CALCULATOR AND COMPUTER PROGRAMS FOR ELEMENTARY MULTIOBJECTIVE DECISION ANALYSIS*

by

Craig W. Kirkwood
and
Hector Ureta

Technical Report #77-11

Department of Industrial and Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109

December, 1977

This work was supported in part by the National Science Foundation under Grant No. ENG-74-22564.

ABSTRACT

This report describes calculator and computer programs to aid in carrying out multiobjective decision analyses. The programs assume that an additive or multiplicative utility function is valid and that the conditional utility functions over each attribute are constant or constant proportional risk averse. The attributes are assumed to be continuous and, once the alternative is specified, probabilistically independent. The Pearson-Tukey approximation is used to calculate expected utilities. The calculator program is written for a Hewlett-Packard HP-25 calculator, and the computer program is written in Level F PL/I.

TABLE OF CONTENTS

CALCULATOR AND COMPUTER PROGRAMS FOR ELEMENTARY MULTIOBJECTIVE DECISION ANALYSIS

The usefulness of multiobjective decision analysis has been established by a number of successful applications. (See, for example, [2, 4,5].) However, applications work would be simplified by easy-to-use computational aids, since the calculations needed in applications are sometimes tedious. The calculator and computer programs discussed in this report aid in a rapid preliminary analysis of a multiobjective decision problem. They assume a multiplicative or additive utility function is valid [6] and that the conditional utility functions over each attribute are either constant or constant proportional risk averse [11]. Also, the attributes are assumed continuous and, once the alternative is specified, probabilistically independent. The Pearson-Tukey approximation [9] is used to carry out the expected utility calculations.

Short calculator programs are used to determine basic parameters of the problem. These are then input to an interactive computer program which carries out the expected utility calculations required for a decision analysis. Sensitivity analysis can be done and the data for a particular decision problem can be stored for future use.

1. THEORETICAL BACKGROUND

Suppose $a_1, a_2, \ldots, a_M$ are the available alternatives in a decision problem and $X_1, X_2, \ldots, X_N$ is a set of attributes, or measures of effectiveness, which describe the possible consequences of the alternatives. Then, if the axioms of decision theory [12] are to be obeyed the alternative a_m should be selected which maximizes the expected utility

$$E[u|a_m] = \int_{x_1} \cdots \int_{x_N} u(x_1,x_2,\ldots,x_N) f_m(x_1,x_2,\ldots,x_N) dx_1, dx_2, \ldots, dx_N \quad (1)$$

where f_m is the probability density function over $\{X_1,X_2,\ldots,X_N\}$ given that a_m is selected, and u is the von Neumann-Morgenstern utility function.

Utility function structure. The programs described in this report assume that for any alternative the attributes are mutually probabilistically independent [1] so that

$$f_m(x_1,x_2,\ldots,x_N) = \prod_{n=1}^{N} f_m^n(x_n) \qquad (2)$$

where $f_m^n(x_n)$ is the marginal probability density function over X_n given that a_m is selected. Also, the attributes are assumed mutually utility independent [7, Theorem 6.1] so that either

$$u(x_1,x_2,\ldots,x_N) = \sum_{n=1}^{N} k_n u_n(x_n) \qquad (3a)$$

or

$$ku(x_1,x_2,\ldots,x_N) + 1 = \prod_{n=1}^{N} [kk_n u_n(x_n)+1], \qquad (3b)$$

where $u_n(x_n)$ is a conditional utility function over X_n, and the k_n's are scaling constants. The scaling constant $-1<k \neq 0$ is the solution to

$$k + 1 = \prod_{n=1}^{N} (kk_n+1). \tag{4}$$

If (2) and (3) hold then (1) can be rewritten as either

$$E[u|a_m] = \sum_{n=1}^{N} k_n E[u_n(x_n)|a_m] \tag{5a}$$

or

$$kE[u|a_m]+1 = \prod_{n=1}^{N} \{kk_n E[u_n(x_n)|a_m]+1\} \tag{5b}$$

where

$$E[u_n(x_n)|a_m] \equiv \int_{X_n} u_n(x_n) f_m^n(x_n) dx_n. \tag{5c}$$

Expected utility calculations. The single attribute expected utilities defined by (5c) are evaluated using the Pearson-Tukey approximation [9]

$$E[u_n(x_n)|a_m] \cong 0.630u_n(x_{nm}^{.50}) + 0.185[u_n(x_{nm}^{.95}) + u_n(x_{nm}^{.05})] \tag{6}$$

where $x^{.05}_{nm}$ = 0.05-fractile of $f^n_m(x_n)$

$x^{.50}_{nm}$ = 0.50-fractile of $f^n_m(x_n)$, and

$x^{.95}_{nm}$ = 0.95-fractile of $f^n_m(x_n)$.

