

Technical Report No. 140

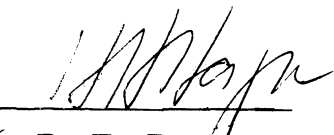
4967-1-T

COMPUTATIONAL TECHNIQUES FOR SCHEDULING
PROBLEMS WITH DEFERRAL COSTS

by

E. L. Lawler

Approved by:


for B. F. Barton

for

COOLEY ELECTRONICS LABORATORY

Department of Electrical Engineering
The University of Michigan
Ann Arbor

Project 04967

Contract No. AF 30(602)-2661
Rome Air Development Center
Griffiss Air Force Base
New York

April 1964

TABLE OF CONTENTS

	Page
LIST OF ILLUSTRATIONS	iv
ABSTRACT	v
1. INTRODUCTION	1
2. KNOWN RESULTS	3
3. DYNAMIC PROGRAMMING TECHNIQUES	4
4. TRANSPORTATION METHODS, UNIFORM PROCESSING TIMES	6
5. TRANSPORTATION METHODS, NONUNIFORM PROCESSING TIMES	9
6. LINEAR PROGRAMMING TECHNIQUES	12
7. CONCLUSIONS	16
REFERENCES	17
DISTRIBUTION LIST	18

ABSTRACT

A class of scheduling problems involving deferral costs has been formulated by McNaughton, who has described a simple method of solution for the linear, single-processor case. In this report dynamic programming and linear programming techniques are applied to nonlinear and multiple-processor problems. A dynamic programming solution of the nonlinear, single-processor problem is possible, provided the number of jobs is small. Transportation methods of linear programming can be used to solve large nonlinear, multiple-processor problems, provided the processing times for the jobs are equal. Approximate and/or partial solutions are possible for other cases.

1. INTRODUCTION

This report summarizes the results of an investigation of computational techniques for scheduling problems involving deferral costs. These problems are characterized as follows: There are n jobs and m identical processors. For each job i ($i=1, 2, \dots, n$), there is a fixed processing time a_i that does not depend upon which processor performs the job, or upon which jobs precede or follow the job on the same processor. Also associated with job i is a deferral cost $c_i(t)$, which is assumed to be monotonically nondecreasing with t , the time of completion of the job. It is desired to find a schedule for the jobs, with completion times t_1, t_2, \dots, t_n such that the total deferral cost,

$$\sum_{i=1}^n c_i(t_i),$$

is as small as possible.

Deferral costs may be associated with "relative" and "absolute" deadlines, as indicated in Fig. 1. These problems have been studied in this context by McNaughton (Ref. 1), who describes a simple procedure for finding a minimal schedule, subject to the very severe restrictions

- (a) there is only one processor, i. e. , $m = 1$.
- (b) the deferral costs are all linear, i. e. , $c_i(t) = p_i t$, for
some $p_i \geq 0$.

McNaughton also proves a basic theorem concerning the scheduling of multiple processors, but concludes that this more general problem is quite difficult. These results are summarized in Section 2 of this report. The remaining sections describe the application of dynamic programming and linear programming methods to nonlinear and multiple-processor problems.

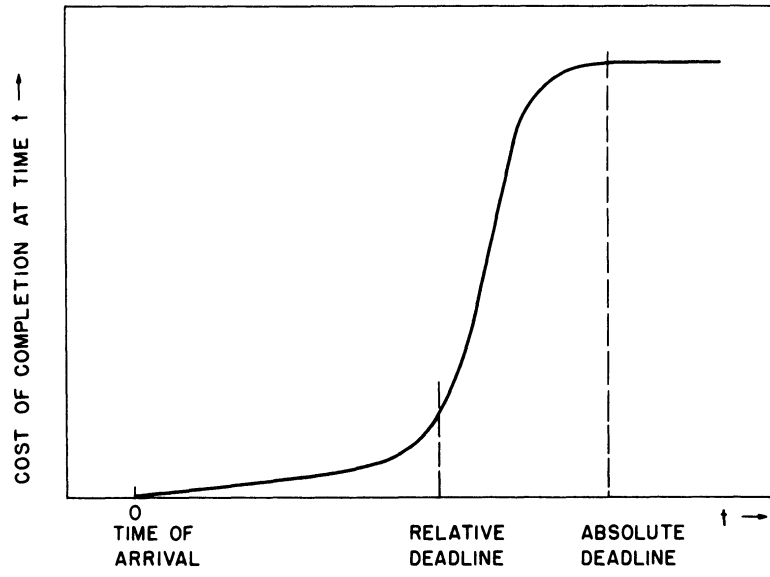


FIG.1. DEFERRAL COST FUNCTION

2. KNOWN RESULTS

The most important results of McNaughton (Ref. 1) can be summarized as follows: First, in the case of a single processor and linear deferral costs, where

$$\left. \begin{array}{l} c_i(t) = p_i t, \\ p_i \geq 0, \end{array} \right\} \quad (i = 1, 2, \dots, n)$$

jobs should be ordered according to the ratios p_i/a_i , with the job having the largest ratio being performed first.

