123456}	0 0.0 0.000	2 2.8 0.0079	3 4.2 0.0426	4 5.6 0.1500	5 7.0 0.2741	6 8.4 0.4483	7 9.8 0.6133	8 11.2 0.7410	9 12.6 0.8238
(3,3,0,0,6) 14.1 0.4338									(3,3,0,0,6,9) 26.7 0.900
(2,4,0,0,7) 15.0 0.4480									(2,4,0,0,7,9) 27.6 0.9027
(3,3,0,0,7) 15.1 0.5379									(3,3,0,0,7,9) 27.7 0.9133
(3,3,0,0,8) 16.1 0.5432									(3,3,0,0,8,9) 28.7 0.9195
(3,4,0,0,7) 16.4 0.5925									(3,4,0,0,7,9) 29.0 0.9282
(3,4,0,0,8) 17.4 0.6336									(3,4,0,0,8,9) 30.0 0.9354
(4,4,0,0,7) 17.8 0.6437									(4,4,0,0,7,9) 30.4 0.9372
(3,5,0,0,8) 18.7 0.6654									(3,5,0,0,8,9) 31.3 0.9410
(4,4,0,0,8) 18.8 0.6884									(4,4,0,0,8,9) 31.4 0.9451
(4,5,0,0,8) 20.1 0.7229									(4,5,0,0,8,9) 32.7 0.9512
(4,6,0,0,8) 21.4 0.7320									(4,6,0,0,8,9) 34.0 0.9528
(4,7,0,0,8) 22.7 0.7338									(4,7,0,0,8,9) 35.3 0.9531
(5,6,0,0,8) 22.8 0.7378									(5,6,0,0,8,9) 35.4 0.9538
(5,7,0,0,8) 24.1 0.7397									(5,7,0,0,8,9) 36.7 0.9541
(6,6,0,0,8) 24.2 0.7404									(6,6,0,0,8,9) 36.8 0.9543
(6,7,0,0,8) 25.5 0.7423									(6,7,0,0,8,9) 38.1 0.9546
(2,2,4,6,8) 26.6 0.7470								37.8 0.9345	(2,2,4,6,8,9) 39.2 0.9554

a_{78} A_{123456}	0 0.0 0.000	2 2.8 0.0079	3 4.2 0.0426	4 5.6 0.1500	5 7.0 0.2741	6 8.4 0.4483	7 9.8 0.6133	8 11.2 0.7410	9 12.6 0.8238
(2,3,4,5,8) 26.7 0.7500									(2,3,4,5,8,9) 39.3 0.9560
(2,3,4,6,8) 27.9 0.7672									(2,3,4,6,8,9) 40.5 0.9590
(3,3,4,5,8) 28.1 0.7750									(3,3,4,5,8,9) 40.7 0.9604
(3,3,4,6,8) 29.3 0.7866									(3,3,4,6,8,9) 41.9 0.9624
(3,4,4,5,8) 29.4 0.7920									(3,4,4,5,8,9) 42.0 0.9633
(3,4,4,6,8) 30.6 0.7999									(3,4,4,6,8,9) 43.2 0.9647
(4,4,4,5,8) 30.8 0.8022									(4,4,4,5,8,9) 43.4 0.9651
(3,5,4,6,8) 31.9 0.8045									(3,5,4,6,8,9) 44.5 0.9655
(4,4,4,6,8) 32.0 0.8079									(4,4,4,6,8,9) 44.6 0.9660
(4,5,4,6,8) 33.3 0.8130									(4,5,4,6,8,9) 45.9 0.9671
(4,6,4,6,8) 34.6 0.8143									(4,6,4,6,8,9) 47.2 0.9673
(4,7,4,6,8) 35.9 0.8146									(4,7,4,6,8,9) 48.5 0.9673
(5,6,4,6,8) 36.0 0.8152									(5,6,4,6,8,9) 48.6 0.9674
(5,7,4,6,8) 37.3 0.8155									(5,7,4,6,8,9) 49.9 0.9675
(6,6,4,6,8) 37.4 0.8156									(6,6,4,6,8,9) 50.0 0.9675
(6,7,4,6,8) 38.7 0.8158									(6,7,4,6,8,9) 51.3 0.9675
(7,7,4,6,8) 40.1 0.8159									(7,7,4,6,8,9) 52.7 0.9676

