

AD-A152 137

SOLUTION TECHNIQUES FOR WHOLESALE PROVISIONING OF
REPLACEMENT PARTS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY
CA W A GOULDING SEP 84

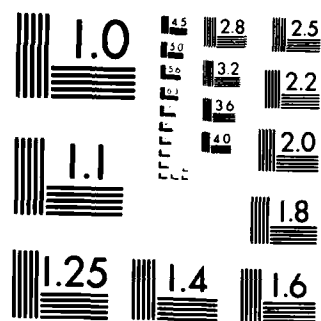
1/1

UNCLASSIFIED

F/G 15/5

NL

END



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2

AD-A152 137

DTIC FILE COPY

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

SOLUTION TECHNIQUES FOR
WHOLESALE PROVISIONING
OF REPLACEMENT PARTS

by

William A. Goulding

September 1984

Thesis Advisor:

G. T. Howard

Approved for public release; distribution unlimited.

DTIC
ELECTE

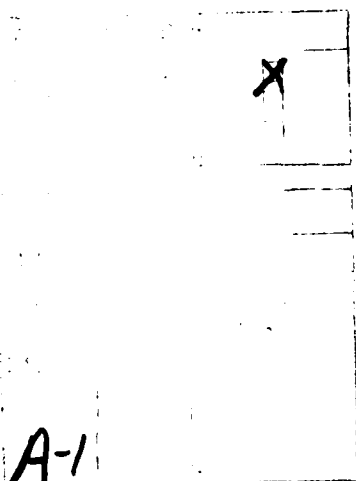
APR 05 1985

85 03 15 045

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A152137	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Solution Techniques for Wholesale Provisioning of Replacement Parts		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William A. Goulding		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943		12. REPORT DATE September 1984
		13. NUMBER OF PAGES 84
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Wholesale Provisioning, Dynamic Programming		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The purpose of this thesis is to present solution techniques for Provisioning problems arising in the Navy's wholesale purchase of replacement parts. The objective is to minimize the Mean Supply Response Time (MSRT) subject to a budget constraint. The problem can be formulated as a Dynamic Program (DP), however, it is too large and complex for a standard recursive dynamic approach. Consequently, a		

20. (Continued)

variation of the normal DP approach was developed that significantly reduces the required computations. An existing DP computer program was modified to implement this DP variation. The result is a useable approach considering speed and ease of manipulation.



Approved for public release; distribution unlimited.

Solution Techniques for
Wholesale Provisioning
of Replacement Parts

by

William A. Goulding
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
SEPTEMBER 1984

Author:

William A. Goulding
William A. Goulding

Approved by:

G.T. Howard

G.T. Howard, Thesis Advisor

F.R. Richards

F. R. Richards, Second Reader

A.R. Washburn

A. R. Washburn, Chairman,
Department of Operations Research

K.T. Marshall

K. T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

The purpose of this thesis is to present solution techniques for provisioning problems arising in the Navy's wholesale purchase of replacement parts. The objective is to minimize the Mean Supply Response Time (MSRT) subject to a budget constraint. The problem can be formulated as a Dynamic Program (DP), however, it is too large and complex for a standard recursive dynamic approach. Consequently, a variation of the normal DP approach was developed that significantly reduces the required computations. An existing DP computer program was modified to implement this DP variation. The result is a usable approach considering speed and ease of manipulation.

TABLE OF CONTENTS

I.	HISTORY AND INTRODUCTION	8
II.	SOLUTION TECHNIQUES	12
	A. MARGINAL ANALYSIS	14
	B. STANDARD DYNAMIC PROGRAMMING	18
	C. A DYNAMIC PROGRAMMING VARIATION	23
III.	THE FUNNEL PROGRAM	27
	A. MODIFICATION FOR THE FUNNEL PROCEDURE	27
	B. THE SUBROUTINE ADJUST	32
	C. IMPLEMENTATION PROBLEMS	39
IV.	ANALYSIS	43
V.	CONCLUSIONS	54
	APPENDIX A: DP6 CODE	57
	LIST OF REFERENCES	83
	INITIAL DISTRIBUTION LIST	84

LIST OF TABLES

I.	Sample Output from the Marginal Analysis Program	17
II.	Sample Problem Input Parameters	43
III.	Marginal Analysis Program Return, Budget=\$195.00	44
IV.	Funnel Program Results, Budget=\$195.00	45
V.	Marginal Analysis Program Return, Budget=\$200.00	46
VI.	Funnel Program Results, Budget=\$200.00	47
VII.	Marginal Analysis Program Results, Budget=\$205.00	48
VIII.	Funnel Program Results, Budget=\$205.00	49
IX.	Kit Function Results, Budget=\$205.00	50
X.	Input Parameters for a Larger Problem	51
XI.	Marginal Analysis Program Results, Budget=\$74825.00	52
XII.	Funnel Program Results, Budget=\$74825.00	53

LIST OF FIGURES

2.1	Example of a One-Stage Decision Problem	18
2.2	Example of an N Stage Decision Problem	20
3.1	Flow Diagram of the DP5 Procedure	28
3.2	Flow diagram of the DP6 procedure	30
3.3	Example of DP6 Output	31
3.4	Example of a Window on a Stage i	33
3.5	A Funnel Diagram for N Stages of a Problem . . .	35
3.6	The Initialization Run	36
3.7	The Second Run	37
3.8	The Window Adjustments	38
3.9	The Third Run	38
3.10	Boundary Manipulation	40

I. HISTORY AND INTRODUCTION

The U. S. Navy is a hardware oriented organization requiring a wide variety of spare and replacement parts. The Naval Supply Systems Command (NAVSUP) is the organization within the Navy responsible for maintaining a sufficient stock of parts and replenishing this stock when quantities run low. NAVSUP is keenly interested in the quality of the models it uses to provide support for the Navy's weapons systems. Faculty of the Naval Postgraduate School were asked to examine the existing wholesale and retail provisioning and replenishment models and to provide comments on their correctness and suggestions for possible improvements. In response, a report by Richards and McMasters [Ref. 1] addresses the problems and various solution techniques. This thesis examines a specific solution technique that may be applicable to one of the problems. The following is a brief synopsis of the existing models and the environment in which they operate.

Within NAVSUP there are two centers that use provisioning models, 1) Ships Parts Control Center (SPCC) and 2) Aviation Supply Office (ASO). Presently the Inventory Control Points (ICPs) within the centers have two retail provisioning models which generate, 1) COSALS (Coordinated Shipboard Allowance Lists) and 2) AVCALS (Aviation Coordinated Allowance Lists). These models supply sufficient parts support for a short term ship deployment and try to provide protection from replacement part depletion until the wholesale replenishment models can buy or repair the needed spare parts.

Weapon system procurement requires that provisioning support be available by a set preliminary operational

capability (POC) date. The initial support requirements are determined by a wholesale provisioning model. The support must be comprehensive enough to supply a new weapon system until the replenishment buy can be made and delivered. It should be noted that the provisioning lead time is usually two to two and one half years. Variability in production and installation causes the quality of the forecasted demand to be poor. Hence, the provisioning buy may result in large quantities of excess parts in the supply system, or there may be many stockouts before the replenishment buy becomes available. The Assistant Secretary of Defense (Installations and Logistics) has provided guidance [Ref. 2] to help prevent large excesses. From this instruction a third model (COSDIF) was developed to conservatively estimate the quantities of new parts required for wholesale provisioning. For existing equipment no additional stock is procured for the new weapons system. If the increase in demand from the new system is large enough, the inventory manager can anticipate the increase and start a replenishment buy in sufficient time to meet the need. Otherwise, inventory optimization models may be used to control the stock. Wholesale replenishment for a new system is initiated at different times by the two centers within NAVSUP. For either center the reorder point for replacement parts is subject to change as a function of the actual demand the weapon system places on the supply system. The reorder point for an existing system occurs at part depletion levels set by NAVSUP.

Budgeting constraints for support of existing systems are partially driven by the peculiarities of the Department of Defense Planning, Programming and Budgeting System (PPBS). Those purchases for new weapon systems are funded as part of the procurement process. For existing systems, each Service requests from Congress sufficient funding to

meet specific levels of readiness and availability set by the respective Service Head. NAVSUP receives this direction from the Chief of Naval Operations (CNO) and then prioritizes the parts required with a procedure called "Variable Threshold". This technique considers several inputs peculiar to the weapon system to develop the priority list for items to be purchased. The quantity (depth) to be purchased is determined from historical data or from engineering estimates of the number of anticipated failures for a period of time equal to one procurement lead time plus one quarter. Proceeding from the top of the priority list the parts are purchased to their 'depth' until the budget is expended. Those parts unfunded for purchase are placed in a file for Unfunded Requirements.

Obviously, the optimal quantities to purchase, as a function of the inputs, and the maximum utilization of the budget are of high interest. Howard in [Ref. 3] addresses several different approaches that provide solutions for this type problem. The thrust of this thesis is a variation of one of those solution techniques for a similarly defined provisioning problem. The objective is to minimize the Mean Supply Response Time (MSRT) subject to a budget constraint.

The overall MSRT is a nonlinear composite function of the purchase quantity for all items. The individual item's MSRT functions are also nonlinear with performance as a function of four variables peculiar to the item type. The budget constraint is a linear function with constant costs for each item and a set budget limit. There are no cost advantages for large magnitude buys or item combination buys. The individual costs and return functions of the MSRT lend themselves to formulation as a dynamic programming (DP) problem. However, the problem is large and complex for a standard recursive dynamic approach due to the required number of stages, the magnitude of the budget and the range

of costs. Consequently, a variation of the normal DP approach was developed that significantly reduces the required computations. This variation shall be referred to as 'the funnel' because of its funneling characteristics for computation of the resource variable bounds. Further explanation of its construction and procedure may be found in Chap 2&3. Modifications of an existing DP computer program described in [Ref. 3] to operate as the DP variation makes the 'funnel' method a usable approach considering speed and ease of manipulation. Additionally, the DP variation has the advantage of providing an optimal integer solution. In this report 'the funnel' will be compared to a full DP procedure and a computerized marginal analysis approach which operates very efficiently but which may not generate the optimal solution.

II. SOLUTION TECHNIQUES

This chapter will discuss three methods for solving the provisioning problem described in Chapter 1, 1) marginal analysis, 2) pure dynamic programming, and 3) a modified dynamic programming procedure. The first technique provides a quick solution that is a good estimate but may not be optimal. The second technique provides a global optimal solution but requires a large amount of computer time for any moderate sized problem. The third technique provides a local, and possibly a global, optimal solution and requires significantly less time than option two.

The following definitions will be used in the formulation and discussions in this chapter. Capital letters represent the variable while lower case letters indicate a subscript. In the expression X_{i-1} the quantity $i-1$ is the subscript although it appears on the same line as the symbol X .

N	= the total number of items considered for provisioning
B	= pre-assigned maximum budget amount
C_i	= the unit cost of item i
E_i	= the essentiality code for item type i
K_i	= the fixed time required to satisfy a requisition for item i if stock is available
L_i	= the demand rate for item i (λ_i)
S_i	= the decision variable for the number of items of type i
T_i	= the procurement lead time for item i
$P^i(S_i)$	= the probability of S_i demands for item i during the provisioning interval

$P_i(S_i)$ = the probability of S_i or fewer demands
for item i during the provisioning
interval

$Z_i(S_i)$ = the performance measure for item i when
 S_i units are stocked.

The objective of this problem is to minimize the Mean Supply Response Time (MSRT) as a function of S_i . MSRT is a widely used measure of supply performance because of its role in determining availability and because it is an indicator of the success of a supply system in meeting response time goals. It considers two major factors in computation,

- 1) the likelihood of satisfying demands from stock on hand and
- 2) the length of the delay in satisfying demands when the system runs out of stock.

It may be represented by the following expression,

$$MSRT = \min \sum_{i=1, \dots, N} E_i \cdot L_i \cdot T_i \cdot Z_i(S_i) / \sum_{i=1, \dots, N} E_i \cdot L_i \cdot T_i \quad (2.1)$$

where the likelihood adjustments for conditions one and two are factors in the $Z_i(S_i)$ function. This function represents the performance measure of item i when S_i items are considered and is defined by the following equation for the case where demands during the provisioning leadtime are assumed to be Poisson distributed:

$$Z_i(S_i) = K_i + ((1 - P_i(S_i)) \cdot (L_i \cdot T_i - 2S_i + S_i \cdot (S_i + 1)) / L_i \cdot T_i) + P_i(S_i) \cdot (L_i \cdot T_i - S_i) / 2L_i. \quad (2.2)$$

The constraint for this problem is the budget or the resource in dollars. It is linear and may be represented by the following inequality,

$$\sum C_i S_i \leq B \quad i=1, \dots, N.$$

(2.3)

The following subsections present solution techniques which are viable procedures for solving the above formulated provisioning problem.

A. MARGINAL ANALYSIS

The marginal analysis method is a quick, simple and effective approach for generating a solution to the previously defined provisioning problem. For the problem considered here, the marginal analysis approach generates solutions which may not be optimal. However, the solutions generated by the procedure do possess some very desirable properties which are stated below.

Let $Z_i(S_i)$ be the essentiality weighted MSRT for item i when S_i units are stocked. Let $\Delta Z_i(S_i) = Z_i(S_{i-1}) - Z_i(S_i)$ be the reduction in essentiality weighted MSRT achieved by increasing the stockage level for item i from S_{i-1} units to S_i units. Let $S' = (S_1, S_2, \dots, S_N)$ be the vector of allocations for the N items: i.e. S_i is the number of units allocated to item i . The marginal approach begins with a budget of B dollars and an allocation vector $S' = (0, 0, \dots, 0)$. It computes $\Delta Z_i(1)/C_i$ for $i=1, 2, \dots, N$ and selects that item for which this marginal benefit to cost ratio is greatest. Without loss of generality, assume that the item selected is item 1. This requires an expenditure of C_1 dollars and results in the allocation vector $S' = (1, 0, 0, \dots, 0)$. The procedure next computes $\Delta Z_1(2)/C_1$, the marginal benefit to cost ratio achieved by increasing the stockage for item 1 by one unit from a level of one to two units. The maximum value is then selected from among the ratios in 2.4. An additional unit of the item, say item j , for which the maximum occurs is selected and the remaining budget is

$$\frac{\Delta z_1(2)}{C_1}, \frac{\Delta z_2(1)}{C_2}, \frac{\Delta z_3(1)}{C_3}, \dots, \frac{\Delta z_N(1)}{C_N} \quad (2.4)$$

decremented by C_j dollars. The process continues in this manner allocating a single unit iteratively until the budget is expended or no additional units can be purchased. At each step, the ratios compared are

$$\frac{\Delta z_1(S_1+1)}{C_1}, \frac{\Delta z_2(S_2+1)}{C_2}, \dots, \frac{\Delta z_N(S_N+1)}{C_N}, \quad (2.5)$$

where the current allocation vector is $S'=(S_1, S_2, \dots, S_N)$. Since only one of the ratios changes at each step of the process, the computations required are very few.

It can be shown [Ref. 4], that for the problem considered here the solution is undominated. That is, if $B(S')$ is the budget required to fund the allocation vector S' , and if S'' is any other allocation vector such that $B(S'') \leq B(S')$ then the overall essentiality weighted MSRT for S' is smaller than that for S'' . If it should happen that $B(S') = B$, the original budget, for some step of the marginal analysis procedure, then the solution S' is optimal.

In actual practice, what usually happens in the marginal analysis procedure is that at some step purchase of the item with the largest marginal benefit to cost ratio requires an expenditure of more money than that which remains. The allocation vector at that point is not feasible and various heuristics are usually applied to continue the process. The most popular heuristic is to backtrack to the previous feasible solution and select that item which has the maximum benefit to cost ratio among those items with unit costs no larger than the budget remaining. On applying such heuristics, one can not claim that the final solution so obtained is undominated. However, one can make use of the undominated property to obtain a bound on the optimal value of the objective function.

Let $Z(SS)$ be the objective function value at the occasion at which the marginal analysis procedure first generates a solution for which the budget required is equal to or greater than B . If equality exists, SS is optimal. Suppose $B(SS) = BS > B$ and let $S^*(X)$ be the optimal solution for a budget of X dollars. Then, since SS is undominated, $Z(SS) \leq Z(S^*(BS)) \leq Z(S^*(B))$, which is the value of the objective function at the optimal solution. Thus $Z(SS)$ is a lower bound on the optimal value of the objective function. One can thus assess the potential benefit of a more extensive search for the optimal solution by comparing the actual performance obtained at any stage with the lower bound discussed above.

A computer program was written by Richards to implement this method. A sample of this Marginal Analysis program output in table I shows the iteration number, which items are purchased, the numbers of items purchased, the benefit to cost ratios and the budget remaining. The table shows the sequence in which the items are purchased. Note the decreasing benefit to cost ratio. The program selects item number one five times in a row before the first unit of item two provides a better marginal return. The last attempt to purchase chooses an item that drives the remaining budget negative. The program automatically re-evaluates the marginal returns and then chooses the next largest benefit to cost ratio which in this case drives the remaining budget to zero. As previously discussed this solution may not be optimal.

The Marginal Analysis program has several other capabilities based on the same approach. There is a job selection menu that provides a choice of 1) spares determination, 2) budget determination or 3) kit evaluation. The first option selects the quantities to be purchased within a given budget. The second option solves a related problem by

TABLE I

Sample Output from the Marginal Analysis Program

```
*****
*   METHOD: FIXED      BUDGET =    205.00   *
*****
```

```
*****
STEP  ITEM  SI      RATIO      BR
1      1      1  0.480808496    200.00
2      1      2  0.365659833    195.00
3      1      3  0.260618091    190.00
4      1      4  0.172421157    185.00
5      1      5  0.105280340    180.00
6      2      1  0.0735758543    170.00
7      3      1  0.06000002967    155.00
8      3      6  0.0591956675    150.00
9      3      2  0.05333369593    135.00
10     3      3  0.0466887541    120.00
11     3      4  0.0400909968    105.00
12     3      5  0.0336193480    90.00
13     1      7  0.0306577347    85.00
14     3      6  0.0273999237    70.00
15     3      7  0.0216008648    55.00
16     2      2  0.0207276568    45.00
17     3      8  0.0164023377    30.00
18     1      8  0.0146531016    25.00
19     3      9  0.0119544677    10.00
20     3     10  0.0083406679    -5.00
20     1      9  0.0064818673     5.00
21     1     10  0.0026624922     0.0
```

```
*****
MINIMIZE MSRT  EXCESS:  0.0
```

```

I      SI      MSRT
1      10     0.190
2      2     10.396
3      9     12.176
```

OVERALL 5.385

```
REALIZED PERFORMANCE = 5.3852
BOUNDS ( 6.0033, 4.3120)
```

computing the required budget to meet a specified constraint, in this case MSRT. The third option evaluates the available performance for a given decision set. The program will conduct the optimization problem with a Lagrange Multiplier or marginal analysis. In either case the answer is the same. The speed and ease of manipulation of this procedure are desirable; however, to be guaranteed an optimal solution a means of verification must be found. That is the goal of this thesis.