Second, in the case of multiple processors and linear deferral costs, if S is a schedule with at least one "split," but only finitely many, then there is a schedule for the same set of jobs with no splits and with a total deferral cost that is no greater than that of S . A split is defined as an interruption in the execution of a job, or the shifting of a job from one processor to another.

3. DYNAMIC PROGRAMMING TECHNIQUES

We shall now develop a dynamic programming method for solving single-processor problems with deferral costs that are nonlinear but monotone nondecreasing with time.

If deferral costs are monotone nondecreasing with time, it follows that there exists an optimal schedule in which each job is followed immediately by the next job, i. e., there is no "idle time." Thus, if jobs 1, 2, 3, . . . , k are performed by the single processor before any others, the processor is busy with these jobs from $t = 0$ to $t = d$, where

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

This is true without regard to the order in which the jobs are performed.

Let I denote the complete set of jobs, as represented by their indexes, i. e., $I = \{1, 2, \dots, n\}$, and let J denote an arbitrary subset, e. g., $J = \{1, 3, 5\}$. Then let $d(J)$ be defined as follows:

$$d(J) = \sum_{i \notin J} a_i .$$

It is seen that $d(J)$ represents the length of time that the processor is busy performing those jobs not included in the subset J .

Suppose we are given a schedule, and that the first k jobs performed under that schedule are represented by $I-J$ and the remaining $n-k$ jobs by J . Bellman's Principle of Optimality (Ref. 2) may now be cited: If the schedule is optimal, it has the property that, regardless of the order in which the first k jobs are performed, the remainder of the schedule constitutes an optimal schedule of the jobs J , subject to the restriction that none of these jobs is commenced before $t = d(J)$.

We now let $C(J)$ denote the minimal cost of performing the jobs J , subject to the restriction that no job is commenced before $t = d(J)$. C is a well-defined function from the set of all subsets of I to the set of real numbers. The solution of the problem involves finding the value of $C(I)$ and the schedule of jobs that yields this value as its cost.

With some reflection, it becomes apparent that

$$C(I) = \text{Min}_{i \in I} \left\{ c_i(a_i) + C(I - \{i\}) \right\}$$

More generally,

$$C(J) = \text{Min}_{i \in J} \left\{ c_i(a_i + d(J)) + C(J - \{i\}) \right\} ,$$

and

$$C(\emptyset) = 0 ,$$

where \emptyset denotes the empty set. These functional equations are all that is required to define a computational procedure for the solution of the single processor problem (Ref. 2).

Computational difficulty can be estimated as follows. There are 2^n distinct subsets J , where n is the number of jobs, and hence 2^n values $C(J)$ to be stored in computer memory. The number of arithmetic operations is related to $n2^n$. These figures imply that it is a simple matter with an IBM 7090 to solve problems for which $n \leq 15$, and that somewhat larger values of n are not intractable.

The dynamic programming approach can be extended to the multiple-processor problem in various ways. In doing so, it is possible to make use of the fact that a schedule is optimal only if the jobs assigned to each individual processor are sequenced as they would be for a single-processor problem. However, no method for the multiple-processor problem known to the author is sufficiently nonexhaustive to be practicable.

4. TRANSPORTATION METHODS, UNIFORM PROCESSING TIMES

We shall now develop a method for solving multiple-processor problems with nonlinear costs, subject to the restriction that the jobs have uniform processing times. This method is based on the solution of the well-known transportation problem, a special type of linear program.