4. Conclusions

The algorithm developed in this paper can be easily used by logistic practioners who do not possess highly mathematical backgrounds to balance the number of spare parts for the components of their weapon systems. Additional research might be fruitful about the development of a computer program to automate the allocation algorithm. A computer program for implementing the algorithm would be easy to develop if the program is to ignore the computational advantages offered by the "elimination rules". A program which systematically searches for an opportunity to eliminate blocks of possibilities from consideration would be more efficient; albeit more difficult to develop.

References

1. Aggarwal, K.K., "Redundancy Optimization in General Systems," IEEE Trans. Rel., Vol. R-25, pp.330-332, Dec. 1976.
2. Barlow, Richard E. and Frank Proschan, Statistical Theory of Reliability and Life Testing, Holt, Rinehart and Winstion, New York, 1975.
3. Bellman, R.E. and S.E. "Dreyfus, "Dynamic Programming and Reliability of Multicomponent Devices," Opns. Res., Vol. 6, pp.200-206, May 1958.
4. Burton, R.M. and G.T. Howard, "Optimum System Reliability for a Mixed Series and Parallel Structure," Journal of Mathematical Analysis and Applications, Vol. 28, No. 2, pp.370-382, Nov. 1969.
5. Everett H., "Generalized Lagrange Multiplier Method for Solving Problems of Optimal Allocation of Resources," Opns. Res., Vol. 11, pp.339-417, 1963.
6. Ghare, P.M. and R.E. Taylor, "Optimal Redundancy for Reliability in Series System." Opns. Res., Vol. 17, pp.838-847, Sep. 1969.
7. JEE, M.W., "Mathematical Models for Operational Availability," Phd. Dissertation, U.S. Naval Postgraduate School, Sep. 1980.
8. Kettelle, J.D., "Least-Cost Allocation of Reliability Investment," Opns. Res., Vol. 10, pp.249-265, May 1967.
9. Kuo, W., C.L. Hwang, and F.A. Tillman, "A Note on Heuristic Methods in Optimal System Reliability," IEEE Trans. Rel., Vol. R-27, No. 5, pp.320-324, Dec. 1978.
10. Mizukami, K., "Optimum Redundancy for Maximum System Reliability by the Method of Convex and Integer Programming," Opns. Res., Vol. 16, pp.392-408, Mar.-Apr. 1968.
11. Moskowitz, F. and J.B. McLean, "Some Reliability Aspects of System Design," IRE Trans. Rel. and Qual. Contr., Vol. PGRQC-8, pp.7-35, Sep. 1956.
12. Nakagawa, Y. and K. Nakashima, "A Heuristic Method for Determining Optimal Reliability Allocation," IEEE Trans. Rel., Vol. R-26, No. 3, pp.156-161, Aug. 1977.
13. Proschan, F. and T.H. Bray, "Optimal Redundancy Under Multiple Constraints," Opns. Res., Vol. 13, pp.800-814, Sep. 1965.
14. Sharma, J. and K.V. Venkateswaran, "A Direct Method for Maximizing the System Reliability," IEEE Trans. Rel., Vol. R-20, pp.256-259, Nov. 1971.
15. Sherbrooke, C.C., "METRIC: A Multi-Echelon Technique for Recoverable Item

Control," Opns. Res., Vol. 16, pp.122-141, 1968.

16. Srinivasan, V.S., "The Effect of Standby Redundancy in System's Failure with Repair Maintenance," Opns. Res., Vol. 14, pp.1024-1036, Nov. 1966.

17. Tillman, E.A., and J.M. Liittschwager, "Integer Programming Formulation on Constrained Reliability Problems," Mgt. Sci., Vol. 13, pp.887-899, July 1967.