B. STANDARD DYNAMIC PROGRAMMING

The standard dynamic programming approach can be applied to this type of provisioning problem. The following sample problem illustrates how the concept may be applied. A single stage problem corresponding to a single item may be characterized as in fig 2.1.

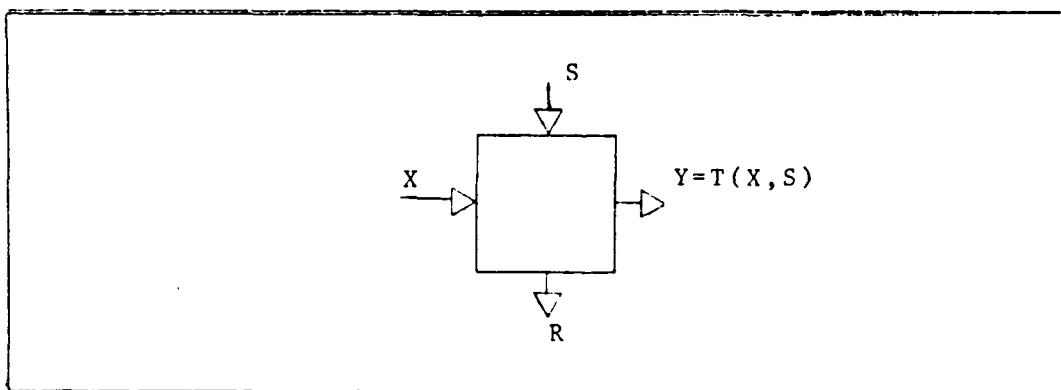


Figure 2.1 Example of a One-Stage Decision Problem.

In figure 2.1,

X is the input state variable which is the resource available for use at this stage.

S is the decision variable that represents the decision

within the stage.

R is the stage return function that produces a return based on the input X and the decision S .

Y is the output state which is a function of X and S .

T is the stage transformation function that expresses the components of the output state Y as a function of the input state and decisions, $Y=T(X,S)$.

An '*' after an X_i or S_i variable indicates the optimal value for that stage, [Ref. 5 p22-23.].

The one-stage initial-state optimization problem is to find the best stage return as a function of the input state X . The optimal return from the stage will be denoted as $F(X)$ and the optimal decision as $S^*(X)$. Thus,

$$F(X) = \min_{S(X)} R(X, S(X)) = R(X, S^*(X)), \quad (2.6)$$

where the decision $S^*(X)$ provides the best return for that particular input value X . If X were to change the values of $S^*(X)$ and the return function should also change.

Now consider a multistage problem as shown in figure 2.2. This is the standard recursive DP structure with backward numbering where stage N is on the left and the stage 1 on the right. The optimal return for stages $i, i-1, \dots, 1$ is represented by $F_i(X_i)$. The solution technique solves succeeding stages from right to left providing the optimal return $F_N(X_N)$ up to the current stage under consideration. In the single stage problem Y was the output state variable, but now Y serves as the input to the next stage so $X_{i-1} = Y = T_i(X_i, S_i)$. Thus, in an N stage problem, the output from stage i becomes the input to stage $i-1$ for $i=1, \dots, N$. In figure 2.2

X_i is the state variable

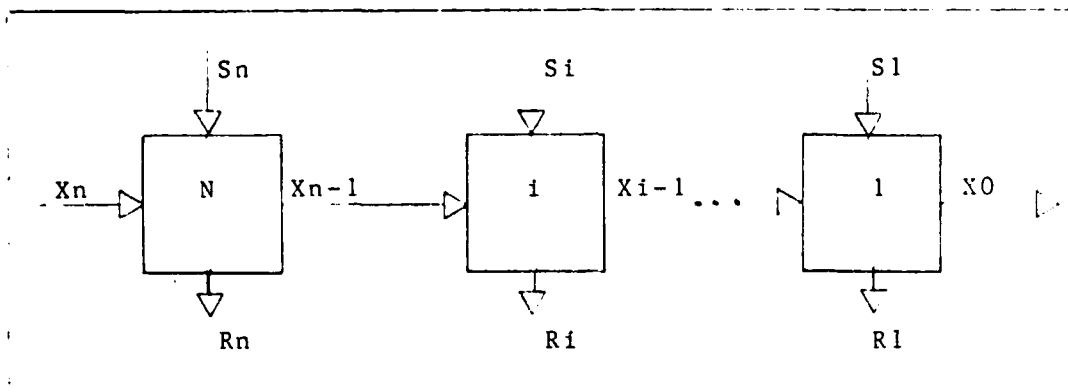


Figure 2.2 Example of an N Stage Decision Problem.

S_i is the decision variable

R_i is the stage return function

T_i is the stage transformation function

* after an X_i or S_i variable indicates the optimal value for that stage,

where $i=1, \dots, N$ is the stage index. For each stage a decision $S_i^*(X_i)$ will be made that yields the optimal return $F_i(X_i)$ as a function of the present stage return $R(X_i, S_i^*(X_i))$ and the optimal return $F_{i-1}(X_{i-1})$ for the stages 1 through $i-1$. The recursive equation for a minimization procedure would be,

$$F_i(X_i) = \min_{S_i} (R_i(X_i, S_i) + F_{i-1}(X_{i-1})) \quad i=1, \dots, N, \quad (2.7)$$

where

$$X_{i-1} = T_i(X_i, S_i) = X_i - C_i S_i \quad i=1, \dots, N. \quad (2.8)$$

The stage transformation is simply the budget minus the quantity purchased multiplied by its cost. This recursion assumes the overall return function is the weighted sum of

the individual return functions for all replacement parts. The equation for this relationship is,

$$MSRT = \sum_{i=1}^N R_i(X_i, S_i) \quad i=1, \dots, N. \quad (2.9)$$

The problem is to make a series of decisions $S_N \dots S_1$ that will optimize the overall return function while remaining within the budget constraints.

For the previously discussed provisioning problem, using the different replacement parts as the stages, the state variable X_i is the resource, i.e., the dollars remaining at each stage. The decision variable S_i is the number of parts purchased in stage i . The stage return function is the $MSRT$ which must be computed from four operating characteristics for each part, (essentiality, procurement lead time, demand rate, performance as a function of quantity procured). The following equation represents the return function at each stage,

$$R_i(X_i, S_i) = E_i * L_i * T_i * Z_i(S_i) / _E_i * L_i * T_i \quad (2.10)$$

$$i=1, \dots, N.$$

The Standard DP approach computes the function $F_i(X_i)$ from equation 2.7 recursively for $i=1, \dots, N$ and for all possible values of X_i . At each stage i this is done by examining all feasible decisions for each X_i , computing the return for each decision, and then adding the corresponding optimal return $F_i(X_i-1)$ from the remaining stages to arrive at the optimal return for the present X_i . Then for all X_i values in the stage, the best total return is chosen and assigned to $F_i(X_i)$. At the N th stage the process is complete and $F_N(X_N)$ represents the optimal return for the problem. This process is efficient for small problems, however it becomes computationally more difficult as the

number of stages and the number of possible state variable values increase in each stage.

For an example of the computational burden, consider a very simple problem with a budget B , a constant increment K for the state variable X_i within each stage, and a max of D decisions for each state variable value. In this case the number of evaluations for one stage is equal to $(B/K) * D$. Assuming the transformation function and the recursive equation for the optimal return equation require only one addition each, then there are $2 * (B/K) * D$ additions per stage. Each state variable value will require $D-1$ comparisons, hence, there are $(B/K) * (D-1)$ comparisons per stage. The total number of operations (additions plus comparisons) required for all stages may be represented by,

$$N((2 * B * D) / K) + (B * (D-1) / K), \quad (2.11)$$

or simplified this becomes

$$(N * B / K) * (3D-1). \quad (2.12)$$

This example is actually a gross over-simplification of the computational problem. It is possible the return function could be very complex requiring many operations for each decision at each X_i value through the range of the budget. However, for a numerical comparison, consider a 25-stage problem with a budget of 74000, a state variable increment of 1000 and a maximum of 100 decisions. This leads to 553150 operations as shown in equation 2.13,

$$((25 * 74000) / 1000) * ((3 * 100) - 1) = 553,150. \quad (2.13)$$

This is the required number of additions and comparisons for a 25 stage problem. For today's high speed computer an

addition operation at machine language level takes about twenty-four machine cycles while a comparison takes eighteen. Using twenty cycles as an average and a machine speed of two million cycles per second, the above number of comparisons and additions would take approximately nine seconds. This time estimate is for an incredibly simple problem that does not take into account any of the administrative overhead, objective or constraint computations or increased calculations required by complex functions. By setting $K=10$, which is equivalent to reducing the stage increment from 1000 to 10, equation 2.14 shows that the number of operations will increase to

$$((25*74000)/10)*((3*100)-1)=55,315,000. \quad (2.14)$$

Hence, it is easy to see the limitation of standard dynamic programming. While it gives integer solutions, the computational burden is considerable even in simple problems.

C. A DYNAMIC PROGRAMMING VARIATION

The computational workload of the full dynamic programming procedure makes it undesirable for large problems. A technique that will be referred to as 'FUNNELING' because of its control of the resource boundaries between the stages, was developed and applied to a standard recursive DP computer program. The result is a shortened procedure that provides a locally optimal integer solution to problems that can be formulated in the DP construct.

For any optimal solution to an N -stage DP problem there is a series of decisions S_i^* for $i=1, \dots, N$. Corresponding to this solution there is a sequence of X_i^* values for $i=1, \dots, N$. These optimal state variable values X_i^* , represent the best quantities of resource available for

consumption at that stage and are not known in advance. If it were possible to find an approximate value for X_i^* in each stage, the optimal solution then could be localized with an iterative process. That is, rather than compute $F(X)$ for all X values there could be, a 'window' of interest, centered around the approximate X_i^* value in each stage. The computations would be performed on the limited range of X_i values in that window thereby reducing the required workload significantly. This concept of controlling computational boundaries around an approximate optimal X_i^* value is the foundation for the DP variation.

The problem initially is how to localize the regions of the best X_i . This is solved by the first run of the funnel program using a large state variable increment. The resulting X_i values are then used as an approximation to the initial X_i^* values $i=1, \dots, N$. A subroutine of the funnel program then constructs a window around the area of interest for further refinement. The first estimate of the X_i^* values is quite rough and the decisions, especially the S_i which have costs less than the initializing increment, may change significantly as the X_i are examined with a lesser state variable increment.

The second program run is the first refinement of the initial solution. Using the large increments the program will have chosen the best return available though it may not be close to the optimal value. The increased sensitivity of the reduced state variable increment will allow consideration of more X_i values in the window permitting the optimization process to move closer to the best return. It is conceivable that any X_i may change as much as the window of its stage will allow. This much variation in the decisions indicates that the best solution lies outside the window and the boundaries must be changed accordingly. This situation may be present in several stages for any program iteration.

Due to the characteristics of the DP procedure, a change in the state variable consumed at stage N will cause the quantity of resource available at stage N-1 and succeeding stages to change. The concept is that repetitive runs of the funneling program will allow the computational bounds for each stage to localize and center upon their respective X_i^* . When each of the stages has chosen its X_i^* , the S_i^* for $i=1, \dots, N$ will represent the best solution for a given objective and budget. This solution will be a local and possibly a global optimum. Further discussion of the program specifics may be found in Chap 3.

Additional computations may be saved in the funneling program by ranking the stages, largest to smallest, by the cost of their respective parts. Purchasing the most expensive items first will reduce the magnitude of the budget and simplify the computations in the remaining stages. The amount of computation depends on three things, 1) the width of the window for each stage, 2) the number of state variable increments within the window and 3) the number of decisions available for each state variable increment.

Consider again the computational example of section B, where the number of operations is estimated for the dynamic programming solution procedure. To use that analysis for the funnel procedure, B in equation 2.12 is replaced by the width of the window W. This width will vary between stages but this example will assume W_i is constant and equal to 10000. Setting the remaining factors the same, $K=10$, $S=100$, and $N=25$, the number of operations required from equation 2.12 is,

$$((25*10000)/10) * ((3*100) - 1) = 7,475,000. \quad (2.15)$$

This is roughly 13.5% of the full DP requirement computed in equation 2.11, a marked improvement in efficiency. There

are other ways to make the algorithm more efficient and save computations. Further discussion of the details of the modified DP procedure may be found in chapter 3.

III. THE FUNNEL PROGRAM

This chapter discusses the operation of the standard DP program, the modifications required to employ the funneling concept and some of the implementation problems encountered in the process. Specific subroutine operations are explained in the second section.

A. MODIFICATION FOR THE FUNNEL PROCEDURE

The DP variation, called the Funnel program, is a modified version of a standard DP computer program named DP5. The modified program will be referred to as DP6 for the sake of clarity in the following discussion. The DP5 program as described in [Ref. 3] has four subroutines, STGRET, STORE, TRANFM and DLIMIT. The following list describes what function each subroutine performs in the DP process.

- 1) STORE - enters the constants to be used in the other subroutines and the main program.
- 2) DLIMIT - defines the range of the decision values S_i for any value of X_i in stage i .
- 3) STGRET - defines and computes the return function for any values of X_i and S_i in stage i , $R_i(X_i, S_i)$.
- 4) TRANFM - defines and computes the transformation function for any values of X_i and S_i in stage i , $T_i(X_i, S_i)$.

Providing DP5 with the specific input data as required in [Ref. 3], the program will compute the optimal return by the standard recursive DP procedure. The following figure shows the flow diagram for the general solution technique.

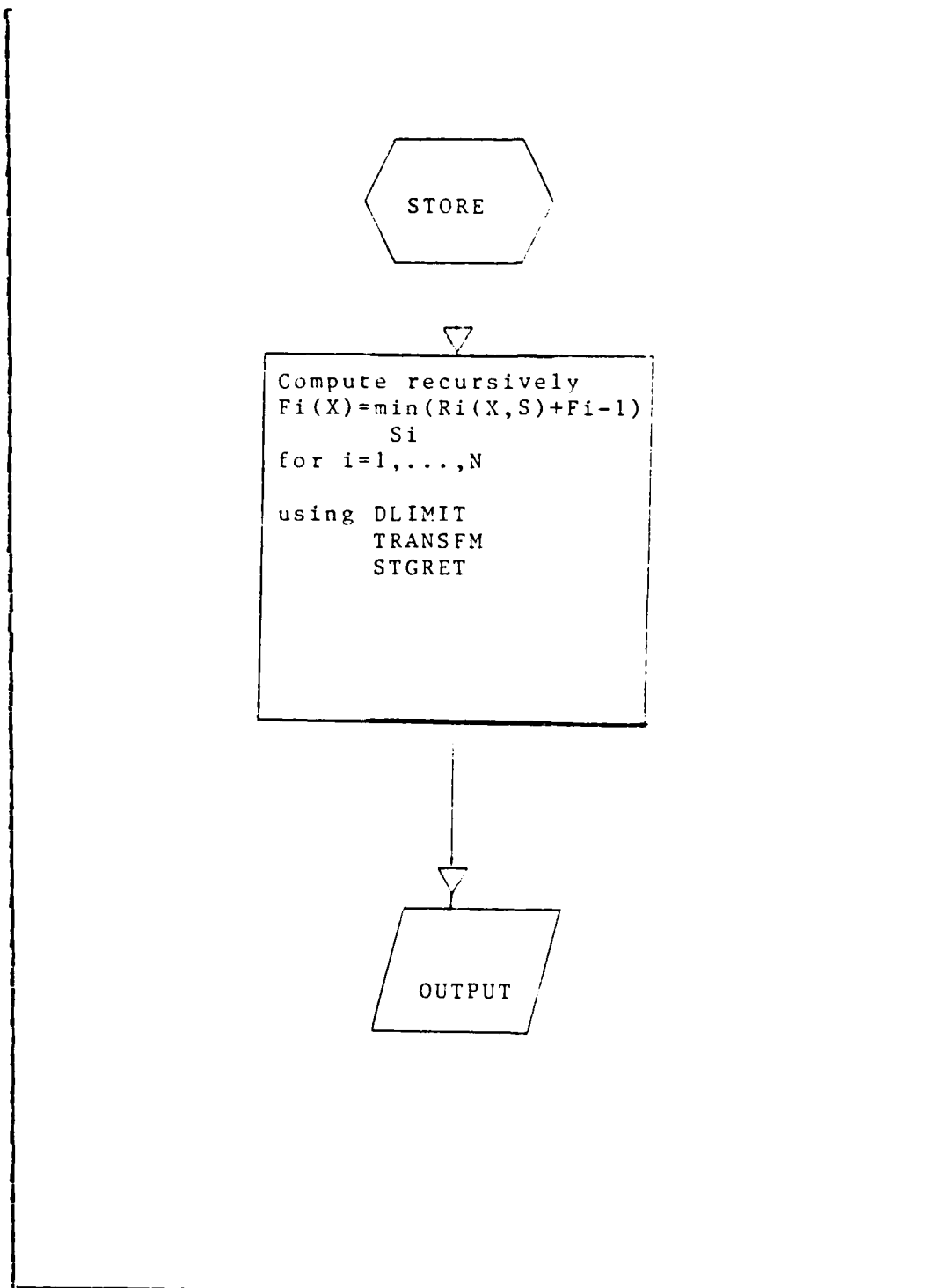


Figure 3.1 Flow Diagram of the DP5 Procedure.

The input data are the parameters for the particular problem being solved. When DP5 is run it uses the STORE subroutine to place the parameters in memory, then calls the other subroutines, DLIMIT, TRANFM, and STGRET to perform the DP procedure.

To use the concept of funneling described in Chap 2, additional subroutines must be added to DP5 to control the state variable computational boundaries in each of the stages. There are two possible starting conditions for the funnel procedure, 1) some input information is available for the decisions at each stage, or 2) no prior knowledge of the decisions is available. To respond to these conditions two subroutines ENTER and ADJUST have been created to correctly manipulate the inputs. The following list describes what function each of the subroutines performs in the Funnel program.

- 1) ENTER - reads a decision set from a file then
 creates the input data file in the correct
 format for the Funnel program,
- 2) ADJUST - manipulates the computational boundaries of
 an input data file to reflect the changes
 required at each program iteration.