One standard form of the transportation problem is as follows. Given an $r \times s$ cost matrix $C = \|\|c_{ij}\|\|$ and two sets of constants $\alpha_1, \alpha_2, \dots, \alpha_r$, and $\beta_1, \beta_2, \dots, \beta_s$, determine an $r \times s$ solution matrix $X = \|\|x_{ij}\|\|$ so as to

$$\text{minimize } \sum_{i=1}^r \sum_{j=1}^s c_{ij} x_{ij} \quad (1)$$

$$\text{subject to } \sum_{j=1}^s x_{ij} \geq \alpha_i \quad (i = 1, 2, \dots, r) \quad (2)$$

$$\sum_{i=1}^r x_{ij} \leq \beta_j \quad (j = 1, 2, \dots, s) \quad (3)$$

$$\text{and } x_{ij} \geq 0. \quad (4)$$

Transportation problems with quite large dimensions, e. g. , $r = s = 200$, can be solved efficiently by large scale digital computers like the IBM 7090. Many standard routines for this purpose exist (Ref. 3).

Let us now proceed with the formulation of the scheduling problem. We assume that all jobs have the same processing time, i. e. , $a_i = a_j = a$ for all i, j . As in the previous section, we assume that the deferral costs are monotone nondecreasing with time, so that there exists an optimal schedule in which each job is followed immediately by

the next job. Hence there will be an optimal schedule in which the time of completion of each job will be

$$t = ka$$

for some positive integer k .

It is a peculiar feature of transportation problems that there always exists an optimal solution in which the variables take on nonnegative integer values, rather than merely nonnegative real values. (This assumes all α_i, β_j are integers.) Moreover, the standard routines always obtain such solutions. We make use of this fact in the following interpretation of the solution variables. We let $r = n$ and associate all variables having row index i with the job i . Then

$$\begin{aligned} x_{ij} &> 0 \text{ if job } i \text{ is assigned the completion} \\ &\text{time } t = ja. \\ &= 0 \text{ otherwise} \end{aligned}$$

For each i , if we set

$$\sum_{j=1}^s x_{ij} \geq 1, \quad (i=1, 2, \dots, n) \quad (2a)$$

we will cause exactly one of the x_{ij} to take on the value 1. Then, the condition $x_{ij} > 0$ is equivalent to $x_{ij} = 1$, and $x_{ij} = 1$ if and only if job i is completed at time $t = ja$.

Each value of j corresponds to a possible time of completion, and we choose s to be sufficiently large to allow all jobs to be performed, i. e., s is chosen as the least integer not less than n/m , where m is the number of processors. It is clear that no more than m jobs can be completed at the same time, since there are only m processors. Hence we require

$$\sum_{i=1}^n x_{ij} \leq m, \quad (j=1, 2, \dots, s) \quad (3a)$$

Finally, we must represent the deferral costs by appropriate cost coefficients c_{ij} . This is quite simple, i. e.,

$$c_{ij} = c_i(ja), \quad (i=1, 2, \dots, n; j=1, 2, \dots, s)$$

where $c_i(t)$ is the original nonnegative, monotone nondecreasing cost function for job i .

5. TRANSPORTATION METHODS, NONUNIFORM PROCESSING TIMES

Let us consider the possible extension of transportation methods to multiple-processor problems in which the jobs have nonuniform processing times.

Suppose all the processing times are integral values. We can then divide job i into a_i unit jobs, each having a processing time of one time unit. We then assign the following interpretation to the solution variables x_{ij} :

$$\begin{aligned}
x_{ij} &= 1 && \text{if one unit of the job } i \text{ is assigned} \\
&&& \text{completion time } t = j . \\
&= 0 && \text{otherwise.}
\end{aligned}$$

Since a_i units of job i must be performed, we have, instead of Eq. 2a,

$$\sum_{j=1}^s x_{ij} \geq a_i, \quad (i=1, 2, \dots, n)$$

The constraint Eq. 3a remains the same, since it is possible to perform only m unit jobs simultaneously.

A schedule without splits will be obtained if and only if the i th row of the solution matrix $X = \|x_{ij}\|$ takes on the form

$$(0, 0, \underbrace{1, 1, \dots, 1}_{a_i \text{ times}}, 0, 0) . \quad (5)$$

It is not possible to enforce this restriction within the framework of the transportation problem, and therein lies the difficulty of the method. It is, however, possible to insure that the variables x_{ij} take on only the values 0 and 1, and not larger integer values. This is done by applying the constraint

$$x_{ij} \leq 1, \quad (i=1, 2, \dots, n; j=1, 2, \dots, s) . \quad (6)$$

A transportation problem with a constraint like Eq. 6 is said to be "capacitated." There are standard methods of solution for capacitated problems (Ref. 3), and these are virtually as efficient as those for the noncapacitated case.