Empirical work [3,10] indicates (6) is an accurate approximation for a wide variety of probability distributions.

Single attribute utility functions. Three different types of utility functions can be used for each attribute:

i) Constant risk averse [7,11] with increasing preferences:

$$u_n(x_n) \sim \begin{cases} -e^{-c_n x_n} & c_n>0 \\ x_n & c_n=0 \\ e^{-c_n x_n} & c_n<0 \end{cases} \qquad (7a)$$

ii) Constant risk averse [7,11] with decreasing preferences:

$$u_n(x_n) \sim \begin{cases} -e^{c_n x_n} & c_n>0 \\ -x_n & c_n=0 \\ e^{c_n x_n} & c_n<0 \end{cases} \qquad (7b)$$

iii) Constant proportional risk averse [7,11] with increasing preferences (for this case x_n must be positive):

$$u_n(x_n) \sim \begin{cases} -x^{-(c_n-1)} & c_n>1 \\ \log x_n & c_n=1 \\ x^{1-c_n} & c_n<1 \end{cases} \qquad (7c)$$

(In these expressions $\sim$ means "is strategically equivalent to," and c_n is an unspecified constant.)

Examination of (5), (6) and (7) shows that the expected utilities of the available alternatives will be completely determined if k_n, c_n, n=1, 2,...,N, and $x^{.05}_{nm}$, $x^{.50}_{nm}$, $x^{.95}_{nm}$; n=1,2,...,N; m=1,2,...,M, are specified where N is the number of attributes and M is the number of alternatives.

2. PROGRAM USER'S GUIDE

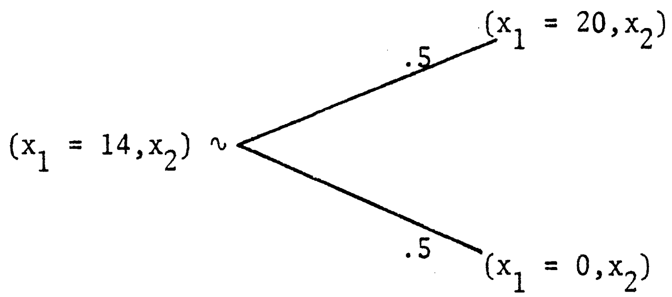
The programs described here include a calculator program and a computer program. The calculator program assists in determining the c_n's and k_n's. These, along with the fractiles for the probability distributions are input to the computer program. This calculates the expected utilities for the various alternatives. In addition, it allows the input data to be changed easily so that sensitivity analyses can be carried out.

The use of the programs will be explained with an example. A Decision Maker (DM) was considering a change in the process that his company used to manufacture widgets. His options were to select either of two new processes or to continue using the current process. A Decision Analyst (DA) was called in to aid in analyzing the problem.

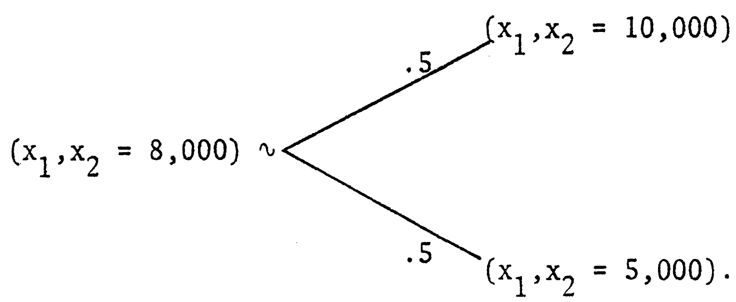
2.1 Data Collection

DM and DA decided there were two attributes of interest, X_1 = number of defects per batch of widgets and X_2 = cost, in dollars, to manufacture a batch of widgets. The ranges of interest for these were $0 \leq x_1 \leq 20$ and $\$5000 \leq x_2 \leq \$10,000$. For a preliminary analysis DA assumed X_1 and X_2 were mutually utility independent and that preferences were constantly risk averse and decreasing in each attribute.