The general procedure for DP6 is shown in figure 3.2, The input parameters are read into memory with the STORE function. Then, depending on the starting condition, the ENTER function may be used to create an input data file for the Funnel program from a previously available solution. This solution may be obtained from, 1) some other solution method, 2) a rule-of-thumb estimate of the solution, or from 3) past information on similar problems. In any of these cases the estimated or best guess decision set is transformed to the correct input format for the Funnel program.

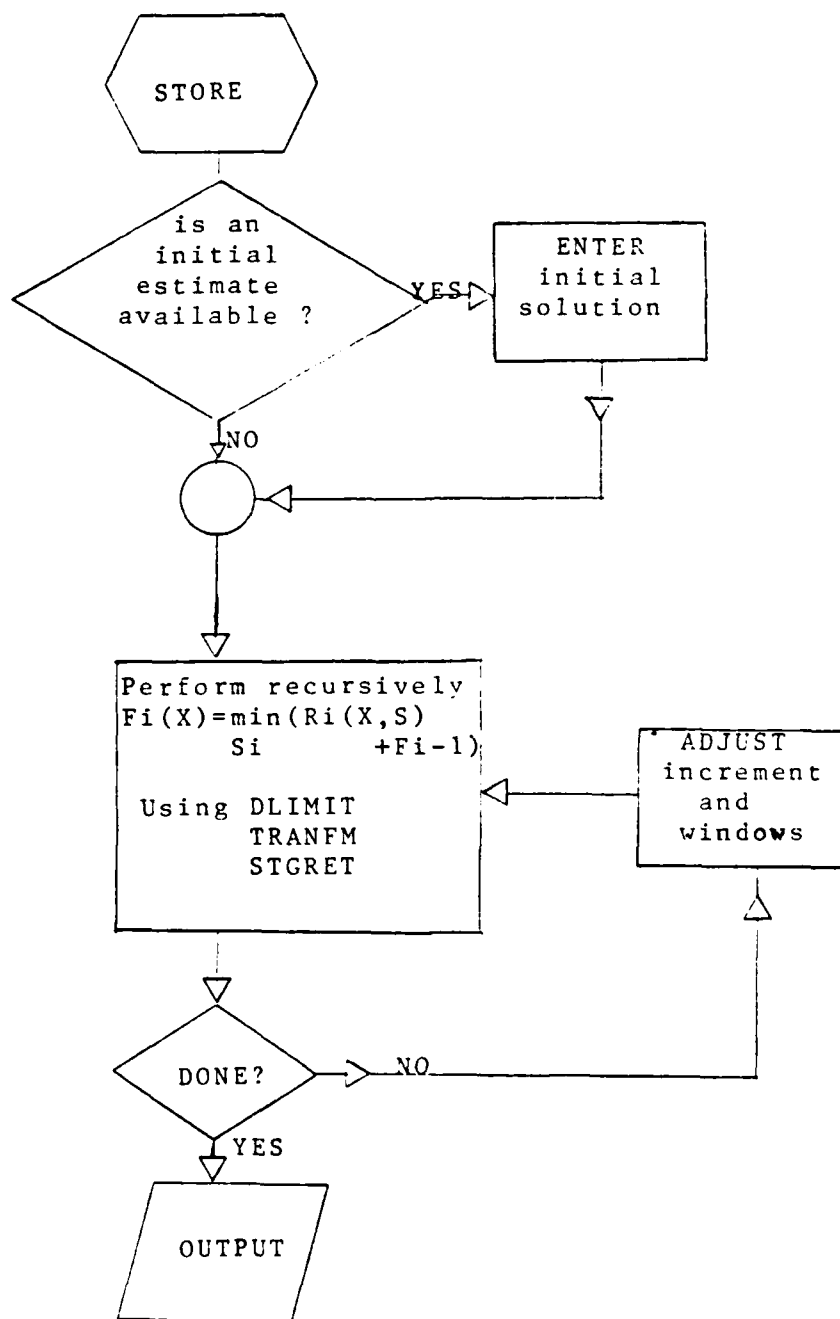


Figure 3.2 Flow diagram of the DP6 procedure.

When there is no previous solution, the beginning data file may be constructed in the correct format as defined by the user instructions located in the comment section of DP6, see appendix. This input file, as described, may be used directly with the program. The next step in the figure is the first iteration of the Funnel process utilizing the DLIMIT, TRANFM, and SIGRET subroutines. When the calculations are complete the results must be examined to see if the process has found the optimal solution, i.e., 'is it DONE'. The requirements for the 'DONE' condition are 1) an all integer solution, 2) no window boundaries violated and 3) the budget remaining must be less than the smallest cost for items to be purchased. If all of these conditions are not met the ADJUST subroutine is activated and the input data file is manipulated to correct the problem areas. With each program iteration this corrective procedure is repeated until all the conditions are met and an optimal solution is produced. An example of the Funnel program output is shown in figure 3.3.

THE OBJECTIVE IS MSRT
CONSTRAINT ON COST

THE DESCRIPTION WHICH APPEARS BELOW FOR STAGE 1
APPLIES TO STAGE 1 THRU STAGE 3 INCLUSIVE
THE PROBLEM IS TO MINIMIZE A 3 STAGE PROCESSIN
IS TO BE CHOSEN OPTIMALLY BETWEEN $XN=2.05000E+02$
AND $XN=2.0500D+02$

OPTIMAL $XN=2.05000D+02$ OPTIMAL RETURN= $5.30246D+00$

N	XN	SN	XN-1
3	$2.0500D+02$	$1.00D+01$	$5.5000D+01$
2	$5.5000D+01$	$2.00D+00$	$3.5000E+01$
1	$3.5000D+01$	$7.00D+00$	0.0

Figure 3.3 Example of DP6 Output.

The objective and constraint are stated in the first two lines. The description that follows gives the number of stages, the budget quantity, the optimal return and detailed stage results. Specifically delineated for each of the stages are the input budget quantities, the decisions and the output budget quantities. For this particular example the budget remaining at the end of the problem is zero.

Two ancillary functions were added, one to the main program and one to a subroutine to provide information to the user. The first is a timing function that provides the program run time. The second is a counting function that presents to the screen and stores in a file the numbers of the stages in which the optimal return values are too close to the computational boundaries. Both functions have provided valuable insights that will be discussed in Chapter 5.

B. THE SUBROUTINE ADJUST

The subroutine ADJUST is the heart of the funneling technique. To gain the computation reduction desired, this subroutine manipulates the windows around the 'current' state variable values, X_i' , in each stage for each program iteration. Initially the upper and lower limits HX_i' and LX_i' on the window at stage i are defined by,

$$HX_i' = X_i' + K * C_{i+1}, \quad (3.1)$$

and

$$LX_i' = X_i' - K * C_{i+1}, \quad (3.2)$$

where K is a constant. These bounds are illustrated in fig 3.4. Further adjustment of these limits may be necessary to

assure that one of the X_i values considered in the computation of $F(X_i)$ is X_i' . At stage i in the computation of $F(X_i)$ the optimization considers first $X_i = LX_i$, then X_i is incremented by C_i to give the next value, etc. Thus if LX_i is not properly set, the value $X_i = X_i'$ will never be considered.

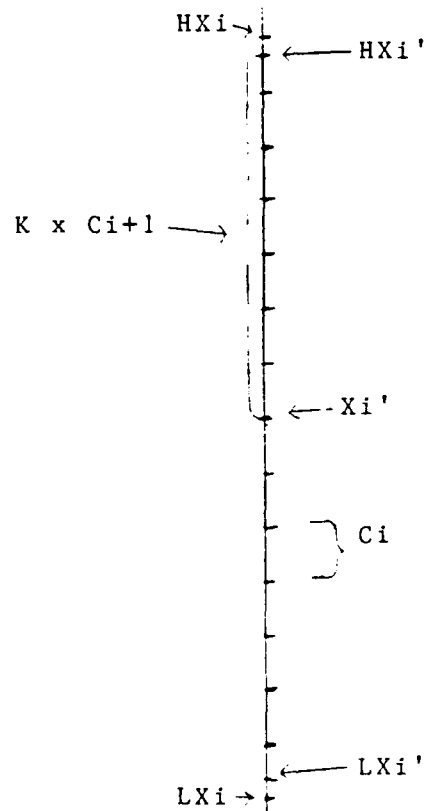


Figure 3.4 Example of a Window on a Stage i .

This adjustment of LX_i' is made by decrementing from the temporary stage value, X_i' , by the cost C_i for that stage. When this decrementing process first yields a value less than or equal to LX_i' that value is set as the new lower limit on the window and is called LX_i . A similar adjustment is made to the upper limit by incrementing from X_i' until the value HX_i' is equalled or exceeded. The new upper limit is called HX_i . This process is repeated for all the stages until the funnel is formed. In figure 3.5, the funnel may be seen for N stages of a provisioning problem. The budget quantity B is represented by the vertical lines for all stages. The upper and lower bounds are represented by HX_i , LX_i for $i=1, \dots, N$ where i is the stage index. In each stage the bounds are at least a distance equal to the product of the constant K and C_{i+1} from the X_i' value. Stages where the bounds are further from the center indicate the additional adjustment for compatibility between the successive X_i' values.

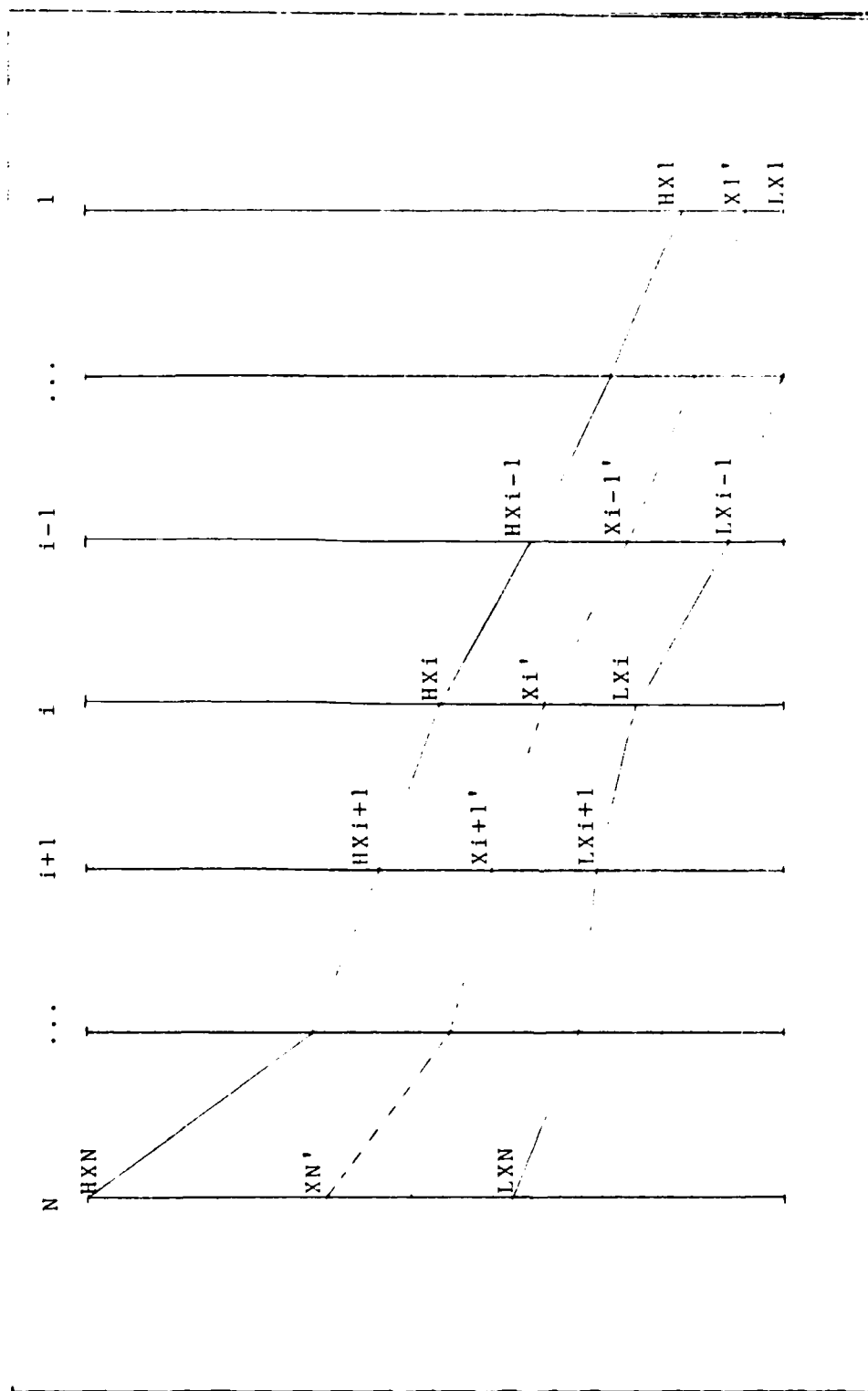


Figure 3.5 A Funnel Diagram for N Stages of a Problem.

The reason for multiplying the succeeding stage costs by the constant K is to allow the DP procedure some computational freedom in each of the stages while searching for the X_i and $S_i^*(X_i)$ values. During the computational process, should a stage choose an X_i' that is within C_{i+1} of the computational bound, the boundary is said to be violated. This is because the DP procedure may actually prefer a value outside the bound but is unable to reach it. Figures 3.6 through figure 3.9 show an example of the stage boundary manipulation as the iterative process of the DP6 program searches for the optimal solution.

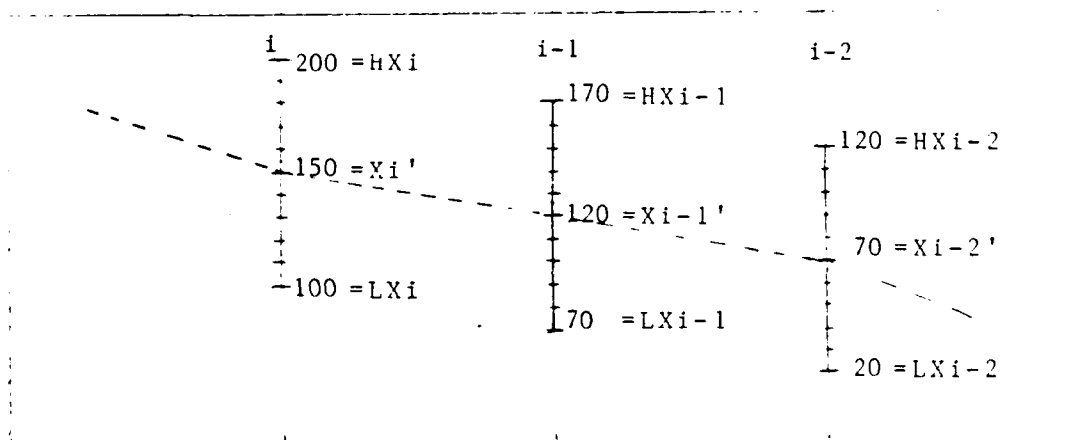


Figure 3.6 The Initialization Run.

The initializing run depends on the starting condition for the problem. If the ENTER function is used, the state variable increments will represent the costs of the respective stages. Figure 3.6 shows a starting condition of no prior knowledge; hence, the state variable increment is a constant, in this example 10, for all stages and of equal number on either side of the current X_i' estimate. A second run of the program provides an opportunity for the DP process to evaluate the estimated optimal returns with a

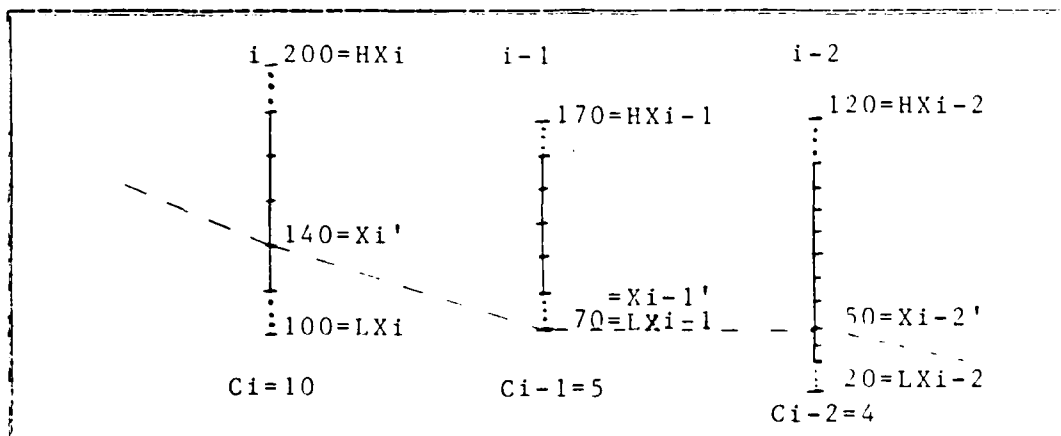


Figure 3.7 The Second Run.

smaller state variable increment or a revised boundary value and to adjust the window around a new X_i' accordingly. Figure 3.7 shows the funnel after the second run but before the windows are adjusted. The program has selected $X_i'=140$, $X_{i-1}'=70$, and $X_{i-2}'=50$. These are the current estimates of the optimal X_i' values at each stage. Notice that stage $i-1$ has a boundary that is violated, i.e., the X_{i-1}' for the stage is within $C_i=10$ of the boundary. In this case the optimization function may prefer an X_{i-1} outside the window but the boundary prevents its selection. Also in the second run the state variable increments have been changed to the cost of the item associated with the stage. In most stages these values are smaller than the initial increment making the return functions more sensitive. This increase in sensitivity should improve the optimization selection.

The violation of the lower boundary in stage $i-1$ causes the iteration to fail the conditions of the DONE checkpoint of figure 3.2. Consequently, the ADJUST subroutine is activated again to manipulate the boundaries producing the results shown in figure 3.8.