We have yet to determine a suitable set of cost coefficients. Suppose these are determined in accordance with the following constraint:

$$c_{ij} = c_i(j) - \sum_{k=j-a_i+1}^{j-1} c_{ik} . \quad (i=1, 2, \dots, n; j=1, 2, \dots, s) \quad (7)$$

Then, any solution satisfying Eq. 5 will have the same cost as the deferral cost of the corresponding schedule. For any given i such that

$$\begin{aligned} x_{ij} &= 1, & j_i - a_i + 1 \leq j \leq j_i \\ &= 0 & \text{otherwise} \end{aligned}$$

it follows that

$$\begin{aligned} \sum_{j=1}^s c_{ij} x_{ij} &= \sum_{j=j_i - a_i + 1}^{j_i} c_{ij} x_{ij} \\ &= \sum_{j=j_i - a_i + 1}^{j_i - 1} c_{ij} x_{ij} + c_{ij_i} x_{ij_i} \\ &= c_i(j_i). \end{aligned}$$

One can now adopt the following strategy: Set up a capacitated transportation problem with the cost coefficients determined in accordance with Eq. 7. Solve the problem and see if the solution satisfies Eq. 5. If it does, an optimal schedule has been obtained. If it does not, the value of the cost function at least gives a valid lower bound on the cost of an optimal solution in which there are no splits. One can then try a new set of cost coefficients, perhaps one designed to enhance the possibility of satisfying Eq. 5. Because transportation problems are easily solved, one can afford to make several attempts of this type.

6. LINEAR PROGRAMMING TECHNIQUES

In this section we describe a method that may be useful for verifying the optimality of a given multiple-processor schedule, if all deferral costs are linear.

Assume for the moment that all processing times are equal, i. e. , $a_i = 1$ for $i = 1, 2, \dots, n$. Then each job is completely characterized by its nonnegative cost coefficient p_i , where

$$c_i(t) = p_i t .$$

We suppose that no two coefficients are equal (the coefficients may always be perturbed infinitesimally to satisfy this requirement), and we let $r(p_i)$ denote the rank of p_i . For example, if

$$p_2 > p_4 > p_3 > p_1 ,$$

then

$$r(p_2) = 1 , \quad r(p_4) = 2 , \quad r(p_3) = 3 , \quad r(p_1) = 4 .$$

An optimal schedule is obtained by the rule:

Job i is scheduled for completion at $t=j$ if and only if

$$m(j-1) < r(p_i) \leq mj . \tag{8}$$

This rule says merely that all jobs of rank 1 through m must be scheduled for the first time interval, those of rank $m+1$ through $2m$ for the second time interval, etc. An equivalent form of this rule is as follows:

Job i is scheduled for completion at $t=j$ if and only if

$$s_{j-1} \geq p_i \geq s_j, \quad (9)$$

where s_j is any value such that

$$s_j \leq \min_i \{p_i \mid r(p_i) \leq mj\},$$

$$s_j \geq \max_i \{p_i \mid r(p_i) \geq mj+1\}$$

Now let us consider the case in which all processing times are not equal. We assume that each a_i is an integral value and so divide job i into a_i unit jobs, as in the previous section. We assign cost coefficients $q_{i1}, q_{i2}, \dots, q_{ia_i}$ to the unit jobs obtained from job i , subject to the constraint

$$\sum_{k=1}^{a_i} q_{ik} = p_i. \quad (10)$$

The satisfaction of Eq. 10 insures that for any t ,

$$\sum_{k=0}^{a_i-1} q_{ik}(t+k) = p_i t + \text{constant}.$$

Hence, if the unit jobs obtained from job i are performed in consecutive time intervals, the total of the costs of the unit jobs will differ from the deferral cost of the original job only by a constant.

Suppose one is able to divide a set of jobs into unit jobs and to assign cost coefficients q_{ik} to these unit jobs in such a way that (1) constraint Eq. 10 is satisfied, and (2) an optimal schedule of the unit jobs by Eq. 8 or Eq. 9 places all the unit jobs of each of the original jobs in consecutive time intervals. (The coefficients q_{ik} take the place of p_i in these rules.) The result will then correspond to an optimal schedule for the original problem.

There does not always exist a set of coefficients q_{ik} that will yield a solution in this way. Nor does there appear to be a simple method to determine such coefficients

when they do exist, so that they may be used to set up an optimal schedule. It is, however, possible to set up a linear programming problem to determine these coefficients for a given schedule, if the coefficients exist. In this way an arbitrary schedule may be tested for optimality.