DM decided on the following certainty equivalents:

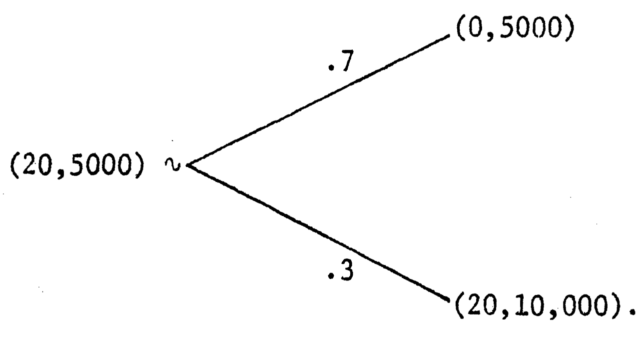


and

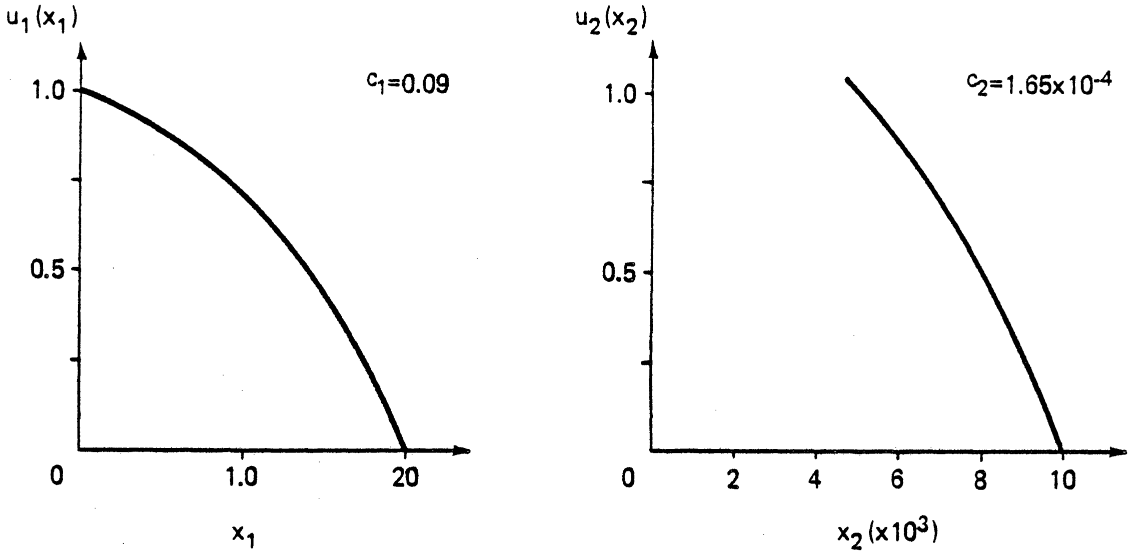


DA then used a calculator program to determine the risk aversion coefficients c_1 and c_2 for $u_1(x_1)$ and $u_2(x_2)$. This program is discussed in detail in Appendix A. It calculates the certainty equivalent for a two fork lottery for a utility function with a specified risk type (either constant or constant proportional risk averse) and risk aversion coefficient. The program can be used to calculate the risk aversion coefficient by trial and error if the certainty equivalent is known. For c_1 DA assumed constant risk aversion and tried 0.1, 0.05, 0.08 and finally 0.09, and for c_2 he also assumed constant risk aversion and tried 1×10^{-4}, 1.5×10^{-4}, 2×10^{-4}, 1.8×10^{-4}, 1.6×10^{-4}, 1.7×10^{-4} and finally 1.65×10^{-4}. Then, following the procedure discussed in Appendix A, he used the calculator to plot $u_1(x_1)$ and $u_2(x_2)$. These are shown in Figure 1a.

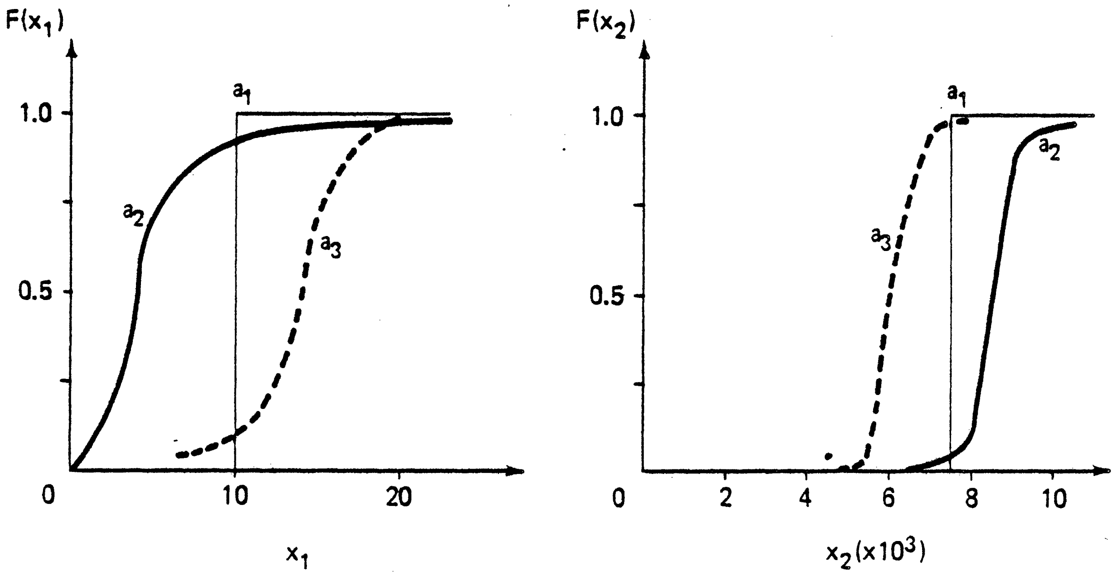
The scaling constants k_1 and k_2 were then assessed using Keeney and Raiffa's procedure [7]. In particular, DM decided (20,6000) $\sim$ (0,10,000) and