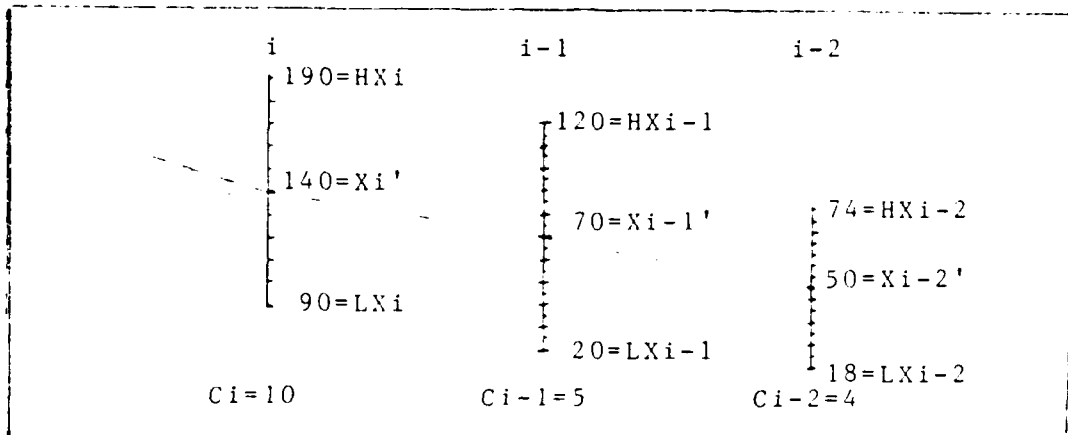


Figure 3.8 The Window Adjustments.

The points in this figure represent the input file for the third run of the program. In stage $i-1$ the window boundaries are centered on the previous iteration LX_{i-1} value. Notice the window bounds for stage $i-2$ have automatically adjusted downward to compensate for the increased resource consumption and avoid cascading boundary violations.

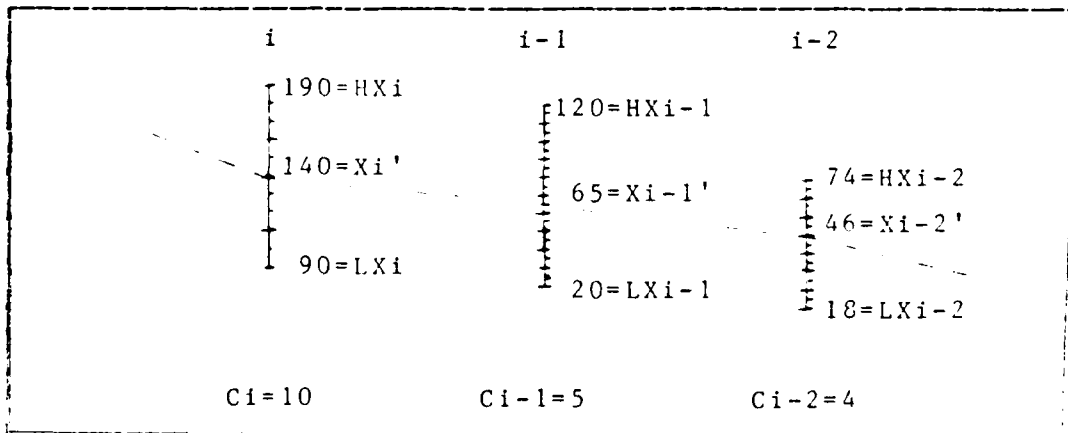


Figure 3.9 The Third Run.

Figure 3.9 shows the funnel after the third run. The program has selected $X_i^*=140$, $X_{i-1}^*=65$ and $X_{i-2}^*=46$ for the current estimates of the X_i^* values at each stage. Notice that the X_{i-1}^* value is just outside the previous LX_{i-1} , which was 70. In this example the Funnel program manipulated the boundaries of the windows to allow the DP process to choose different X_i^* values for the three stages, i , $i-1$, $i-2$. Subsequent runs of the program will determine the X_i^* for each stage. The process is 'DONE' when the remaining budget $\leq$ minimum C_i , an optimal integer solution is provided, and there are no boundaries violated.

C. IMPLEMENTATION PROBLEMS

From the characteristics of the funneling process, there are problems that arise when transitioning between stages. When computing the maximum and minimum value of the decision to be made for each X_i in a stage i , the upper and lower bounds and item cost of stage $i-1$ must be considered. In figure 3.10, the left side is the window for stage i , the right side is the window for stage $i-1$. For the stages i , $i-1$, the highest values of the state variables are represented by HX_i and HX_{i-1} while the lowest values are represented by LX_i , LX_{i-1} . In the program DP6, the two variables DLOW and DHIGH represent the minimum and maximum decisions available for the state variable in each stage. It is essential that DHIGH for any X_i value in the window of the i th stage be of such a magnitude that it will not force the X_{i-1} value below the lower bound LX_{i-1} . Additionally, DLOW must be of such magnitude that for any X_i in the window of the i th stage, it will not force the X_{i-1} value to violate the upper bound HX_{i-1} . If either of these decision quantities is incorrect at stage i the resource quantity will be outside the window at stage $i-1$. The derivation of the

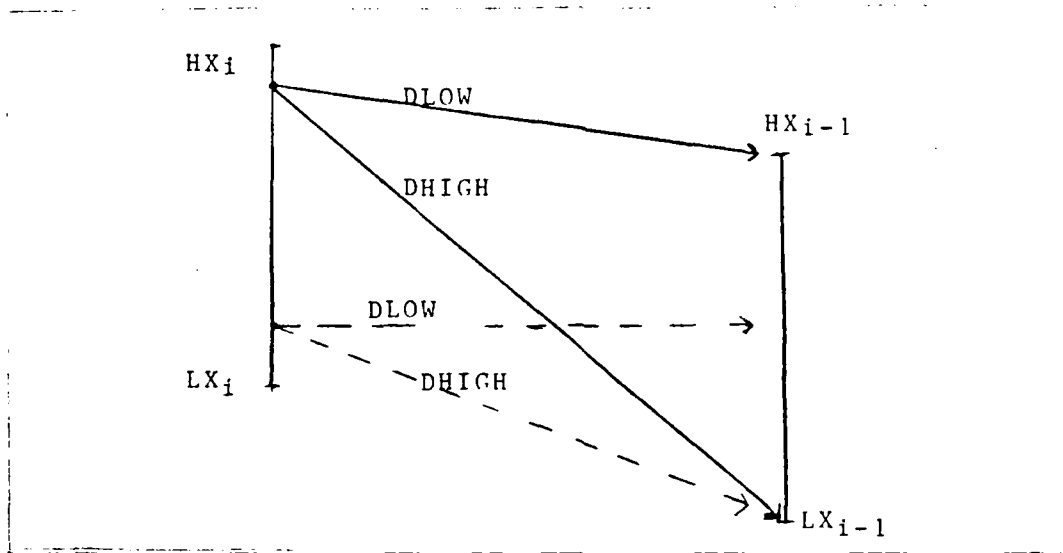


Figure 3.10 Boundary Manipulation.

formulas for these values follows. The transformation function as represented by equation 2.8 of Chap 2 is,

$$X_{i-1} = X_i - C_i S_i \quad i=1, \dots, N. \quad (3.3)$$

If this transformation is to take place from a window in stage i to a window in stage $i-1$ the following conditions must be true.

$$L_{X_{i-1}} \leq X_i - C_i * S_{HIGH}, \quad (3.4)$$

where S_{HIGH} is the maximum number of items to be purchased and

$$X_i - C_i * S_{LOW} \leq H_{X_{i-1}}, \quad (3.5)$$

where S_{LOW} is the minimum number of parts to be purchased. By algebraic manipulation a relationship for S_{LOW} and S_{HIGH} may be established,

$$SHIGH \leq (X_i - LX_{i-1}) / C_i, \quad (3.6)$$

and

$$SLOW \geq (X_i - HX_{i-1}) / C_i. \quad (3.7)$$

The expressions SHIGH AND SLOW are equivalent to the DHIGH, DLOW required in DP6 and are coded in the subroutine DLIMIT to control the boundary calculations. It is conceivable that at the extreme state variable values of X_i the computed boundaries could become negative or larger than the assigned budget. Thus, the decisions are further limited by $DHIGH = \min(SHIGH, 200)$ and $DLOW = \max(0, SLOW)$, where 200 is an arbitrarily chosen value.

The initialization for the problem is either from input decisions or else a large state variable increment is used. After the subroutine ADJUST manipulates the windows the first time, the state variable increments are decreased or the boundary conditions are changed to allow examination of the new possible X_i values for the best return. The subroutine DLIMIT automatically adjusts the DHIGH and DLOW values to compensate for the changes in the state variable increment. With this closer scrutiny, the optimal return value may change. The ADJUST subroutine moves the boundaries up or down accordingly when an optimal value is within one increment of the bounds. Recall that if the optimal return value is within one increment of window bound, it may be that the DP process would prefer a value outside the window. Hence, the boundary is recorded as violated, and the window is moved for the next iteration. This process is repeated until all the optimal return values are centered within the windows for all stages.

To save computations and avoid computing X_i values that are not compatible between stages, the state variable

increment for each stage is the cost of the item to be purchased in the previous stage. A problem arises in that the original DP5 program may provide solutions that are not integer. This is possible because in the stage being considered, the best return may require an X_i' value from succeeding stages that has not been computed. DP5 handles this by interpolating linearly between the state variable values it does have to arrive at an estimate for the one required. In DP6 to avoid this approximation and obtain integer solutions, the subroutine ADJUST, (as discussed in section A), manipulates the lower and upper window values to ensure the previous stage optimal return values are available for the X_i being considered. This adjustment process is also repetitive and is done in conjunction with the boundary control.

Another problem arises from the effort to save computer computation time. If there is a wide range of costs, the use of the item cost as the state variable increment within a stage becomes troublesome. Initially the ADJUST subroutine creates the state variable windows for the individual stages using a constant multiplier that may be set by the user. To remedy the problem of the extreme range of the costs of items for the provisioning problem being considered, (\$10000.00 to .50), the multiplier was changed to a variable that depends directly on the item cost. This was done to decrease the program repetitions required to reach an optimal solution. Under the constant multiplier method, a change in the purchase quantity of a larger cost item, (i.e. \$1500), could require many program repetitions to re-allocate the resource available for the lower cost items, (i.e. \$10, 5, 3.75, etc). The smaller window range, due to the constant multiplier, hampers the re-allocation effort. The variable multiplier allows greater compensation for lower cost items to facilitate a more rapid change if required.

IV. ANALYSIS

The primary topic of this chapter will be the comparison of the Marginal Analysis and Funneling solution techniques for the previously stated problem. The discussion will cover the results of both programs and the solution difference. The standard dynamic program procedure will not be addressed because of the computational advantage of the DP variant and the similarity of the two techniques.

For simplicity and ease of comparison a small three stage problem will be defined and solved using both solution procedures. As in the original provisioning problem the objective is to minimize the MSRT with a constraint on cost. Three different budget quantities will be examined to demonstrate the differences in the solution techniques. The following table is a list of the input parameters which are cost C_i , demand rate L_i , essentiality E_i , and procurement lead time T_i . For this particular example T_i is one year.

TABLE II
Sample Problem Input Parameters

I	DEMAND	COST	T	ESSEN
1	5.000	5.00	1.00	3.0
2	1.000	10.00	1.00	2.0
3	10.000	15.00	1.00	1.0

For the first budget quantity of \$195.00 the Marginal Analysis program returns the following output.

TABLE III				
Marginal Analysis Program Return, Budget=\$195.00				

* METHOD: FIXED BUDGET = 195.00 *				

STEP	ITEM	SI	RATIO	BR
1	1	1	0.480808496	190.00
2	1	2	0.365659833	185.00
3	1	3	0.260618091	180.00
4	1	4	0.172421157	175.00
5	1	5	0.105280340	170.00
6	2	1	0.0735758543	160.00
7	3	1	0.0600002967	145.00
8	1	6	0.0591956675	140.00
9	3	2	0.0533369593	125.00
10	3	3	0.0466887541	110.00
11	3	4	0.0400909968	95.00
12	3	5	0.0336193480	80.00
13	1	7	0.0306577347	75.00
14	3	6	0.0273999237	60.00
15	3	7	0.0216008648	45.00
16	2	2	0.0207276568	35.00
17	3	8	0.0164023377	20.00
18	1	8	0.0146531016	15.00
19	3	9	0.0119544677	0.0

MINIMIZE MSBT EXCESS: 0.0				
I	SI	MSBT		
1	8	1.302		
2	2	10.396		
3	9	12.176		
OVERALL			6.003	
REALIZED PERFORMANCE = 6.0033				
BOUNDS (6.0033, 6.0033)				

The decision results are, S1=8, S2=2, S3=9 and the return is Z= 6.0033, with no budget remaining. Viewing the purchase sequence in table III it may be seen that the Marginal

Analysis program has reached a solution uninterrupted, that is without implementing the heuristic developed in [Ref. 1]. Consequently, for this budget quantity the solution is optimal. Notice another indicator of the optimal solution in the last line of table III. The bounds described are the possible upper and lower values for the return. When these quantities are the same value, the solution is the best available.

TABLE IV
Funnel Program Results, Budget=\$195.00

THE OBJECTIVE IS MSBT
CONSTRAINT ON COST

THE DESCRIPTION WHICH APPEARS BELOW FOR STAGE 1
APPLIES TO STAGE 1 THRU STAGE 3 INCLUSIVE
THE PROBLEM IS TO MINIMIZE A 3 STAGE PROCESS
IS TO BE CHOSEN OPTIMALLY BETWEEN $XN=1.9500D+02$
AND $XN=1.9500D+02$
OPTIMAL $XN=1.95000D+02$ OPTIMAL RETURN= $6.00332D+00$

N	XN	S_N	$XN-1$
3	$1.9500D+02$	$9.00D+00$	$6.0000D+01$
2	$6.0000D+01$	$2.00D+00$	$4.0000D+01$
1	$4.0000D+01$	$8.00D+00$	0.0

The same budget and parameters entered in the Funnel program produce the results in table IV. The output is in the same format discussed in Chapter 3. The initial budget quantity, XN and the optimal return are shown in the line just above the individual stage results. The variable X_i is the budget quantity at the beginning of the stage, S_i is the decision or quantity purchased, and X_{i-1} is the budget at the end of the stage. Recall that item 3 is the most expensive and item 1 the least expensive. The optimal solution for the \$195.00 budget is $S_1^*=8$, $S_2^*=2$, and $S_3^*=9$ with a Z value equal to 6.0033 and no budget remaining. This answer

matches exactly the Marginal Analysis program results shown in table III, confirming that the Marginal Analysis solution is optimal.

TABLE V

Marginal Analysis Program Return, Budget=\$200.00

 * METHOD: FIXED BUDGET = 200.00 *

STEP	ITEM	SI	RATIO	BR
1	1	1	0.480808496	195.00
2	1	2	0.365659833	190.00
3	1	3	0.260618091	185.00
4	1	4	0.172421157	180.00
5	1	5	0.105280340	175.00
6	2	1	0.0735758543	165.00
7	3	1	0.0600002967	150.00
8	1	6	0.0591956675	145.00
9	3	2	0.0533369593	130.00
10	3	3	0.0466887541	115.00
11	3	4	0.0400909968	100.00
12	3	5	0.0336193480	85.00
13	3	7	0.0306577347	80.00
14	3	6	0.0273999237	65.00
15	3	7	0.0216008648	50.00
16	2	2	0.0207276568	40.00
17	3	8	0.0164023377	25.00
18	1	8	0.0146531016	20.00
19	3	9	0.0119544677	5.00
20	3	10	0.0083406679	-10.00
20	1	9	0.0064818673	0.0

 MINIMIZE MSRT EXCESS: 0.0

I	SI	MSRT
1	9	0.514
2	2	10.396
3	9	12.176

OVERALL 5.565

REALIZED PERFORMANCE = 5.565
 BOUNDS (6.0033, 4.3120)

For the second example the budget quantity of \$200.00 was used. The Marginal Analysis program returns the output

in table V. The results are $S_3=9$, $S_2=2$, and $S_1=9$ with a Z value of 5.5652 and no budget left over. Examining the detailed output of the Marginal Analysis program, table V, it may be seen that the purchase of the twentieth item drives the budget negative. The program implements the heuristic, re-evaluating the benefit to cost ratios, then chooses the number one item for the next purchase. Notice that in the bottom line of table V the upper and lower bounds are not equal. This is a good indication the solution may not be optimal and should be verified.

TABLE VI
Funnel Program Results, Budget=\$200.00

THE OBJECTIVE IS MSRT
CONSTRAINT ON COST

THE DESCRIPTION WHICH APPEARS BELOW FOR STAGE 1
APPLIES TO STAGE 1 THRU STAGE 3 INCLUSIVE
THE PROBLEM IS TO MINIMIZE A 3 STAGE PROCESS
IS TO BE CHOSEN OPTIMALLY BETWEEN $X_N=2.0000D+02$
AND $X_N=2.0000D+02$
OPTIMAL $X_N=2.0000D+02$ OPTIMAL RETURN=5.56519E+00

N	X_N	SN	X_{N-1}
3	2.0000D+02	9.00D+00	6.5000D+01
2	6.5000D+01	2.00D+00	4.5000D+01
1	4.5000D+01	9.00D+00	0.0

The same budget and parameters entered in the Funnel program produce the results given in table VI. In this case the answer is again identical, $S_1=9$, $S_2=2$, and $S_3=9$ with the optimal return $Z=5.56519$. Incidentally, the heuristic in the Marginal Analysis program makes the correct choice at the twentieth step when it selects the item with the second highest benefit to cost ratio.

For a third case, a budget of \$205.00 is entered in both programs. The results for the Marginal Analysis code are shown in table VII.

TABLE VII

Marginal Analysis Program Results, Budget=\$205.00

 * METHOD: FIXED BUDGET = 205.00 *

STEP	ITEM	SI	RATIO	BB
1	1	1	0.480808496	200.00
2	1	2	0.365659833	195.00
3	1	3	0.260618091	190.00
4	1	4	0.172421157	185.00
5	1	5	0.105280340	180.00
6	2	1	0.0735758543	170.00
7	3	1	0.0600002967	155.00
8	1	6	0.0591956675	150.00
9	3	2	0.0533369593	135.00
10	3	3	0.0466887541	120.00
11	3	4	0.0400909968	105.00
12	3	5	0.0336193480	90.00
13	1	7	0.0306577347	85.00
14	3	6	0.0273999237	70.00
15	3	7	0.0216008648	55.00
16	2	2	0.0207276568	45.00
17	3	8	0.0164023377	30.00
18	1	8	0.0146531016	25.00
19	3	9	0.0119544677	10.00
20	3	10	0.0083406679	-5.00
20	1	9	0.0064818673	5.00
21	1	10	0.0026624922	0.0

 MINIMIZE MSRT EXCESS: 0.0

I	SI	MSRT
1	10	0.190
2	2	10.396
3	9	12.176

OVERALL 5.385

REALIZED PERFORMANCE = 5.3852
 BOUNDS (6.0033, 4.3120)

The solution is S1=10, S2=2, and S3=9 with a return value of Z= 5.3852. The upper and lower bounds are not equal indicating the heuristic has been applied and the solution should be verified. Examining the output from the Marginal Analysis program, table VII, it may be seen the budget was again driven negative at the twentieth item. The heuristic

was activated purchasing two of the number one items to drive the budget to zero.