Suppose that a given schedule calls for job i to be completed at $t = t_i$. Let $T = \max_i t_i$. Then we desire to find nonnegative values for q_{ik} ($i=1, 2, \dots, n; k=1, 2, \dots, a_i$) and s_j ($j=1, 2, \dots, T$), subject to

$$\sum_{k=1}^{a_i} q_{ik} = p_i, \quad (i=1, 2, \dots, n) \quad (10)$$

$$s_{k'} \leq q_{ik} \leq s_{k'-1}, \quad (i=1, 2, \dots, n; k=1, 2, \dots, a_i) \quad (11)$$

where $k' = t_i - a_i + k$.

It is to be emphasized that the value of each q_{ik} must be nonnegative, or else the rules of Eq. 8 and Eq. 9 are not valid. If there are nonnegative values that satisfy Eq. 10 and the inequalities of Eq. 11, these values can be found by the simplex method of linear programming. If such values exist, it is known that the given schedule is an optimum one. If no such values exist, then one can make no certain statement about the optimality or nonoptimality of the given schedule.

It is feasible to solve linear programs with as many as 150 constraints. The present problem has $n + 2A$ constraints where

$$A = \sum_{i=1}^n a_i.$$

Hence, one may hope to apply this method to problems involving a moderate number of jobs, say $n = 20$ or $n = 50$.

An example for which the above method is of no use is the following two-processor problem:

i	p_i	a_i
1	12	2
2	8	2
3	3	1
4	9	2
5	6	2
6	6	2

An optimal solution consists of having jobs 1, 2, and 3 performed in that order by one processor, and jobs 4, 5, and 6 by the other. The reader may convince himself, first, that the schedule is optimal, and second, that there is no way to satisfy Eq. 10 and Eq. 11 for this schedule.

7. CONCLUSIONS

The applicability of the methods described in this report is summarized in Table I below. These methods, of course, do not begin to exhaust the computational techniques that might be applied to the present class of scheduling problems. For example, it is also possible to formulate these problems as quadratic assignment problems (Ref. 4) or as integer linear programming problems. Various methods of limited exhaustion might also be applied. However, these other approaches seem to be of questionable value. The efficient solution of large, nonlinear, multiple-processor problems, with nonuniform processing times, is for the moment beyond our grasp.

	Linear Costs	Nonlinear Costs
Uniform Processing Times	Trivial problem	Transportation method satisfactory for both single and multiple processors ($n \leq 200$).
Nonuniform Processing Times	McNaughton's method satisfactory for single processor (n unlimited); transportation method at least yields lower bound for multiple processors ($n \leq 200$); linear programming may sometimes prove optimality of a given solution ($n \leq 50$).	Dynamic programming satisfactory for single processor ($n \leq 15$); transportation method at least yields lower bound for multiple processors ($n \leq 200$).

Table I. Summary of methods.

REFERENCES

1. R. McNaughton, "Scheduling with Deadlines and Loss Functions," Management Science, Vol. 6, October 1959, pp. 1-12.
2. R. E. Bellman and S. E. Dreyfus, Applied Dynamic Programming, Princeton University Press, 1962.
3. L. R. Ford, Jr., and D. R. Fulkerson, "Solving the Transportation Problem," Management Science, Vol. 3, 1956, pp. 24-32.
4. E. L. Lawler, "The Quadratic Assignment Problem," Management Science, (to appear).

DISTRIBUTION LIST

Technical Report No. 140

1. John McLean, EMIIF (1)
RADC
Griffiss AFB
Rome, New York
2. Lt. L. W. Odell, EMICA (2)
RADC
Griffiss AFB
Rome, New York
3. Pat Langendorf, EMIIT (1)
RADC
Griffiss AFB
Rome, New York
4. Fred Dion, EMIIT (1)
RADC
Griffiss AFB
Rome, New York
5. Rome Air Development Center
Griffiss AFB
Rome, New York
Attn: RAT (1)
RAALD (1)
RAAPT (1)
RAL (1)
RALC (1)
RALS (1)
RALT (1)
RAO (1)
RAD (1)
RAS (1)
RASG (1)
ROAMA (ROAEPP-1) (4)
EMIIH (5)
EMIT (5)
EMIF (5)
EMID (5)
RASH (1)
RASS (1)
RAU (1)
RAUA (1)
RAUE (1)
RAUM (1)
RAUO (1)
RAWC (1)
RAWE (1)
RAWI (1)
RAYA (1)
RAI (1)

DISTRIBUTION LIST (Continued)