Thus, $k_1 = k_2 u_2(6000)$ and $k_2 = 0.7$. From Figure 1a, $u_2(6000) = .7$ and thus $k_1 = 0.49$.



a) CONDITIONAL UTILITY FUNCTIONS



b) MARGINAL PROBABILITY DISTRIBUTIONS

Figure 1. DECISION PROBLEM DATA

The marginal cumulative probability distributions were assessed for each alternative using standard techniques [14]. The resulting distributions are shown in Figure 1b. (Note that a_1 is the current manufacturing process and there is no uncertainty about its attribute values. Loosely speaking, a_2 is a high quality, high cost option while a_3 is a lower quality, lower cost alternative.) From Figure 1b the fractiles required for the Pearson-Tukey approximation were determined.

2.2 Expected Utility Calculations

After determining the utility and probability data DA left DM and went to a computer terminal to input the data to the computer program described in Appendix B. The data input session is recorded in Exhibit I. Note that the program requests data from the user in an interactive fashion. In general Exhibit I is self-explanatory. The program requests that files be attached to devices INP and STORE. INP is the file that will be used if the user tells the program to read the decision problem data from a file. (The use of this feature will be described later.) STORE is the file where the data for the current problem will be stored at the end of the computer session.

After specifying these two files the user must type "space RETURN" to start execution of the program. Numerical data entry is free-format, i.e., decimal points can be entered or not as desired. However, each number, including the last one on a line must be followed by a blank. (If you forget the last blank before RETURNing enter it on the next

EXHIBIT I

DECISION PROBLEM COMPUTER INPUT

```
#EXECUTION BEGINS
  INP       - SPECIFY FDNAME OR SEND END-OF-FILE
?WIDGIN
  STORE     - SPECIFY FDNAME OR SEND END-OF-FILE
?WIDGET




 ENTER 1 IF YOU WANT TO READ FROM FILE, ZERO IF NOT
 0




 ENTER NUMBER OF ATTRIBUTES AND ALTERNATIVES
 2 3

 ENTER RISK TYPE, CONSTANT AND RANGES FOR ATTRIBUTE: 1
 2 .09 0 20

 ENTER RISK TYPE, CONSTANT AND RANGES FOR ATTRIBUTE: 2
 2 .000165 5000 10000

 ENTER SCALING CONSTANT NUMBER: 1
 .49

 ENTER SCALING CONSTANT NUMBER: 2
 .7
```

EXHIBIT I (concluded)

```
ENTER FRACTILES FOR ALTERNATIVE: 1 AND ATTRIBUTE: 1
10 10 10

ENTER FRACTILES FOR ALTERNATIVE: 1 AND ATTRIBUTE: 2
7500 7500 7500

ENTER FRACTILES FOR ALTERNATIVE: 2 AND ATTRIBUTE: 1
1 4 14

ENTER FRACTILES FOR ALTERNATIVE: 2 AND ATTRIBUTE: 2
7500 8500 9500

ENTER FRACTILES FOR ALTERNATIVE: 3 AND ATTRIBUTE: 1
8 14 18

ENTER FRACTILES FOR ALTERNATIVE: 3 AND ATTRIBUTE: 2
5500 6000 7000
```

line and RETURN.) The program does a limited amount of error checking on input data, however, a serious error will terminate execution.

Since DA is entering a new problem he tells the program he does not want to read the data from a file. It then prompts him to enter the required data. RISK TYPE can be any of the three shown in Figure 2. CONSTANT is the value of the risk aversion coefficient c_n, and RANGES are the lower and upper limits of the range over which $u_n(x_n)$ will be assessed. SCALING CONSTANT is the value of k_n for the specified attribute. The required FRACTILES are the 0.05, 0.50 and 0.95.