Entering the same input and budget parameters in the Funnel program produces the results shown in table VIII.

TABLE VIII			
Funnel Program Results, Budget=\$205.00			
THE OBJECTIVE IS MSRT CONSTRAINT ON COST			
THE DESCRIPTION WHICH APPEARS BELOW FOR STAGE 1 APPLIES TO STAGE 1 THRU STAGE 3 INCLUSIVE THE PROBLEM IS TO MINIMIZE A 3 STAGE PROCESSIN IS TO BE CHOSEN OPTIMALLY BETWEEN $XN=2.05000D+02$ AND $XN=2.0500D+02$ OPTIMAL $XN=2.05000D+02$ OPTIMAL RETURN= $5.30246D+00$			
N	XN	SN	XN-1
3	$2.0500D+02$	$1.00D+01$	$5.5000D+01$
2	$5.5000D+01$	$2.00D+00$	$3.5000D+01$
1	$3.5000D+01$	$7.00D+00$	0.0

The answers are not the same. The Funnel program results indicate the best part selection is $S1=7$, $S2=2$, and $S3=10$ for an optimal return of $Z=5.30246$. This selection provides an MSRT that is roughly .08 better than the marginal analysis method.

Recall from section A of Chapter 2 that the Marginal Analysis program has a function called KIT that will determine the return from a given decision set. Entering the optimal solution from the Funnel program into the KIT function produces the result in table IX. The resulting objective value is identical to the value computed by the DP variant. This shows that the computational procedures are the same for both programs.

Now consider the definition of a much larger problem. It is of the same type as the previous examples, an

TABLE IX		
Kit Function Results, Budget=\$205.00		
EVALUATION OF SPECIFIED ALLCCATION....		
I	SI	MSRT
1	7	3.085
2	2	10.396
3	10	7.610
OVERALL		5.302

objective of minimizing MSRT with a constraint on cost. The input parameters cost C_i , demand rate L_i , essentiality E_i , and procurement lead time T_i , are shown in in table X. The budget is \$74825.00 with twenty five stages and costs ranging from \$10000.00 to .50. Due to the length of the detailed results an abbreviated output of the Marginal Analysis solution is shown in table XI. The return is 2.853 with this decision set. Note in the last line of the table, the split in the bounds indicates the heuristic was implemented and the solution may not be optimal. Examining the abbreviated output of the Marginal Analysis program, Table 3.4 shows that the budget was driven negative ten iterations from the end. Entering the same parameters and budget quantity for both starting conditions of the Funnel program produces identical results shown in table XII. Comparing the results in tables XI and XII it may be seen that the return and decision set for the problem are the same for both programs. Interestingly, the Marginal Analysis program was able to correctly select ten items after the heuristic implementation to match the optimal solution produced by the Funnel program. The heuristic employed by Richards and McMasters appears to have been a good choice, although the solution must still be verified.

TABLE X
Input Parameters for a Larger Problem

I	DEMAND	COST	T	ESSEN
1	2.000	0.50	1.90	1.0
2	17.000	3.00	2.20	3.0
3	0.250	3.75	2.50	3.0
4	50.000	5.00	2.60	1.0
5	1.000	10.00	1.60	3.0
6	1.500	12.00	2.50	1.0
7	18.000	15.00	1.50	3.0
8	3.000	20.00	2.50	1.0
9	0.250	25.00	1.50	1.0
10	1.000	25.00	2.00	1.0
11	2.000	25.00	1.50	2.0
12	0.330	50.00	1.50	3.0
13	0.100	50.00	2.00	3.0
14	1.000	61.00	2.00	2.0
15	7.000	80.00	2.30	2.0
16	1.000	91.00	3.70	3.0
17	2.500	150.00	1.75	3.0
18	20.000	210.00	1.70	3.0
19	3.000	450.00	2.70	2.0
20	0.500	500.00	1.75	1.0
21	1.500	710.00	3.00	2.0
22	6.000	1000.00	1.80	1.0
23	2.000	1500.00	2.50	3.0
24	12.000	1500.00	2.00	2.0
25	0.500	10000.00	1.00	1.0

TABLE XI

Marginal Analysis Program Results, Budget=\$74825.00

 * METHOD: FIXED BUDGET = 74825.00 *

STEP	ITEM	SI	RAYIC	BR
1	1	1	2.82236671	74824.50
2	2	1	2.14117622	74821.50
3	2	2	2.08235264	74818.50
4	2	3	2.02352905	74815.50
5	2	4	1.96470547	74812.50
:	:	:	:	:
:	:	:	:	:

440	24	21	0.0004187953	2386.50
441	2	55	0.0004181664	2383.50
442	22	9	0.0003934940	1383.50
443	19	11	0.0003861645	933.50
444	18	40	0.0003540027	723.50
445	4	154	0.0003520278	718.50
446	24	22	0.0003425654	-781.50
446	12	3	0.0003396191	668.50
447	15	23	0.0003359483	588.50
448	7	39	0.0003278039	573.50
449	8	14	0.0003014780	553.50
450	20	2	0.0002939128	53.50
451	4	155	0.0002806289	48.50
452	2	56	0.0002605470	45.50
453	10	6	0.0002369717	20.50
454	1	12	0.0002271836	20.00
455	4	156	0.0002225503	15.00
456	7	40	0.0002018004	0.0

REALIZED PERFORMANCE = 2.8532
 BOUNDS (2.9796, 2.6762)

TABLE XII

Funnel Program Results, Budget=\$74825.00

THE OBJECTIVE IS MSBT
CONSTRAINT ON COST

THE PROBLEM IS TO MINIMIZE A 25 STAGE PROCESS
 XN IS TO BE CHOSEN OPTIMALLY BETWEEN $XN=7.48250D+04$
 AND $XN=7.48250D+04$
 OPTIMAL $XN=7.48250D+04$ OPTIMAL RETURN= $2.85315D+00$

N	XN	SN	XN-1
25	7.4825D+04	0.0	7.4825D+04
24	7.4825D+04	2.10D+01	4.3325D+04
23	4.3325D+04	6.00D+00	3.4325D+04
22	3.4325D+04	9.00D+00	2.5325D+04
21	2.5325D+04	6.00D+00	2.1065D+04
20	2.1065D+04	2.00D+00	2.0065D+04
19	2.0065D+04	1.10D+01	1.5115D+04
18	1.5115D+04	4.00D+01	6.7150D+03
17	6.7150D+03	8.00D+00	5.5150D+03
16	5.5150D+03	8.00D+00	4.7870D+03
15	4.7870D+03	2.30D+01	2.9470D+03
14	2.9470D+03	5.00D+00	2.6420D+03
13	2.6420D+03	2.00D+00	2.5420D+03
12	2.5420D+03	3.00D+00	2.3920D+03
11	2.3920D+03	7.00D+00	2.2170D+03
10	2.2170D+03	6.00D+00	2.0670D+03
9	2.0670D+03	2.00D+00	2.0170D+03
8	2.0170D+03	1.40D+01	1.7370D+03
7	1.7370D+03	4.00D+01	1.1370D+03
6	1.1370D+03	9.00D+00	1.0290D+03
5	1.0290D+03	6.00D+00	9.6900D+02
4	9.6900D+02	1.56D+02	1.8900D+02
3	1.8900D+02	4.00D+00	1.7400D+02
2	1.7400D+02	5.60D+01	6.0000D+00
1	6.0000D+00	1.20D+01	0.0

V. CONCLUSIONS

This chapter will discuss the advantages and disadvantages of the two primary solution techniques examined in this paper. Various starting modes of the Funnel program will be looked at in an effort to develop an efficient approach to this type of provisioning problem.

The Marginal analysis program provides a quick, simple and effective approach for generating a solution to the previously defined provisioning problem. However, the solution is not guaranteed to be optimal because the program may implement the Richards heuristic to consume all available resources. The Funnel program provides the optimal solution to the same problem though it takes more computer time to operate. The DP variant has the capability to start the problem from two different conditions, 1) no prior knowledge and 2) prior knowledge or an estimate of the solution. For Case 2, utilizing the ENTER function of the Funnel program allows the rapid construction of the input data file used by DP6 to examine the X_i^* values and manipulate the window boundaries as necessary. If the entered solution set is not optimal, the Funnel program will drive the decisions to the optimal decision set in subsequent iterations. Using the timing function mentioned in Chap. 3 a rough approximation of run time is available for each iteration of the program. For Case 1, no prior knowledge, the total estimated time for the \$74825.00 problem is nearly 3000 seconds. For Case 2, prior knowledge, a near optimal starting solution for the same problem is driven to the optimal solution in approximately 300 seconds. This 90% savings on computation time makes it clear that an approximation method as an input to the Funnel program would be a better approach to this type

of problem. It should be noted that the run time for the case 2 start is related to the accuracy of the entered decision set. In this instance the entered decision set was specifically constructed to force the Funnel program to adjust the window boundaries in all stages at least once. A better solution than the one entered could produce a much shorter run time.

There are tradeoffs in the Funnel Program for computational efficiency. Variations of the multiplier that creates the window can drive the individual program run time directly. A window that is too large wastes computation time on resource values that are not needed. If the window width is too small, multiple iterations of the program may be required to compensate for the increased number of boundary violations in the effort to reallocate the resource consumed at the various stages. For the verification of a decision set provided by the Marginal Analysis program the multiplier could be set to 1.0 providing a very narrow window of resource to be examined. Due to the narrow width of the window, if the X_i^* value changes in any stage the boundary will be recorded and displayed to the user as violated. This notification would indicate that the entered solution is not optimal for the problem.

The computer time expended for the verification alternative would be quite low even for large budget problems. In the 25-stage problem, for example, approximately one second is required to verify the correctness of the optimal solution if it is entered initially. If the verification alternative indicates the decision is not optimal, the time required to drive the entered solution to the optimal solution is controlled by the magnitude of the changes in the decision set and the number of iterations required to re-allocate the resources throughout all the stages. The user must make a value judgement on the importance of the optimal solution versus the additional computer costs.

In conclusion, the emphasis of this paper is to create an algorithm to determine the optimal solution in a quick and easy manner. It is evident that a combination of the Marginal Analysis and Funnel programs is an efficient approach for solving this type of problem. Using the Marginal Analysis program to generate a best estimate of the solution, the ENTER function with the Funnel program can then be applied to confirm the optimal solution or to create an input data file that DP6 may use to find the optimal solution. The provisioning problems faced by NAVSUP, as described in Chap 1, may be solved with this method if they can be formatted in the recursive DP construct. The program DP6 in particular solves only the minimization of MSRT with a constraint on cost. Variations on this program could be devised to solve additional objectives like supply material availability(SMA), time weighted units short(TWUS), etc., by changing the computation procedures in the subroutines. The constraint however, is currently restricted to the linear form,

$$\sum W_i * S_i \leq B \quad i=1, \dots, N \quad (5.1)$$

where W_i is resources required per unit, S_i is the number of units of item i , and B is the total resource available.

APPENDIX A

DP6 CODE

This appendix contains the computer listing of the main program and subroutines for the funneling procedure. Some of the subroutines contain code for objectives and constraints other than those discussed in this thesis.

DATA DESCRIPTION FOR CARD NUMBER 1				
COLUMNS	JUSTIFY	VARIABLE NAME	MEANING	
1 - 5	COL 5	NMAX	THE NUMBER OF STAGES IN THE PROBLEM	DP600031)
6 - 10	COL 10	MTAPE	LOGICAL TAPE NO OF THE MAIN TAPE IF THIS IS LEFT BLANK, THE COMPUTER WILL USE STANDARD SCRATCH TAPE 3	DP600020 DP600030 DP600040 DP600050 DP600060 DP600070 DP600080 DP600090 DP600100 DP600110 DP600120 DP600130 DP600140 DP600150 DP600160 DP600170 DP600180 DP600190 DP600200 DP600210 DP600220 DP600230
11 - 15	COL 15	SOLVE	= OLD IF THE OPTIMAL DECISION FUNCTIONS HAVE ALREADY BEEN CALCULATED AND ARE LOADED ON LOGICAL TAPE MTAPE	DP600240 DP600250 DP600260 DP600270 DP600280 DP600290 DP600300 DP600310 DP600320 DP600330 DP600340 DP600350 DP600360 DP600370 DP600380 DP600390 DP600400 DP600410 DP600420 DP600430 DP600440 DP600450 DP600460 DP600470 DP600480
			= NEW IF THE OPTIMAL DECISION FUNCTIONS MUST BE CALCULATED AND STORED ON LOGICAL TAPE MTAPE BEFORE SOLVING THE PROBLEM	

DATA DESCRIPTION FOR THE NEXT GROUP OF CARDS				
COLUMNS	JUSTIFY	VARIABLE NAME	MEANING	
1 - 5	COL 5	NSTAGE	LOWEST NUMBERED STAGE FOR WHICH THIS CARD APPLIES	
6 - 10	COL 10	NDITTO	HIGHEST NUMBERED STAGE FOR WHICH THIS CARD APPLIES. IF THESE COLUMNS ARE LEFT BLANK, THEN NDITTO WILL BE TAKEN AS EQUAL TO NSTAGE	
11 - 20	ANY	XLOW	LOWEST VALUE OF XN FOR STAGE NSTAGE	
21 - 30	ANY	XHIGH	HIGHEST VALUE OF XN FOR STAGE NSTAGE	

31 - 40	ANY	DELX	INCREMENT IN XN FOR STAGE NSTAGE	DP600490
41 - 46	COL 46	XMODE	= MIN IF STAGE NSTAGE IS TO BE MINIMIZED	DP600500
			= MAX IF STAGE NSTAGE IS TO BE MAXIMIZED	DP600510
47 - 52	COL 52	XSTAGE	= SUM IF THE COMPOSITION OPERATOR BETWEEN STAGES NSTAGE AND NSTAGE-1 IS ADDITION	DP600520
			= MULT IF THE COMPOSITION OPERATOR BETWEEN STAGES NSTAGE AND NSTAGE-1 IS MULTIPLICATION	DP600530
			= MINMAX IF THE COMPOSITION OPERATOR BETWEEN STAGES NSTAGE AND NSTAGE-1 IS TO MINIMIZE THE MAXIMUM INDIVIDUAL STAGE RETURN	DP600540
			= MAXMIN IF THE COMPOSITION OPERATOR BETWEEN STAGES NSTAGE AND NSTAGE-1 IS TO MAXIMIZE THE MINIMUM INDIVIDUAL STAGE RETURN	DP600550
			= THIS VARIABLE IS NOT USED WHEN NSTAGE=1, THEREFORE COLUMNS 47 TO 52 ARE IGNORED WHEN NSTAGE=1 AND MAY BE LEFT BLANK HOWEVER IF NDLTTO IS GREATER THAN 1, THEN XSTAGED MUST BE SPECIFIED IN ACCORDANCE WITH THE COMPOSITION OPERATOR FOR THE REST OF STAGES DESCRIBED THIS CARD	DP600560
53 - 55	COL 55	MODES	= 1 IF THE FIBONACCI SEARCH IS TO BE USED	DP600570
			= 2 IF THE FIBONACCI SEARCH IS	DP600580

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

NOT TO BE USED

56 - 64	ANY	XMIN1L	THIS VARIABLE IS THE LOWEST BOUNDARY VALUE ANTICIPATED FOR THIS STAGE	DP60097J DP600980 DP600990 DP601000 DP601010 DP601020 DP601030 DP601040 DP601050 DP601060 DP601070 DP601080 DP601090 DP601100 DP601110 DP601120 DP601130 DP601140
65 - 73	ANY	XMIN1H	THIS VARIABLE IS THE HIGHEST BOUNDARY VALUE ANTICIPATED FOR THIS STAGE	DP601150 DP601160 DP601170 DP601180 DP601190 DP601200 DP601210 DP601220 DP601230 DP601240 DP601250 DP601260 DP601270 DP601280 DP601290 DP601300 DP601310 DP601320 DP601330 DP601340 DP601350 DP601360 DP601370 DP601380 DP601390 DP601400 DP601410 DP601420 DP601430 DP601440

DATA DESCRIPTION OF LAST CARD FOR PROBLEM

THE OPTIMUM DECISIONS ARE TO BE PRINTED OUT FOR A PROCESS WITH NMAX STAGES, FURTHER, FN(XN) IS TO BE OPTIMIZED WITH RESPECT TO XN WHEN THE PROGRAM COMPLETES THE REQUESTED PRINTOUT IT WILL RETURN TO THE BEGINNING AND START A NEW PROBLEM IF THERE IS DATA, IF NO DATA, IT WILL EXIT

COLUMNS	JUSTIFY	VARIABLE NAME	MEANING
1 - 15	ANY	XN1	AT STAGE NMAX, OPTIMIZE FN(XN) FOR XN NOT LESS THAN XN1
16 - 30	ANY	XN2	AT STAGE NMAX, OPTIMIZE FN(XN) FOR XN NOT GREATER THAN XN2

DESCRIPTION OF THE SUBROUTINE TRANFM

THE MAIN PROGRAM TRANSMITS XN, DN, AND NUMBER (THE SEQUENCE NUMBER OF THE PROBLEM) TO THE SUBROUTINE AND THE SUBROUTINE CALCULATES YN (THE OUTPUT OF THE STAGE I.E. YN=XN-1)

DESCRIPTION OF THE SUBROUTINE STGRET

THE MAIN PROGRAM TRANSMITS XN, DN, YN, AND NUMBER TO THE SUBROUTINE, AND THE SUBROUTINE CALCULATES RN (THE STAGE RETURN)

DESCRIPTION OF SUBROUTINE STORE

THE MAIN PROGRAM CALLS THE SUBROUTINE RIGHT AFTER READING THE FIRST DATA CARD AND DEFINING NUMBER. THE PURPOSE OF THIS SUBROUTINE IS TO ALLOW THE STORING OF CONSTANTS IN COMMON STORAGE FOR THE POSSIBLE USE OF SUBROUTINES STGRET AND TRANFM

DATA WHICH WILL BE READ FROM DATA CARDS AT OBJECT TIME BY
 THE SUBROUTINE STORE MUST BE INSERTED BETWEEN THE FIRST DATA CARD
 AND THE GROUP OF CARDS WHICH DESCRIBE THE INDIVIDUAL STAGES

DESCRIPTION OF SUBROUTINE DLIMIT
 THE MAIN PROGRAM TRANSMITS N, XN AND NUMBER TO THE SUBROUTINE,
 AND THE SUBROUTINE CALCULATES DLOW, DHIGH, AND DELD
 WHERE DLOW = LOWEST VALUE OF DN FOR THAT PARTICULAR VALUE OF N AND DDEL
 XN
 DHIGH = HIGHEST VALUE OF DN FOR THAT PARTICULAR VALUE OF N
 AND XN
 DELD = INCREMENT IN DN FOR THAT PARTICULAR VALUE OF N AND XN
 IF ANY OF THESE VARIABLES DO NOT CHANGE, THEY CAN BE SET IN
 SUBROUTINE STORE AND NOT MENTIONED IN SUBROUTINE DLIMIT

IF ANY SUBROUTINE SETS NUFF TO -1 THEN THE PROGRAM IGNORES THE
 VALUE OF DN BEING PROCESSED AT THAT MOMENT, AND IGNORES ALL LARGER
 VALUES OF DN FOR THAT PARTICULAR VALUE OF XN AT THAT PARTICULAR
 STAGE--THIS FEATURE IS DISABLED WHEN USING THE FIBONACCI SEARCH

DESCRIPTION OF THE SUBROUTINE ENTER
 THIS SUBROUTINE READS A DECISION SET FROM A FILE DESIGNATED BY THE
 AND FORMS AN INPUT DATA FILE THAT IS COMPATIBLE WITH THE ADJUST
 TIME FOR BOUNDARY MANIPULATION. IT MAY ALSO BE USED FOR VERIFICATION
 AN OPTIMAL SOLUTION BY SETTING EQUAL TO ONE THE CONSTANT MULTIPLIED
 CONTROLS THE WINDOW WIDTH.