6. GEEIA
Griffiss AFB
Rome, New York
Attn: ROZMA (1)
ROZMC (1)
ROZME (1)
ROZMN (1)
ROZMCAT (1)
7. RAOL (Maj. Shields) (1)
Griffiss AFB
Rome, New York
8. RAOL (S/ L Tanner) (2)
Griffiss AFB
Rome, New York
9. AFSC
Andrews AFB
Washington 25, D. C.
Attn: SCSE (1)
SCLDE (1)
SCFRE (1)
10. Redstone Scientific Information Center (2)
U. S. Army Missile Command
Redstone Arsenal, Ala.
Attn: Chief, Document Section
11. Bureau of Naval Weapons (2)
Main Navy Building
Washington 25, D. C.
Attn: Tech. Lib. DL1-3
12. Chief, Bureau of Ships (1)
Code 454 F
Main Navy Building
Washington 25, D. C.
13. Central Intelligence Agency (1)
2430 E Street NW
Washington 25, D. C.
Attn: OCR Mail Room
14. U. S. Army Material Command (1)
Harry Diamond Laboratories
Washington 25, D. C.
Attn: AMXDO-TIB
15. Scientific and Technical Information Facility (2)
P. O. Box 5700
Bethesda, Maryland
Attn: NASA Representative (S-AK/DL)

DISTRIBUTION LIST (Continued)

16. Director (1)
National Security Agency
Ft. George G. Meade, Maryland
Attn: C3/TDL
17. Commander (1)
Naval Missile Center
Tech Library (Code No. 3022)
Pt. Mugu, California
18. Commanding Officer and Director (1)
U. S. Navy Electronic Lab (Lib)
San Diego 52, California
92152
19. Office of Chief of Naval Operations (Op-723 E) (1)
Rm. 1602
Building T-3, Navy Department
Washington 25, D. C.
20. Office of Naval Research (1)
Chief Scientist (Code 427)
Washington 25, D. C.
21. The Rand Corp. (Tech Lib) (1)
1700 Main Street
Santa Monica, California
22. Hq TAC (DORQ-S) (1)
Langley AFB
Virginia
23. Hq TAC (OA) (1)
Langley AFB
Virginia
24. Commander, TAC Comm Region (1)
Langley AFB
Virginia
25. USAFSS (ECD) (1)
San Antonio
Texas
26. Commanding General (2)
US Army Electronic Proving Ground
Attn: Tech Library
Fort Huachuca, Arizona
27. Commanding Officer (1)
US Army Electronics Rand D Lab
Attn: SELRA/ADT
Ft. Monmouth, New Jersey

DISTRIBUTION LIST (Continued)

28. Institute of Technology Library (1)
MCLI-LIB, Bldg 125 Area "B"
Wright-Patterson AFB
Ohio
29. NAFEC Library (1)
Bldg. 3
Atlantic City, New Jersey
30. Commandant (1)
US Army War College (Lib)
Carlisle Barracks, Pa.
31. Commanding Officer (1)
US Army Electronic Research Unit
P O Box 205
Mountain View, California
32. Electronic Defense Labs (1)
Attn: Library
P O Box 205
Mountain View, California
33. U S Naval Avionics Facility (Library) (1)
Indianapolis 18
Indiana
34. Commander (1)
US Naval Air Dev Cen (NADC Lib)
Johnsville, Pa.
35. Director (2)
US Naval Research Lab (Code 2027)
Washington 25, D. C.
36. Hq USAF (1)
Attn: AFRSTE (1)
AFRDPC (2)
AFGOA (1)
Washington 25, D. C.
37. National Aeronautics and Space Admin (3)
Langley Research Center
Langley Station
Hampton, Virginia
Attn: Librarian
38. RTD (RTH) (1)
Bolling AFB 25 D. C.
39. Federal Aviation Agency (1)
Information Retrieval Br Hq-630
Washington 25, D. C.



DISTRIBUTION LIST (Continued)

- 40. RTD (RTHR/ Capt. K. O. Malkemes (2)
Bolling AFB, D. C.
20332
- 41. OSD (DDR and E/ Mr. Willie Moore (2)
Washington D C
20330
- 42. Hq USAF (AFRST/ Lt. Col. A. H. Grinsted, Jr.) (1)
Washington D C
20330
- 43. Advanced Research Projects Agency (1)
Command and Control Research
(Dr. J. C. R. Licklider)
Washington, D. C.
- 44. National Security Agency (1)
Engineering Research Div, Code R42
(Mr. O. H. Bartlett, Jr.)
Fort Meade, Maryland