After the data is collected the program summarizes it in tabular form and calculates expected utilities for each alternative. This output is shown in Exhibit II. INDIVIDUAL ATTRIBUTE EXPECTED UTILITY VALUES refers to $E[u_n(x_n)|a_m]$ as defined in (5a) and (5b).

After seeing this output DA signed off the computer. When the program terminated execution it stored the data for the decision problem in file WIDGET which DA had earlier specified as his STORE file.

2.3 Sensitivity Analysis

The computer program allows changes to be made in the data for a decision problem without re-entering the entire problem. Three types of changes can be made:

i) Additional alternatives can be added,

Risk Type	Explanation	Equation
1	Constant risk aversion with increasing preferences	$u_n(x_n) \sim \begin{cases} -e^{-c_n x_n} & c_n>0 \\ x_n & c_n=0 \\ e^{-c_n x_n} & c_n<0 \end{cases}$
2	Constant risk aversion with decreasing preference	$u_n(x_n) \sim \begin{cases} -e^{c_n x_n} & c_n>0 \\ -x_n & c_n=0 \\ e^{c_n x_n} & c_n<0 \end{cases}$
3	Constant proportional risk aversion with increasing preferences ($x_n>0$)	$u_n(x_n) \sim \begin{cases} -x^{-(c_n-1)} & c_n>1 \\ \log x_n & c_n-1 \\ x^{1-c_n} & c_n<1 \end{cases}$

Figure 2. ALLOWABLE RISK TYPES

EXHIBIT II

DECISION PROBLEM COMPUTER OUTPUT

***** MULTIATTRIBUTE DECISION ANALYSIS *****

ATTRIBUTES : 2

ALTERNATIVES : 3

- 0 - 0 - 0 - 0 - 0 -

----- INFORMATION ABOUT UTILITY FUNCTION

ATTRIBUTE	RISK TYPE	CONSTANT	RANGES LOWEST	RANGES HIGHEST
1	2	9.0000E-02	0.00	20.00
2	2	1.6500E-04	5000.00	10000.00

---- SCALING CONSTANTS

1	.4900
2	.7000

*** K = -0.5540

----INFORMATION ABOUT PROBABILITIES

ALTERNATIVE 1

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	10.00	10.00	10.00
2	7500.00	7500.00	7500.00

ALTERNATIVE 2

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	1.00	4.00	14.00
2	7500.00	8500.00	9500.00

ALTERNATIVE 3

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	8.00	14.00	18.00
2	5500.00	6000.00	7000.00

EXHIBIT II (concluded)

```
---- INDIVIDUAL ATTRIBUTE EXPECTED UTILITY VALUES

                    ALTERNATIVE:  1

          ATTRIBUTE            EXPECTED UTILITY
              1                     .7109
              2                     .6017

                    ALTERNATIVE:  2

          ATTRIBUTE            EXPECTED UTILITY
              1                     .8500
              2                     .3833

                    ALTERNATIVE:  3

          ATTRIBUTE            EXPECTED UTILITY
              1                     .4978
              2                     .8430


---- EXPECTED UTILITY FOR EACH ALTERNATIVE

       ALTERNATIVE       EXPECTED UTILITY
            1                 .6883
            2                 .6229
            3                 .7543




ENTER 1 IF YOU WANT SENSITIVITY ANALYSIS, ZERO IF NOT
0


#EXECUTION TERMINATED
#
```

ii) Additional attributes can be added, or

iii) Any of the data for the current alternatives and attributes can be changed.

Only one of these three types of changes can be made at a time. Following the changes, the new data and expected utilities are printed out, and further changes can then be made if desired.

DA returned to DM with the computer analysis results. After studying them DM said, "So a_3 has the highest utility. That's interesting, but I've been thinking, and I believe we need another attribute — something to do with worker complaints. I think there will be different numbers of complaints for the three processes." After discussion DM and DA decided to use the attribute X_3 = number of worker complaints per batch of widgets. The range of this was -5 (i.e., five compliments) to 15. Using the same procedure as for X_1 and X_2 the conditional utility function and marginal probability distributions were assessed for X_3. These are shown in Figure 3. Also, the addition of the new attribute required that the scaling constants be reassessed. This was done and they were found to be $k_1 = 0.6$, $k_2 = 0.42$ and $k_3 = 0.39$.