DESCRIPTION OF THE SUBROUTINE ADJUST
 THIS SUBROUTINE MANIPULATES THE UPPER AND LOWER BOUNDARIES OF THE
 COMPUTATIONAL WINDOWS TO ALLOW THE DP PROCESS TO LOCATE THE
 OPTIMAL XI AND SI VALUES.

THIS PROGRAM USES TAPES 3 AND 4 FOR SCRATCH
 NSS2 ZERO FOR ENTIRE PROBLEM SOLVING AND OUTPUT
 NSS2 NON-ZERO FOR OUTPUT FROM PREVIOUS PROBLEM
 DIMENSION FN(10001), FNM1(10001), DNOFXN(10001), A(101), E(101), C(101)
 DIMENSION D(101), E(101), F(101), G(101), BOUND(101), V(101)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

REAL * 8 HOLMIN/' MIN' / HOLOLD/' OLD' /
REAL * 8 XMODE, XSTAGE, SOLVE
REAL * 8 TAB(4)
REAL * 8 AB, E, SUMELT, RN, TERM, TEMP, AB, CDF
COMMON /CON/ TAB
COMMON XN, DN, YN, RN, XLOW, XHIGH, DELX, DLOW, DHIGH, DELD, NUMBER
COMMON YLOW, YHIGH, DELY, NUFF, FN, JTOP, NNFLAG, NMAX, XLAM
COMMON A, B, C, D, E, F, G, V, SUMELT, KODE1, KODE2, KODE3
COMMON XMIN, XMAX, XMINH, XMAXH, XN1, XN2
EQUIVALENCE (FN(1), FNM1(1), DNOFXN(1))
CALL SETIME
XMIN1I=0.0
XMIN1H=0.0
XOUTPE=6
INTAPE=7
KEEP=9
NEXT1=10
IBALL=11
NOUTPE=12
WRITE(NOUTPE, 1000) HOLMIN, HOLOLD, (TAB(I), I=1,4)
1000 FORMAT(1X, 6A6)
CALL SEARCH(0.0,0.0,-1.0,0.0,DNBEST,BEST)
MAX=10001
NPAGE=0
NUMBER=0
240 READ(INTAPE, 100, END=2000) NMAX, MTAPE, SOLVE
6 100 WRITE(KEEP, 100) NMAX, MTAPE, SOLVE
FORMAT(2I5, A5)
MTAP = MTAPE
NDITTC=0
NCALC=4
IF (MTAPE-4) 405, 501, 405
501 NCALC=3
405 GO TO 172
322 IF (MTAPE) 322, 322, 172
172 MTAPE=3
REWIND MTAPE
NUMBER=NUMBER+1
CALL STORE WAS MOVED FROM HERE
C 302 IF (SOLVE-HOLOLD) 301, 302, 301
302 NSS2=1
NPAGE=NPAGE+1
C 152 WRITE(NOUTPE, 152) NPAGE
C 152 FORMAT(1H160X4HPAGEI4)
C 303 WRITE(NOUTPE, 303)
C 303 WRITE(NOUTPE, 1303) MTAPE
C 303 FORMAT (53H THE TABLES OF FN(XN) AND DN(XN) WHICH ARE ON LOGICAL)

```

DP601933
 DP601940
 DP601950
 DP601960
 DP601970
 DP601980
 DP601990
 DP602000
 DP602010
 DP602020
 DP602030
 DP602040
 DP602050
 DP602060
 DP602070
 DP602080
 DP602090
 DP602100
 DP602110
 DP602120
 DP602130
 DP602140
 DP602150
 DP602160
 DP602170
 DP602180
 DP602190
 DP602200
 DP602210
 DP602220
 DP602230
 DP602240
 DP602250
 DP602260
 DP602270
 DP602280
 DP602290
 DP602300
 DP602310
 DP602320
 DP602330
 DP602340
 DP602350
 DP602360
 DP602370
 DP602380
 DP602390
 DP602400

```

C1303  FORMAT(43H TAPEIS, WILL BE USED TO SOLVE THIS PROBLEM)
      READ(MTAPE) NMAX
      REWIND MTAPE
      GO TO 304
301  NSS2=0
      CALL STORE
      NPAGE=NPAGE+1
      WRITE(NOUTPE,152) NPAGE
      WRITE(NOUTPE,305)
      WRITE(NOUTPE,1305) MTAPE
      FORMAT(58H THE COMPUTER IS TO CALCULATE TABLES OF FN(XN) AND DN(XN)
      )
C1305  FORMAT(31H AND STORE THEM ON LOGICAL TAPEIS)
      REWIND NCALC
500  WRITE(MTAPE) NMAX
C304  WRITE(NOUTPE,306) NMAX
C304  CHANGED TO CONTINUE FOR ABBREVIATED PRINTOUT
C304  CONTINUE
C306  FORNAT(16H THE PROBLEM HASI5,7H STAGES)
314  IF(NSS2) 197,109,197
109  LINES = 6
C2006  WRITE(NOUTPE,2006) NMAX, MTAP, SOLVE
121  FORMAT(///,DATA CARD,217,A7)
      DO 123 N=1,NMAX
      NM1=N-1
      IF(NDITTO-N) 241,132,132
      READ(INTAPE,124) NSTAGE,NDITTO,XLOW,XHIGH,DELX,XMODE,XSTAGE,MODES
124  1XMIN1L,XMIN1H
      1XFORMAT(2I5,3E10.8,2A6,I3,2E9.2)
      WRITE(NOUTPE,2007) NSTAGE,NDITTO,XLOW,XHIGH,DELX,XMODE,
      XSTAGE,MODES
      WRITE(NOUTPE,2009) XMIN1L,XMIN1H
      FORMAT(1XN-1,1CW,HIGH ARE:2E9.2)
      1XFORMAT(///,DATA CARD,215,2X,3F10.3,2A6,I3)
      IF(LINES-4) 30,30,31
      NPAGE=NPAGE+1
      WRITE(NOUTPE,152) NPAGE
      LINES=1
      LINES = LINES + 9
      ITOP=(XHIGH-XLOW)/DELX+1.001
      IF(MODES-1) 600,601,600
      600  MODES=-1
      601  IF(NSTAGE-N) 125,245,125
      125  WRITE(NOUTPE,127)
      127  FORMAT(23H1DATA CARD OUT OF ORDER)
      GO TO 2003
      245  IF(NSTAGE-NDITTO) 246,126,126
      246  WRITE(NOUTPE,247) N

```

```

247 WRITE(NOUTPE,1247) NSTAGE,NDITTO
1247 FORMAT(//46H THE DESCRIPTION WHICH APPEARS BELOW FOR STAGE,I5)
      LINES=LINES+4
      XXMODE = XMODE - HOLMIN
      WRITE(6,1001) XXMODE
C1001 FORMAT(1X,A6)
126 IF(XMODE-HOLMIN) 230,231,230
231 XMODE=1.0
C 232 WRITE(NOUTPE,232) N
C 232 FORMAT(//6H STAGEI5,18H IS A MINIMIZATION)
      GO TO 520
230 XMODE=-1.0
C 230 WRITE(NOUTPE,234) N
C 234 FORMAT(//6H STAGEI5,18H IS A MAXIMIZATION)
520 DO 129 I=1,4
      IF(XSTAGE-TAB(I)) 129,130,129
129 CONTINUE
521 NNFLAG=5
130 NNFLAG=1
      CADD HERE TO COMPENSATE FOR ABBREVIATED OUTPUT
      GO TO 132
C 502 CHANGED TO CONTINUE FOR ABBREVIATED OUTPUT
C 502 CONTINUE
C 316 IF(ITOP-MAX) 316,316,112
C 316 WRITE(NOUTPE,110) XLOW,XHIGH,DELX
C 317 WRITE(NOUTPE,110) XLOW,XHIGH,DELX
C 110 FORMAT(54H THE STATE VARIABLE IS BEING TREATED AS DISCRETE, GOING)
C1102 IF(MODES) 318,318,321
C 318 WRITE(NOUTPE,510)
C 510 FORMAT(41H TOTAL ENUMERATION IS USED FOR THIS STAGE)
      GO TO 320
C 321 WRITE(NOUTPE,511)
C 511 FORMAT(40H FIBNACCI SEARCH IS USED FOR THIS STAGE)
C 320 GO TO 11,12,13,14,132,NNFLAG
C 11 WRITE(NOUTPE,16) NM1,N
C 16 FCERMAT(40H THE COMPOSITION OPERATOR BETWEEN STAGESI6,4H ANDI6)
C1016 FORMAT(12H IS ADDITION)
      GO TO 132
C 12 WRITE(NOUTPE,17) NM1,N
C 12 WRITE(NOUTPE,1017)
C 17 FORMAT(40H THE COMPOSITION OPERATOR BETWEEN STAGESI6,4H ANDI6)
C1017 FORMAT(18H IS MULTIPLICATION)
      GO TO 132
C 13 WRITE(NOUTPE,18) NM1,N

```

DP602890
 DP602900
 DP602910
 DP602920
 DP602930
 DP602940
 DP602950
 DP602960
 DP602970
 DP602980
 DP602990
 DP603000
 DP603010
 DP603020
 DP603030
 DP603040
 DP603050
 DP603060
 DP603070
 DP603080
 DP603090
 DP603100
 DP603110
 DP603120
 DP603130
 DP603140
 DP603150
 DP603160
 DP603170
 DP603180
 DP603190
 DP603200
 DP603210
 DP603220
 DP603230
 DP603240
 DP603250
 DP603260
 DP603270
 DP603280
 DP603290
 DP603300
 DP603310
 DP603320
 DP603330
 DP603340
 DP603350
 DP603360

```

C 18 WRITE(NOUTPE,1018)
C1018 FORMAT(40H THE COMPOSITION OPERATOR BETWEEN STAGESI6,4H ANDI6)
C GO TO 132
C 14 WRITE(NOUTPE,19),NM1,N
C WRITE(NOUTPE,1019)
C 19 FORMAT(40H THE COMPOSITION OPERATOR BETWEEN STAGESI6,4H ANDI6)
C1019 FORMAT(34H IS TO MINIMIZE THE MAXIMUM RETURN)
132 DO 136 I=1,ITOP
134 PE=1.0E+35*XMDE
X=I-1
XN=XLOW+X*DELY
NUFF=1
CALL DLIMIT
IF(MODES) 602,602,603
603 CALL SEARCH(DLOW,DHIGH,DELD,XMODE,DNBEST,BEST)
GO TO 400
602 KTOP=(DHIGH-DLOW)/DELD+1.001
C NEXT LINE INSERTED FOR TESTING 1 AUG 84
RNBST=999
223 DO 137 J=1,KTOP
X=J-1
DN=DLOW+X*DELD
CALL TRANSF
CALL STRET
IF(NUFF) 400,401,401
IF(N-1) 504,504,503
401 CN=RN
504 GO TO 143
503 K=(YN-YLOW)/DELD+1.001
IF(K) 212,212,209 N,XN,DN
212 WRITE(NOUTPE,210),YN
210 WRITE(NOUTPE,1210)
1210 FORMAT(//9H AT STAGEI5,9H WITH XN=E15.8,8H AND DN=E15.8)
C WRITE(NOUTPE,5)XMIN1L,XMIN1H
C5 FORMAT(5X,2F10.4)
GO TO 2003
209 IF(K-JTOP) 205,206,212
206 RN1=FNMI(JTOP)
205 X=K-1
X=YLOW+X*DELY
RN1=FNMI(K)+(FNMI(K+1)-FNMI(K))*(YN-X)/DELY
GO TO 212
248 WRITE(NOUTPE,249)N
249 FORMAT(43HICOMPOSITION OPERATOR NOT DEFINED FOR STAGE,I5)
GO TO 2003

```

```

DP603370
DP603380
DP603390
DP603400
DP603410
DP603420
DP603430
DP603440
DP603450
DP603460
DP603470
DP603480
DP603490
DP603500
DP603510
DP603520
DP603530
DP603540
DP603550
DP603560
DP603570
DP603580
DP603590
DP603600
DP603610
DP603620
DP603630
DP603640
DP603650
DP603660
DP603670
DP603680
DP603690
DP603700
DP603710
DP603720
DP603730
DP603740
DP603750
DP603760
DP603770
DP603780
DP603790
DP603800
DP603810
DP603820
DP603830
DP603840

```

DP60385
 DP60386
 DP60387
 DP60388
 DP60389
 DP60390
 DP60391
 DP60392
 DP60393
 DP60394
 DP60395
 DP60396
 DP60397
 DP60398
 DP60399
 DP60400
 DP60401
 DP60402
 DP60403
 DP60404
 DP60405
 DP60406
 DP60407
 DP60408
 DP60409
 DP60410
 DP60411
 DP60412
 DP60413
 DP60414
 DP60415
 DP60416
 DP60417
 DP60418
 DP60419
 DP60420
 DP60421
 DP60422
 DP60423
 DP60424
 DP60425
 DP60426
 DP60427
 DP60428
 DP60429
 DP60430
 DP60431
 DP60432

```

21  CN=RN+RNM1
22  GO TO 143
23  CN=RN+RNM1
24  GO TO 143
25  IF (RN-RNM1) 141,141,142
141  GO TO 143
142  CN=RN+RNM1
24  IF (RN-RNM1) 145,145,146
145  GO TO 143
146  CN=RN
147  IF ((QN-BEST)*XMODE) 144,137,137
148  BEST=CN
149  DNBEST=DN
150  NEXT LINE INSERTED FOR TESTING 1AUG84
151  RNBST=RN
152  WRITE (IBALL,5) DN,BEST
153  FORMAT(' AT',F10.2,' THE TOTAL RETURN IS',F15.9)
154  CC CONTINUE
155  WRITE (NCALC) BEST,DNBEST
156  BEST1=RNBST*365
157  BEST2=SUMELT*BEST1/(E(N)*A(N)*B(N))
158  WRITE (IBALL,2008) N,XN,DNBEST,RNBST,BEST1,BEST2
159  FORMAT(I5,2F10.2,3F15.10)
160  CC CONTINUE
161  REWIND NCALC
162  WRITE (NTAPE) XLOW,XHIGH,DELX,ITOP,XMODE,XMIN1L,XMIN1H
163  DO 404 I=1,ITOP
164  READ (NCALC) DUM,DNOFXN(I)
165  REWIND NCALC
166  WRITE (NTAPE) (DNOFXN(II),II=1,ITOP)
167  DO 402 I=1,ITOP
168  READ (NCALC) FN(I),DUM
169  REWIND NCALC
170  XLOW=XLOW
171  XHIGH=XHIGH
172  DELY=DELX
173  JTCP=ITOP
174  CONTINUE
175  WRITE (NTAPE) (FN(II),II=1,ITOP)
176  END FILE MTAPE
177  REWIND MTAPE
178  READ (MTAPE) NMAX
179  N=NMAX
180  J=N-1
181  IF (J) 407,407,201
  
```

```

01 DO 522 I=1,J
02 READ(MTAPE) XLOW,XHIGH,DELX,ITOP,XMODE
03 READ(MTAPE) (FN(I),I=1,ITOP)
04 READ(MTAPE) TAPE
05 BACKSPACE MTAPE
06 READ(MTAPE) XN1,XN2
07 FORMAT(2E15.8)
08 NPAGE=NPAGE+1
09 WRITE(NOUTPE,152) NPAGE
10 WRITE(NOUTPE,200) XN1,XN2
11 FORMAT(/,/ , DATA CARD ', 2F20.8 ///)
12 IF(XMODE) 323,323,324
13 WRITE(NOUTPE,325) N
14 FORMAT(29H THE PROBLEM IS TO MINIMIZE AI6,14H STAGE PROCESS)
15 GO TO 327
16 WRITE(NOUTPE,326) N
17 FORMAT(29H THE PROBLEM IS TO MAXIMIZE AI6,14H STAGE PROCESS)
18 WRITE(NOUTPE,328) XN1
19 WRITE(NOUTPE,1328) XN2
20 FORMAT(42H AN IS TO BE CHOSEN OPTIMALLY BETWEEN XN=1PE14.5)
21 FCFORMAT(8H AND XN=1PE14.5)
22 IF(XLOW-XN1) 175,175,176
23 WRITE(NOUTPE,177) XLOW,XHIGH
24 FORMAT(47H THE PROGRAM ONLY HAS INFORMATION ON XN BETWEEN E16.8)
25 GO TO 240
26 IF(XN2-XHIGH) 181,181,176
27 IF(XN1-XN2) 178,178,176
28 IX1=(XN1-XLOW)/DELX+1.001
29 IX2=(XN2-XLOW)/DELX+1.001
30 BEST=1.0E+35*XMODE
31 DO 182 J=IX1,IX2
32 IF(FN(J)-BEST)*XMODE) 183,182,182
33 BEST=FN(J)
34 JSAVE=J
35 CONTINUE
36 X=JSAVE-1
37 XN=XLOW+X*DELX
38 BEST=BEST*.365
39 WRITE(NOUTPE,184) XN,BEST
40 FORMAT(12H OPTIMAL XN=1PE14.5,16H OPTIMAL RETURN=1PE14.5)
41 READ(MTAPE) (DNCFXN(II),II=1,ITOP)
42 DNCFXN(JSAVE)
43 CALL TRANSF M

```

```

204 WRITE(NOUTPE,188)
188 FORMAT(7H0,N18X2HXN18X2HDN16X4HXN-1)
203 WRITE(NOUTPE,189) N,XN,DN,YN
1203 WRITE(KEEP,190) N,DN,XN,C(N),XMIN1L,XMIN1H
189 FORMAT(17,1P3E20.8)
190 FORMAT(115,5F10.2)
NN=NN+1
N=N-1
IF((N-1).GE.0.0)GO TO 1243
C1244 CALL ADJUST
1243 IF(N-1)244,193,193
C1244 MOVED 244 LABEL UP ONE LINE SHOULD BE ON GO TO 240
244 IF(KODE3.EQ.0) GO TO 1333
CALL ENTER2
1333 CONTINUE
CALL SADJUS
CALL GETIME (IFT)
ET=ET*.00026
WRITE(NOUTPE,2)ET
WRITE(NOUTPE,2)ET
FORMAT(' ELAPSED TIME FOR EXECUTION WAS ',316.6,' SECONDS.')
C2 CALL FRTCMS(COM(1),COM(3),COM(J5),COM(6),COM(1),COM(2),COM(15),
C1COM(6))
GO TO 240
193 DO 406 I=1,4
406 PACKSPACE MTAPE
READ(MTAPE) XLCK,XHIGH,DELX,ITOP,XMODE,XMIN1L,XMIN1H
READ(MTAPE) (DNOFXN(I),I=1,ITOP)
L=(YN-XLOW)/DELX+1.001
IF(L)214,214,215
L=1
214 NP1=N+1
219 WRITE(NOUTPE,210) NP1,XN,DN
WRITE(NOUTPE,1216) YN
NN=NN+7
215 IF(L-ITOP)218,216,217
216 DN=DNOFXN(L)
XH=YN
GO TO 196
217 L=ITOP
GO TO 219
218 XN=YN
194 X=L-1
X=XLOW+X*DELX
C9876 WRITE(NOUTPE,9876) N,L,DNOFXN(L),FN(L),DNOFXN(L),XN,X,DELX
FORMAT(12I5,5X,3F5.2,3F10.3)
DN=DNOFXN(L)+(DNOFXN(L+1)-DNOFXN(L))*(XN-X)/DELX

```

DP604810
 DP604820
 DP604830
 DP604840
 DP604850
 DP604860
 DP604870
 DP604880
 DP604890
 DP604900
 DP604910
 DP604920
 DP604930
 DP604940
 DP604950
 DP604960
 DP604970
 DP604980
 DP604990
 DP605000
 DP605010
 DP605020
 DP605030
 DP605040
 DP605050
 DP605060
 DP605070
 DP605080
 DP605090
 DP605100
 DP605110
 DP605120
 DP605130
 DP605140
 DP605150
 DP605160
 DP605170
 DP605180
 DP605190
 DP605200
 DP605210
 DP605220
 DP605230
 DP605240
 DP605250
 DP605260
 DP605270
 DP605280

```

196 CALL TRANFM
IF(NN-50) 203,203,220
220 NPAGE=NPAGE+1
C WRITE(NOUTPE,152) NPAGE
GO TO 204
112 WRITE(NOUTPE,113)
WRITE(NOUTPE,1113) MAX,ITOP
113 FORMAT(52H THIS PROGRAM LIMITS THE NUMBER OF DISCRETE STEPS OF)
1113 FORMAT(21H THE STATE VARIABLE TOI6,26H AND THIS PROBLEM REQUIRESI6)
2002 FORMAT(11,1, ERROR HALT')
2003 WRITE(NOUTPE,2002)
STOP
2000 WRITE(NOUTPE,2001)
2001 FORMAT(11,1, END OF DATA FILE')
STOP
END
CSEARCH DISCRETE FIBONACCI SEARCH SUBROUTINE
SUBROUTINE SEARCH(AA,BB,DELYY,XXMODE,YBEST,BEST)
DIMENSION F(150)
C
C OPTIMIZE WITH RESPECT TO Y BETWEEN AA AND 3B IN STEPS OF DELYY
C STORE OPTIMUM Y IN YBEST AND THE OPTIMUM VALUE OF THE OBJECTIVE
C FUNCTION IN BEST
C XXMODE=-1 FOR MAXIMIZE
C XXMODE=1 FOR MINIMIZE
C SUBROUTINE MUST BE INITIALIZED BY CALLING IT WITH DELYY=-1.0 AT
C LEAST ONCE BEFORE IT IS USED FOR A SEARCH
DELYY=DELYY
A=AA
B=BB
XXMODE=XXMODE
IF(DELYY) 100,100,101
100 F{1}=1.0
F{2}=2.0
DO 1 I=3,150
F(I)=F(I-1)+F(I-2)+1.0
IF(F(I)-1.0E+35) 1,2,2
2 II=I
RETURN
1 CONTINUE
II=150
RETURN
101 YHI=B
YLC=A
FNO=(YHI-YLO)/DELYY+1.0
DO 5 N=1,II
IF(FNO-F(N)) 6,6,5
5 CONTINUE

```

DP605290
 DP605300
 DP605310
 DP605320
 DP605330
 DP605340
 DP605350
 DP605360
 DP605370
 DP605380
 DP605390
 DP605400
 DP605410
 DP605420
 DP605430
 DP605440
 DP605450
 DP605460
 DP605470
 DP605480
 DP605490
 DP605500
 DP605510
 DP605520
 DP605530
 DP605540
 DP605550
 DP605560
 DP605570
 DP605580
 DP605590
 DP605600
 DP605610
 DP605620
 DP605630
 DP605640
 DP605650
 DP605660
 DP605670
 DP605680
 DP605690
 DP605700
 DP605710
 DP605720
 DP605730
 DP605740
 DP605750
 DP605760

```

7      WRITE(6,7) 1ERROR, TOO MANY POINTS TO BE SEARCHED)
2002  FORMAT(1,1,ERROR HALT,)
2003  WRITE(6,2002)
      STOP
      6      IF(N-2) 42,41,40
      40      Y1=F(N-2)*DELY+YLO
      CALL FUNCTN(Y1,GY1)
      Y2=F(N-1)*DELY+YLO
      CALL FUNCTN(Y2,GY2)
      16      N=N-1
      103     IF(N-1) 102,102,103
      19      IF((GY2-GY1)*XMODE) 19,19,20
      YLO=Y1+DELY
      Y1=Y2
      GY1=GY2
      Y2=F(N-1)*DELY+YLO
      IF(Y2-B) 104,104,105
      105     GY2=1.0E+35*XMODE
      GO TO 16
      104     CALL FUNCTN(Y2,GY2)
      GO TO 16
      20      YHI=Y2-DELY
      Y2=Y1
      GY2=GY1
      IF(N-2) 34,34,35
      34      FNM2=0.0
      GO TO 36
      35      FNM2=F(N-2)
      36      Y1=FNM2*DELY+YLO
      CALL FUNCTN(Y1,GY1)
      GO TO 16
      102     IF((GY2-GY1)*XMODE) 110,111,111
      110     YIBEST=Y2
      BEST=GY2
      FETURN
      111     YIBEST=Y1
      BEST=GY1
      FETURN
      41      Y1=A
      CALL FUNCTN(Y1,GY1)
      Y2=B
      CALL FUNCTN(Y2,GY2)
      GO TO 102
      42      Y1=A
      CALL FUNCTN(Y1,GY1)
      GO TO 111
      END

```

DP605770
 DP605780
 DP605790
 DP605800
 DP605810
 DP605820
 DP605830
 DP605840
 DP605850
 DP605860
 DP605870
 DP605880
 DP605890
 DP605900
 DP605910
 DP605920
 DP605930
 DP605940
 DP605950
 DP605960
 DP605970
 DP605980
 DP605990
 DP606000
 DP606010
 DP606020
 DP606030
 DP606040
 DP606050
 DP606060
 DP606070
 DP606080
 DP606090
 DP606100
 DP606110
 DP606120
 DP606130
 DP606140
 DP606150
 DP606160
 DP606170
 DP606180
 DP606190
 DP606200
 DP606210
 DP606220
 DP606230
 DP606240

DP606250
DP606260
DP606270
DP606280
DP606290
DP606300
DP606310
DP606320
DP606330
DP606340
DP606350
DP606360
DP606370
DP606380
DP606390
DP606400
DP606410
DP606420
DP606430
DP606440
DP606450
DP606460
DP606470
DP606480
DP606490
DP606500
DP606510
DP606520
DP606530
DP606540
DP606550
DP606560
DP606570
DP606580
DP606590
DP606600
DP606610
DP606620
DP606630
DP606640
DP606650
DP606660
DP606670
DP606680
DP606690
DP606700
DP606710
DP606720

```
CFUNCTN
SUBROUTINE FUNCTN(Y,GY)
DIMENSION FN(1000),FNM1(10001),DNOFXN(100) 1)
REAL * 8 TAB(4)
C      * 8 A,B,E,SUMELT,RN,IS,TERM,TEMP,AB,CDF,IDN
COMMON /CON/TAB
COMMON XN,N,DN,YN,RN,SLOW,XHIGH,DELY,DLOW,DHIGH,DELD,NUMBER
COMMON YLOW,YHIGH,DELY,FN,JTOP,NNFLAG,NMAX,XLAM
COMMON A,B,C,D,E,F,G,V,SUMELT,KODE1,KODE2,KODE3
EQUIVALENCE (FN(1),FNM1(1),DNOFXN(1))
DN=Y
NOUTPE=12
JUMP MAY BE SET TO 1 IN TRANFM TO INDICATE AN INFEASIBLE XN-1
C
C      CALL TRANFM
CALL STGRET
XMCDE=1 FOR MIN, -1 FOR MAX
IF (JUMP.EQ.1) RN=999999*XMDE
IF (N-1) 146,146,401
401 K=(YN-YLOW)/DELY+1.001
IF (K) 212,212,209 N,XN,DN
212 WRITE(NOUTPE,210) N,XN,DN
210 WRITE(NOUTPE,210) YN
1210 FORMAT(//9H AT STAGE15,9H WITH XN=E15.8,8! AND DN=E15.8)
2002 FORMAT(11,1,ERROR HALT)
2003 WRITE(NOUTPE,2002)
STOP
IF (K-JTOP) 205,206,212
206 RNM1=FNM1(JTOP)
GO TO 207
205 X=K-1
X=YLOW+X*DELY
RNM1=FNM1(K)+(FNM1(K+1)-FNM1(K))*(YN-X)/DELY
207 GO TO (21,22,23,24),NNFLAG
21 CN=RN+RNM1
GO TO 137
22 CN=RN-RNM1
GO TO 137
23 IF (RN-RNM1) 141,141,142
141 QN=RN
GO TO 137
142 CN=ENM1
GO TO 137
24 IF (RN-RNM1) 145,145,146
145 QN=ENM1
GO TO 137
146 QN=RN
```



```

C20 F(1) = V(1) - (INT(V(1)/C(1))) * C(1)
20 G(1) = 0.0
C C(1) = (INT((V(1)+INC)/C2)+1) * C2
C W = (INT(V(1)/C2)+1) * C(1)
C W = (INT(V(1)/C(1))+1) * C(1)
C21 WRITE(1BALL,21) F(1), V(1), C(1), W
C98 FORMAT(4F10.2)
C DO 30 I=2,NMAX
C IF (C(I) .LE. 100.) FACTOR=10.0
C IF (C(I) .LE. 50.) FACTOR=100.0
C INC=FACTOR*C(I+1)
C F(I) = (INT((V(I)-INC)/C2)+1) * C2
C G(I) = (INT((V(I)+INC)/C2)+1) * C2
C FTEST=V(I) - INC
C GTEST=V(I) + INC
C IF (F(I) .LE. FTEST) GO TO 60
C IF (F(I) .LE. F(I) - C(I))
C GO TO 30
C IF ((F(I) - GE.F(I-1)) - AND.(F(I) - GE.0.0)) GO TO 70
C GO TO 30
C50 IF (F(I) - GE.F(I-1)) - AND.(F(I) - GE.0.0)) GO TO 70
C50 IF (F(I) - GE.F(I-1)) - AND.(F(I) - GE.0.0)) GO TO 80
C70 IF (G(I) - GE.GTEST) - AND.(G(I) - GE.G(I-1)) GO TO 80
C75 IF (G(I) - GE.G(I-1)) GO TO 80
C75 IF (G(I) - GE.G(I) + C2)
C GO TO 70
C80 WRITE(NOUTPE,1) I, V(I), XN1, F(I), G(I), FTEST, GTEST
C IF (G(I) .LE. XN1) GO TO 90
C G(I) = G(I) - C(I)
C GO TO 80
C90 CONTINUE
C301 CONTINUE
C F(NMAX) = XN1
C G(NMAX) = INT(XN1/C2) * C2
C IF (K2 .NE. 0) GO TO 129
C WRITE(1BALL,131)
C WRITE(1OUTPE,131)
C131 WRITE(NOUTPE,131)
C129 FORMAT(' NO BOUNDS ARE APPROACHED')
C GO TO 30
C127 WRITE(1BALL,127)

```

```

ADJ00460
ADJ00470
ADJ00480
ADJ00490
ADJ00500
ADJ00510
ADJ00520
ADJ00530
ADJ00540
ADJ00550
ADJ00560
ADJ00570
ADJ00580
ADJ00590
ADJ00600
ADJ00610
ADJ00620
ADJ00630
ADJ00640
ADJ00650
ADJ00660
ADJ00670
ADJ00680
ADJ00690
ADJ00700
ADJ00710
ADJ00720
ADJ00730
ADJ00740
ADJ00750
ADJ00760
ADJ00770
ADJ00780
ADJ00790
ADJ00800
ADJ00810
ADJ00820
ADJ00830
ADJ00840
ADJ00850
ADJ00860
ADJ00870
ADJ00880
ADJ00890
ADJ00900
ADJ00910
ADJ00920
ADJ00930

```



```

C820 WRITE (IBALL,819)N,XN,XMIN1L,XMIN1H,C(N),DL,W,DHIGH
C819 FORMAT(1X,I5,6F10.2)
C821 CONTINUE
C      DLOW=AMIN1(DLOW,DHIGH)
C      DEID=1
C      IF (N.GT.1).AND.(C(N).GT.1.0) GO TO 36
C      IF (N.EQ.1) DLOW=0
C      IF (N.EQ.1) DHIGH=10.0
C      WRITE (NOUTPE,50)XN,XMIN1L,XMIN1H,C(N),DLOW,DHIGH
C50  FORMAT(6F10.2)
36  RETURN
37  AB=A(N)*B(N)
DO 555 ID=1,30
IS=ID-1
C USE THE SECOND ONE FOR DOUBLE PRECISION
C      TERM=EXP(-AB)
C      TERM=DEXP(-AB)
C      TEMP=TERM
C      IF (IS.EQ.0) GO TO 11
C      DO 10 I=1,IS
C      TEMP=TEMP*AB/I
C      IF (TERM.GE..99999) GO TO 11
C      IF (TEMP.LE..00001) GO TO 11
10  TERM=TERM+TEMP
11  CDF=TERM
40  TWUS=(1.-CDF)*(AB*AB-2.*AB*IS+IS*(IS+1))/(2.*A(N))
X+TEMP*B(N)*(AB-IS)/2.
C CHANGED FOLLOWING TO 555 FROM 556
C      IF (KODE2.EQ.2.AND.E(N)*TWUS/SUMELT.LE.XN)3) TO 556
C      IF (KODE2.EQ.2) GO TO 555
C      IF (KODE2.EQ.3) GO TO 554
C      SSS=(1.-TEMP)+(IS-AB)*(1.-CDF)/AB
C      RR=E(N)*AB*SSS/SUMELT
C      Y=XN-RR
C      DLOW=0.
C THE NUMBER IN THE NEXT LINE HAS A BIG EFFECT. IT SHOULD NOT.
C      IF (N.EQ.1) DLOW=1000.
C      IF (N.EQ.1.AND.Y.GT.0.-AND.ID.LT.30) GO TO 555
C      IF (N.EQ.1.AND.Y.LE.0.) DLOW=IS
136  DHIGH=30+DLOW
C      IF (DLOW.EQ.1000.) DHIGH=1000.
C      DEID=1.
C      RETURN
554  AMSRT=TWUS/AB
      AMTBF=1./A(N)
      AMTR=D(N)
      AV=AMTBF/(AMTBF+AMTR+AMSRT)
      IF (AV.GE..XN) GC TO 556

```

DLI00250
 DLI00260
 DLI00270
 DLI00280
 DLI00290
 DLI00300
 DLI00310
 DLI00320
 DLI00330
 DLI00340
 DLI00350
 DLI00360
 DLI00370
 DLI00380
 DLI00390
 DLI00400
 DLI00410
 DLI00420
 DLI00430
 DLI00440
 DLI00450
 DLI00460
 DLI00470
 DLI00480
 DLI00490
 DLI00500
 DLI00510
 DLI00520
 DLI00530
 DLI00540
 DLI00550
 DLI00560
 DLI00570
 DLI00580
 DLI00590
 DLI00600
 DLI00610
 DLI00620
 DLI00630
 DLI00640
 DLI00650
 DLI00660
 DLI00670
 DLI00680
 DLI00690
 DLI00700
 DLI00710
 DLI00720