DA entered the changes during the computer session shown in Exhibit III. The resulting output is shown in Exhibit IV. After looking this over DM remarked that he may have been overly optimistic about the number of complaints that would result from a_3. He decided that the

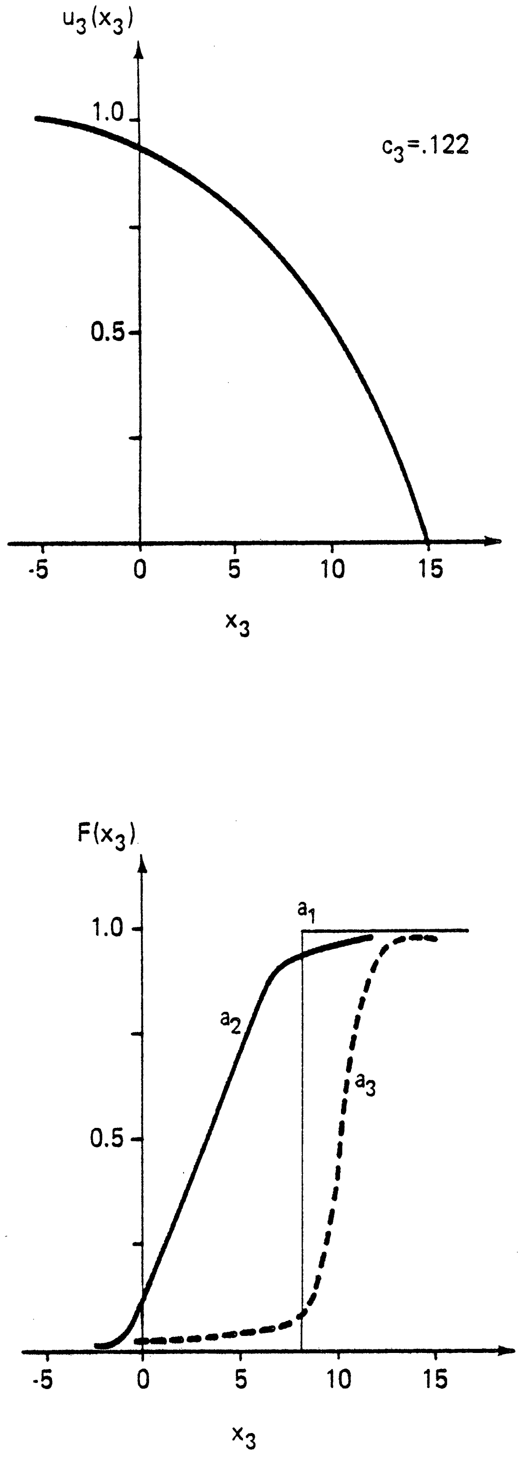


Figure 3. ADDITIONAL ATTRIBUTE X_3

EXHIBIT III

NEW ATTRIBUTE INPUT

```
#EXECUTION BEGINS
  INP       - SPECIFY FDNAME OR SEND END-OF-FILE
?WIDGET
  STORE     - SPECIFY FDNAME OR SEND END-OF-FILE
?WIDGIN
```

```
 ENTER 1 IF YOU WANT TO READ FROM FILE, ZERO IF NOT
 1
```

```
              ***** MULTIATTRIBUTE DECISION ANALYSIS *****

                         ATTRIBUTES : 2

                         ALTERNATIVES : 3

                       - 0 - 0 - 0 - 0 - 0 -


    ---- INFORMATION ABOUT UTILITY FUNCTION
                                                        R A N G E S
         ATTRIBUTE    RISK TYPE     CONSTANT        LOWEST       HIGHEST
            1             2        9.0000E-02         0.00         20.00
            2             2        1.6500E-04      5000.00      10000.00


    ---- SCALING CONSTANTS
                 1      .4900
                 2      .7000

                 *** K =      -0.5540
```

EXHIBIT III (continued)

----INFORMATION ABOUT PROBABILITIES

ALTERNATIVE 1

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	10.00	10.00	10.00
2	7500.00	7500.00	7500.00

ALTERNATIVE 2

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	1.00	4.00	14.00
2	7500.00	8500.00	9500.00

ALTERNATIVE 3

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	8.00	14.00	18.00
2	5500.00	6000.00	7000.00