```

555 CONTINUE
IS=30
556 DLOW=IS
DELD=1.
DHIGH=50.
RETURN
END
BLOCK DATA
COMMON /CON/TAP
REAL * 8 TAB(4) /' SUM', ' MULT', 'MAXMIN', 'MINMAX' /
END
CENTER2
SUBROUTINE ENTER2
DIMENSION FN(10001), FNM1(10001), DNOFXN(10001), A(101), B(101), C(101)
DIMENSION D(101), E(101), F(101), G(101), S(101), V(101)
REAL * 8 A, B, E, SUMELT, RN, TERM, TEMP, AB, CDF
REAL * 8 TAB(4)
COMMON /CON/TAB
COMMON XN, N, DN, YHIGH, DELX, XHIGH, DELX, DLOW, DHIGH, DELD, NUMBER
COMMON YLOW, YHIGH, DELY, NUFF, FN, JTOP, NNFLAG, NMAX, XLAM
COMMON A, B, C, D, E, F, G, V, SUMELT, KODE1, KODE2, KODE3
COMMON XMIN1, XMINH, XMINH, KEEP, NEXT1, XN1, XN2
EQUIVALENCE (FN(1), FNM1(1), DNOFXN(1))
KEEP=9
NEXT1=10
IBALL=11
NOUTPE=12
INDEC=14
IFILE=15
FACTOR=2.0
NMAX1=NMAX-1
TOTAL=0.0
READ (INDEC, 2) NMAX, MTAPE, SOLVE, ITOTAL
WRITE (IBALL, 14) NMAX
FORMAT (I5)
DO 100 I=1, NMAX
READ (INDEC, 11) S(I)
FORMAT (F10.2)
POINT=S(I)*C(I)
TOTAL=TOTAL+POINT
WRITE (IBALL, 11) TOTAL
V(I)=TOTAL
CONTINUE
F(NMAX)=V(NMAX)
G(NMAX)=V(NMAX)
IF (ITOTAL.EQ.0) GO TO 9
READ (INDEC, 4) XN1, XN2

```

DLI00733
DLI00740
DLI00750
DLI00760
DLI00770
DLI00780
DLI00790
DLI00800
DLI00810
DLI00820
DLI00830
ENT00010
ENT00020
ENT00030
ENT00040
ENT00050
ENT00060
ENT00070
ENT00080
ENT00090
ENT00100
ENT00110
ENT00120
ENT00130
ENT00140
ENT00150
ENT00160
ENT00170
ENT00180
ENT00190
ENT00200
ENT00210
ENT00220
ENT00230
ENT00240
ENT00250
ENT00260
ENT00270
ENT00280
ENT00290
ENT00300
ENT00310
ENT00320
ENT00330
ENT00340
ENT00350
ENT00360
ENT00370

```

9      F(NMAX)=XN1
C      G(NMAX)=XN2
C      IF (C(1).LE.100.)FACTOR=10.
C      IF (C(1).LE.50.)FACTOR=100.
C      INC=FACTOR*C(2)
C      TEST=V(1)+INC
C      WRITE(16,12)S(1),C(1),TOTAL,V(1),TEST,INC
C      12  FORMAT(6F10.2)
C      10  IF (G(1).GE.TEST) GO TO 20
C          GO TO 10
C          GO TO 10
C      F(1)=V(1)-(INT(V(1)/C(1)))*C(1)
C      W=(INT(V(1)/C(1))+1)*C(1)
C      98  WRITE(16,14)F(1),G(1)
C      DO 30 J=2,NMAX1
C          IF (C(J).LE.100.)FACTOR=10.0
C          IF (C(J).LE.50.)FACTOR=100.0
C          INC=FACTOR*C(J+1)
C          F(J)=V(J)
C          G(J)=V(J)
C          PTEST=V(J)-INC
C          GTEST=V(J)+INC
C          45  IF (F(J).LE.FTEST) GO TO 60
C          IF (F(J)=F(J)-C(J))
C              GO TO 30
C          IF ((F(J)-GE.F(J-1))-AND.(F(J).GE.0.0)) GO TO 70
C          60  F(J)=F(J)+C(J)
C          GO TO 60
C          IF ((G(J)-GE.GTEST)-AND.(G(J).GE.G(J-1))) GO TO 80
C          75  G(J)=G(J)+C(J)
C          GO TO 70
C          IF (G(J).LE.XN1) GO TO 90
C          G(J)=G(J)-C(J)
C          GO TO 80
C          CONTINUE
C      90  WRITE(16,12)S(J),C(J),TOTAL,V(J),FTEST,GTEST,INC
C      13  FORMAT(7F8.2)
C      301  CCCONTINUE
C          IF (ITOTAL.EQ.0) GO TO 99
C          G(NMAX1)=XN2
C      99  CONTINUE
C          WRITE(16,2)NMAX,MTAPE,SOLVE,ITOTAL
C      2    FORMAT(2I5,A5,I5)
C          K1=1
C          F2=0.0
C          WRITE(16,3)K1,K1,F(K1),G(K1),C(K1),F2,W
C          DO 400 K=2,NMAX

```

```

ENT000380
ENT000390
ENT000400
ENT000410
ENT000420
ENT000430
ENT000440
ENT000450
ENT000460
ENT000470
ENT000480
ENT000490
ENT000500
ENT000510
ENT000520
ENT000530
ENT000540
ENT000550
ENT000560
ENT000570
ENT000580
ENT000590
ENT000600
ENT000610
ENT000620
ENT000630
ENT000640
ENT000650
ENT000660
ENT000670
ENT000680
ENT000690
ENT000700
ENT000710
ENT000720
ENT000730
ENT000740
ENT000750
ENT000760
ENT000770
ENT000780
ENT000790
ENT000800
ENT000810
ENT000820
ENT000830
ENT000840
ENT000850

```

```

400      WRITE (IFILE,3) K,K,F(K),G(K),C(K),F(K-1),G(K-1)
CONTINUE
3      FORMAT(2I5,3F10.2,' MIN SUM 2',2F9.2)
WRITE (IFILE,4) F(NMAX),G(NMAX)
4      FORMAT(2F15.2)
1      FORMAT(I5,5F10.2)
9      FORMAT(I5)
900     RETURN
END
CSTGRET
SUBROUTINE STGRET
DIMENSION FN(1000),FNM1(1000),DNOFXN(100),A(101),B(101),C(101)
DIMENSION D(101),E(101),F(101),G(101),V(101)
REAL * 8 A,B,E,SUMELT,RN,TERM,TEMP,AB,CDF
COMMON /CON/TAB(4)
COMMON XN,N,DN,YN,RN,XLOW,XHIGH,DELX,DLOW,HIGH,DELD,NUMBER
COMMON YLOW,YHIGH,DELY,NUFF,FN,JTOP,NFLAG,NMAX,XLAM
COMMON A,B,C,D,E,F,G,V,SUMELT,KODE1,KODE2,KODE3
COMMON XMIN,XMIN1,XMINH,KEEP,NEXT1,XN1,XN2
EQUIVALENCE (FN(1),FNM1(1),DNOFXN(1))
CONTINUE
NEXT1=10
IBALL=11
NOUTPE=12
IF (KODE1.EQ.0) GO TO 77
IDN=INT(DN)
AB=A(IN)*B(N)
IS=IDN
TERM=0.
TEMP=0.
IF (IS.LT.0) GO TO 11
IF (AB.LE.170.0) GO TO 12
TERM=0.0
GO TO 13
C USE THE SECOND ONE FOR DOUBLE PRECISION
12 TERM=EXP(-AB)
C12 TERM=DEXP(-AB)
13 TEMP=TERM
IF (IS.EQ.0) GO TO 11
DO 10 I=1,IS
TEMP=TEMP/I
CALL OVERFL (JOY)
IF (JOY.NE.2) WRITE(6,999) TEMP,TERM,A(N),B(N),AB,I,IS
C999 FORMAT(1X,5E13.5,2I5)
IF (TERM.GE..99999) GO TO 11
C IF (TEMP.LE..00001) GO TO 11

```

```

ENT000860
ENT000870
ENT000880
ENT000890
ENT000900
ENT000910
ENT000920
ENT000930
ENT000940
ENT000950
ENT000960
ENT000970
ENT000980
ENT000990
ENT001000
ENT001010
ENT001020
ENT001030
ENT001040
ENT001050
ENT001060
ENT001070
ENT001080
ENT001090
ENT001100
ENT001110
ENT001120
ENT001130
ENT001140
ENT001150
ENT001160
ENT001170
ENT001180
ENT001190
ENT001200
ENT001210
ENT001220
ENT001230
ENT001240
ENT001250
ENT001260
ENT001270
ENT001280
ENT001290
ENT001300
ENT001310
ENT001320
ENT001330
ENT001340
ENT001350
ENT001360
ENT001370
ENT001380

```

```

10 TERM=TERM +TEMP
11 CDF=TERM
20 CONTINUE
  IF (CODE1.NE.1) GO TO 40
  C FOLLOWING CHANGED 120183
  C
  SMA= (AB* (1.-TEMP) + (IDN-AB) * (1.-CDF)) / AB
  SMA= (1.-TEMP) + (IDN-AB) * (1.-CDF) / AB
  RN=E(N)*AB*SMA/SUMELT
  RETURN
40 TWUS= (1.-CDF) * (AB*AB-2.*AB*IDN+IDN*(IDN+1)) / (2.*A(N))
  X+TEMP*B(N)*(AB-IDN)/2.
  WRITE(NEXT1,101)KODE1
  C101 FORMAT(I3)
  IF (KODE1.EQ.3) GO TO 50
  IF (XN.GE.XMIN1) GO TO 48
  C 47 WRITE(NEXT1,100)XN,XMIN1
  C100 FORMAT(2F10.2)
  RN=1.0*MODES*10.0E5
  GO TO 49
  C 48 RN=E(N)*TWUS/SUMELT
  C148 RN1=E(N)*TWUS
  IF ((N.LT.3).AND.(C(N).LT.15.0)) GO TO 103
  C WRITE(IBALL,102)XN,DN,RN,RN1
  C102 FORMAT(6X,2F10.2,2F15.9)
  C103 CONTINUE
  RETURN
50 CONTINUE
  AMSRT=TWUS/AB
  AMTBF=1./A(N)
  AMTTR=D(N)
  RN=(AMTBF)/(AMTBF+AMTTR+AMSRT)
  RETURN
77 RN=C(N)*DN
  C 80 IF (KODE2.EQ.1.AND.N.EQ.1.AND.YN.LT.XN) RN=9) 9999
  RETURN
  END
CSTORE
  SUBROUTINE STORE
  DIMENSION FN(10001),FNM1(10001),DNOFXN(10001),A(101),B(101),C(101)
  DIMENSION D(101),E(101),F(101),G(101),V(101)
  REAL * 8 TAB(4)
  COMMON /CON/TAB
  COMMON XN,N,DN,XN,RN,XLOW,XHIGH,DELX,DLOW,HIGH,DEL,D,NUMBER
  COMMON YLOW,YHIGH,DEL,Y,NUFF,FN,STOP,NFLAG,NMAX,XIAM
  COMMON ALB,C,D,E,F,G,V,SUMELT,KODE1,KODE2,KODE3
  COMMON XMIN1,XMINH,KEEP,NEXT1,XN1,XN2
  EQUIVALENCE (FN(1),FNM1(1),DNOFXN(1))

```

STG000390
 STG000400
 STG000410
 STG000420
 STG000430
 STG000440
 STG000450
 STG000460
 STG000470
 STG000480
 STG000490
 STG000500
 STG000510
 STG000520
 STG000530
 STG000540
 STG000550
 STG000560
 STG000570
 STG000580
 STG000590
 STG000600
 STG000610
 STG000620
 STG000630
 STG000640
 STG000650
 STG000660
 STG000670
 STG000680
 STG000690
 STG000700
 STG000710
 STG000720
 STG000730
 STG000740
 STG000750
 STG000760
 STG000770
 STG000780
 STG000790
 STG000800
 STG000810
 STG000820
 STG000830
 STG000840
 STG000850
 STG000860
 STG000870
 STG000880
 STG000890
 STG000900
 STG000910
 STG000920
 STG000930
 STG000940
 STG000950
 STG000960
 STG000970
 STG000980
 STG000990
 STG001000
 STG001010
 STG001020
 STG001030
 STG001040
 STG001050
 STG001060
 STG001070
 STG001080
 STG001090
 STG001100
 STG001110
 STG001120

```

INLATA=3
NOUTPE=12
IOUTPE=6
XMINIH=0.0
READ (INDATA,808) KODE1,KODE2,KODE3
WRITE (NOUTPE,808) KODE1,KODE2,KODE3
808 FORMAT(3I3)
IF (KODE1.EQ.0) WRITE (NOUTPE,110)
IF (KODE1.EQ.1) WRITE (NOUTPE,101)
IF (KODE1.EQ.2) WRITE (NOUTPE,202)
IF (KODE1.EQ.3) WRITE (NOUTPE,303)
IF (KODE1.EQ.3) THE OBJECTIVE IS COST
110 FORMAT(30H THE OBJECTIVE IS COST
101 FORMAT(30H THE OBJECTIVE IS SMRT
202 FORMAT(30H THE OBJECTIVE IS MSRT
303 FORMAT(30H THE OBJECTIVE IS AVAILABILITY)
IF (KODE2.EQ.0) WRITE (NOUTPE,900)
IF (KODE2.EQ.1) WRITE (NOUTPE,901)
IF (KODE2.EQ.2) WRITE (NOUTPE,902)
IF (KODE2.EQ.3) WRITE (NOUTPE,903)
IF (KODE2.EQ.3) CONSTRAINT ON COST
900 FORMAT(30H CONSTRAINT ON SMRT
901 FORMAT(30H CONSTRAINT ON MSRT
902 FORMAT(30H CONSTRAINT ON AVAILABILITY
903 WRITE (NOUTPE,1) NMAX
C1
C1
DO 333 K=1,NMAX
READ (INDATA,999) A(K),C(K),E(K),B(K),D(K)
999 FORMAT(5F10.4)
C 333 WRITE (NOUTPE,999) A(K),B(K),C(K),D(K),E(K)
SUMELT=0
DO 30 I=1,NMAX
30 SUMELT=SUMELT+E(I)*A(I)*B(I)
765 WRITE (NOUTPE,765) SUMELT
FORMAT(F10.3)
RETURN
END
CTRANFM
SUBROUTINE TRANFM
DIMENSION FN(10001),FNM1(10001),DNOFXN(100,1),A(101),B(101),C(101)
DIMENSION D(101),E(101),F(101),G(101),V(101)
REAL *8 A,B,E,SUMELT,RN,TERM,TEMP,AB,CDF
REAL *8 TAB(4)
COMMON /CON/TAE
COMMON YN,N,DN,YN,RN,XLOW,XHIGH,DELX,DLOW,,HIGH,DELD,NUMBER
TI.A000130
ST000140
ST000150
ST000160
ST000170
ST000180
ST000190
ST000200
ST000210
ST000220
ST000230
ST000240
ST000250
ST000260
ST000270
ST000280
ST000290
ST000300
ST000310
ST000320
ST000330
ST000340
ST000350
ST000360
ST000370
ST000380
ST000390
ST000400
ST000410
ST000420
ST000430
ST000440
ST000450
ST000460
ST000470
ST000480
ST000490
ST000500
ST000510
ST000520
TRA00010
TRA00020
TRA00030
TRA00040
TRA00050
TRA00060
TRA00070
TIA00080

```

```

COMMON YLOW, YHIGH, DELY, NUFF, EN, JTOP, NNFLAG, NMAX, XLAM
COMMON A, B, C, D, E, F, G, V, SUMELT, KODE1, KODE2, KODE3
COMMON XMIN1, XMIN1L, XMIN1H, KEEP, NEXT1, XN1, XN2
EQUILVALENCE (FN(1), FN(1), DNOFAN(1))
IF (KODE2.NE.0) GC TO 12
IF (XN.GE.XMIN1L) GO TO 30
  YN=XMIN1L
  GO TO 31
30 YN=XN-C(N)*DN
31 RETURN
  IDN=INT(DN)
  AB=A(N)*B(N)
  IS=IDN
  TERM=0.
  TEMP=0.
  IF (IS.LT.0) GO TO 11
  THE SECOND ONE FOR DOUBLE PRECISION
  C USE TERM=EXP(-AB)
  C TERM=DEXP(-AB)
  TEMP=TERM
  IF (IS.EQ.0) GO TO 11
  DO 10 I=1, IS
    TEMP=TEMP*AB/I
  IF (TEMP.GE..99999) GO TO 11
  IF (TEMP.LE..00001) GO TO 11
  C TERM=TERM +TEMP
  10 CDF=TERM
  11 TWUS=(1.-CDF)* (AB*AB-2.*AB*IDN+IDN*(IDN+1))/(2.*A(N))
  40 X+TEMP*B(N)* (AB-IDN)/2.
  IF (KODE2.EQ.2) GO TO 50
  IF (KODE2.EQ.3) GO TO 51
  SMA=(1.-TEMP)+(IDN-AB)*(1.-CDF)/AB
  YN=XN-E(N)*AB*SMA/SUMELT
  IF (YN.LT.0) YN=0.0
  C IF (N.NE.1.AND.YN.LT.0.OR.N.NE.1.AND.YN.GT.1.) JUMP=1
  C IF (N.EQ.1.AND.YN.GT.0) JUMP=1
  C IF (JUMP.EQ.1) YN=0.
  51 RETURN
  AMSRT=TWUS/AB
  AMTBF=1./A(N)
  AMTTR=D(N)
  AV=(AMTBF)/(AMTBF+AMTTR+AMSRT)
  YN=XN/AV
  AV IS AVAL
  RETURN
  50 YN=XN-E(N)*TWUS/SUMELT

```

TRA00090
 TRA00100
 TRA00110
 TRA00120
 TRA00130
 TRA00140
 TRA00150
 TRA00160
 TRA00170
 TRA00180
 TRA00190
 TRA00200
 TRA00210
 TRA00220
 TRA00230
 TRA00240
 TRA00250
 TRA00260
 TRA00270
 TRA00280
 TRA00290
 TRA00300
 TRA00310
 TRA00320
 TRA00330
 TRA00340
 TRA00350
 TRA00360
 TRA00370
 TRA00380
 TRA00390
 TRA00400
 TRA00410
 TRA00420
 TRA00430
 TRA00440
 TRA00450
 TRA00460
 TRA00470
 TRA00480
 TRA00490
 TRA00500
 TRA00510
 TRA00520
 TRA00530
 TRA00540
 TRA00550
 TRA00560

IF (YN.LE.O.) YN=O.
RETURN
END

TRA00570
TRA00580
TRA00590

LIST OF REFERENCES

1. Naval Postgraduate School Report NPS55-83-026, Wholesale Provisioning Models: Model Development, by F. Russell Richards and Alan W. McMaster, September 1983.
2. Department of Defense Instruction 4140.42, "Determination of Initial Requirements for Secondary Item Spares and Repair Parts", 7 August 1974.
3. Naval Postgraduate School Report NPS55-83-028, Wholesale Provisioning Models: Model Optimization, by Gilbert T. Howard, October 1983.
4. Fox, Bennett, "Discrete Optimization Via Marginal Analysis", Management Science, Vol 13, No. 3, pp. 210-215, November 1966.
5. Nemhauser, George, L., Introduction to Dynamic Programming, pp. 22-23, John Wiley and Sons, Inc., New York, 1966.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0142 Naval Graduate School Monterey, California 93943	2
3. Gilbert T. Howard Naval Postgraduate School Code 55HK Monterey, California 93943	2
4. F. Russell Richards Naval Postgraduate School Code 55RH Monterey, California 93943	2
5. LCDR. William A. Goulding 409 Camino Elevado Bonita, California 92002	3

END

FILMED

5-85

DTIC