---- INDIVIDUAL ATTRIBUTE EXPECTED UTILITY VALUES

ALTERNATIVE: 1

ATTRIBUTE	EXPECTED UTILITY
1	.7109
2	.6017

ALTERNATIVE: 2

ATTRIBUTE	EXPECTED UTILITY
1	.8500
2	.3833

ALTERNATIVE: 3

ATTRIBUTE	EXPECTED UTILITY
1	.4978
2	.8430

---- EXPECTED UTILITY FOR EACH ALTERNATIVE

ALTERNATIVE	EXPECTED UTILITY
1	.6883
2	.6229
3	.7543

EXHIBIT III (concluded)

```
ENTER 1 IF YOU WANT SENSITIVITY ANALYSIS, ZERO IF NOT
1

ENTER 1 IF YOU WANT TO ADD MORE ALTERNATIVES,ZERO IF NOT
0

ENTER 1 IF YOU WANT TO ADD MORE ATTRIBUTES, ZERO IF NOT
1

HOW MANY MORE ATTRIBUTES?
1

ENTER RISK TYPE, CONSTANT AND RANGES FOR NEW ATTRIBUTE: 3
2 .122 -5 15

NOW, PLEASE ENTER ALL THE SCALING CONSTANTS

ENTER SCALING CONSTANT NUMBER: 1
.42

ENTER SCALING CONSTANT NUMBER: 2
.6

ENTER SCALING CONSTANT NUMBER: 3
.39

ENTER FRACTILES FOR ALTERNATIVE: 1 AND NEW ATTRIBUTE: 3
8 8 8

ENTER FRACTILES FOR ALTERNATIVE: 2 AND NEW ATTRIBUTE: 3
-1 3 8

ENTER FRACTILES FOR ALTERNATIVE: 3 AND NEW ATTRIBUTE: 3
6 10 12
```

EXHIBIT IV

NEW ATTRIBUTE OUTPUT

```
*******************************************

          SENSITIVITY ANALYSIS

*******************************************



***** MULTIATTRIBUTE DECISION ANALYSIS *****

              ATTRIBUTES : 3

              ALTERNATIVES : 3

           - 0 - 0 - 0 - 0 - 0 -


---- INFORMATION ABOUT UTILITY FUNCTION
                                                  R A N G E S
     ATTRIBUTE    RISK TYPE     CONSTANT        LOWEST        HIGHEST
        1             2        9.0000E-02         0.00          20.00
        2             2        1.6500E-04      5000.00       10000.00
        3             2        1.2200E-01        -5.00          15.00


---- SCALING CONSTANTS
               1       .4200
               2       .6000
               3       .3900

               *** K =      -0.7064
```

EXHIBIT IV (concluded)

----INFORMATION ABOUT PROBABILITIES

ALTERNATIVE 1

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	10.00	10.00	10.00
2	7500.00	7500.00	7500.00

ALTERNATIVE 2

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	1.00	4.00	14.00
2	7500.00	8500.00	9500.00

ALTERNATIVE 3

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	8.00	14.00	18.00
2	5500.00	6000.00	7000.00

---- INDIVIDUAL ATTRIBUTE EXPECTED UTILITY VALUES

ALTERNATIVE: 1

ATTRIBUTE	EXPECTED UTILITY
1	.7109
2	.6017

ALTERNATIVE: 2

ATTRIBUTE	EXPECTED UTILITY
1	.8500
2	.3833

ALTERNATIVE: 3

ATTRIBUTE	EXPECTED UTILITY
1	.4978
2	.8430

---- EXPECTED UTILITY FOR EACH ALTERNATIVE

ALTERNATIVE	EXPECTED UTILITY
1	.6883
2	.6229
3	.7543

whole distribution for X_3 given a_3 should be moved up by one. This was done, and Exhibits V and VI show the computer input and output for this case.

After seeing the results and noting that a_3 was still the preferred alternative with all of the changes, DM concluded that he should select that alternative. DA commented that they might do a more detailed analysis with a more completely assessed utility function, however, DM felt the analysis just completed was sufficient.

2.4 Data Files

Some analysts may wish to set up or modify data files for the computer program directly rather than using the interactive assessment procedure discussed above. Copies of the data files that resulted from each of DA's two sessions on the computer are shown in Exhibit VII. Comparing this with Exhibits II and VI will show how the data is stored in the file.

3. CONCLUDING REMARKS

The programs described in this report are an intermediate option between doing hand calculations or using a more sophisticated computer program such as MUFCAP [8,13]. They do not provide some of the advanced capabilities of MUFCAP, such as hierarchical structuring of utility functions. However, the capabilities provided should be sufficient for many analyses.

EXHIBIT V

ADDITIONAL COMPLAINTS INPUT

```
ENTER 1 IF YOU WANT SENSITIVITY ANALYSIS, ZERO IF NOT
1

ENTER 1 IF YOU WANT TO ADD MORE ALTERNATIVES,ZERO IF NOT
0

ENTER 1 IF YOU WANT TO ADD MORE ATTRIBUTES, ZERO IF NOT
0

IF YOU WANT TO CHANGE SOME OF THE FOLLOWING VALUES *
RISK TYPE, CONSTANT OR RANGES, ENTER 1, ZERO IF NOT
0

IF YOU WANT TO CHANGE SOME FRACTILE VALUES ENTER 1, ZERO IF NOT
1

OK, HOW MANY CHANGES?
1

CHANGE #  1: PLEASE ENTER ALTERNATIVE #, ATTRIBUTE # AND FRACTILES
3 3 7 11 13

IF YOU WANT TO CHANGE SOME OF THE SCALING CONSTANTS
, ENTER 1, ZERO IF NOT
0
```

EXHIBIT VI

ADDITIONAL COMPLAINTS OUTPUT

****

SENSITIVITY ANALYSIS

****

***** MULTIATTRIBUTE DECISION ANALYSIS *****

ATTRIBUTES : 3

ALTERNATIVES : 3

- 0 - 0 - 0 - 0 - 0 -

----- INFORMATION ABOUT UTILITY FUNCTION

ATTRIBUTE	RISK TYPE	CONSTANT	RANGES LOWEST	RANGES HIGHEST
1	2	9.0000E-02	0.00	20.00
2	2	1.6500E-04	5000.00	10000.00
3	2	1.2200E-01	-5.00	15.00

----- SCALING CONSTANTS

1	.4200
2	.6000
3	.3900

*** K = -0.7064

EXHIBIT VI (continued)

-----INFORMATION ABOUT PROBABILITIES

ALTERNATIVE 1

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	10.00	10.00	10.00
2	7500.00	7500.00	7500.00
3	8.00	8.00	8.00

ALTERNATIVE 2

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	1.00	4.00	14.00
2	7500.00	8500.00	9500.00
3	-1.00	3.00	8.00

ALTERNATIVE 3

ATTRIBUTE	.05 FRACT	.50 FRACT	.95 FRACT
1	8.00	14.00	18.00
2	5500.00	6000.00	7000.00
3	7.00	11.00	13.00

----- INDIVIDUAL ATTRIBUTE EXPECTED UTILITY VALUES

ALTERNATIVE: 1

ATTRIBUTE	EXPECTED UTILITY
1	.7109
2	.6017
3	.6291

ALTERNATIVE: 2

ATTRIBUTE	EXPECTED UTILITY
1	.8500
2	.3833
3	.8208

ALTERNATIVE: 3

ATTRIBUTE	EXPECTED UTILITY
1	.4978
2	.8430
3	.4367

EXHIBIT VI (concluded)

```
----- EXPECTED UTILITY FOR EACH ALTERNATIVE

          ALTERNATIVE        EXPECTED UTILITY
               1                   .7277



               2                   .7295
               3                   .7335



 ENTER 1 IF YOU WANT SENSITIVITY ANALYSIS, ZERO IF NOT
 0


#EXECUTION TERMINATED
#
```

EXHIBIT VII

DATA FILES

i) **For output in Exhibit II**

```
 LIST WIDGET
>      1         2   3
>      2         2     9.0000E-02          0.00        20.00
>      3         2     1.6500E-04       5000.00     10000.00
>      4                 10.00       10.00       10.00
>      5               7500.00     7500.00     7500.00
>      6                  1.00        4.00       14.00
>      7               7500.00     8500.00     9500.00
>      8                  8.00       14.00       18.00
>      9               5500.00     6000.00     7000.00
>     10             .4900
>     11             .7000
#END OF FILE
#
```

ii) **For output in Exhibit VI**

```
 LIST WIDGIN
>      1         3   3
>      2         2     9.0000E-02          0.00        20.00
>      3         2     1.6500E-04       5000.00     10000.00
>      4         2     1.2200E-01         -5.00        15.00
>      5                 10.00       10.00       10.00
>      6               7500.00     7500.00     7500.00
>      7                  8.00        8.00        8.00
>      8                  1.00        4.00       14.00
>      9               7500.00     8500.00     9500.00
>     10                 -1.00        3.00        8.00
>     11                  8.00       14.00       18.00
>     12               5500.00     6000.00     7000.00
>     13                  7.00       11.00       13.00
>     14             .4200
>     15             .6000
>     16             .3900
#END OF FILE
#
```

REFERENCES

1. Drake, A.W., Fundamentals of Applied Probability Theory, McGraw-Hill, New York, 1967.

2. Hax, A.C., and K.M. Wiig, "The Use of Decision Analysis on Capital Investment Problems, "Sloan Management Review, pp. 19-49 (Winter, 1976).

3. Keefer, D.L., "A Decision Analysis Approach to Resource Allocation Planning Problems with Multiple Objectives," PhD dissertation, Department of Industrial and Operations Engineering, The University of Michigan, 1976. Available from University Microfilms, Ann Arbor, Michigan.

4. Keefer, D.L., and C.W. Kirkwood, "A Multiobjective Decision Analysis: Budget Planning for Product Engineering," Technical Report No. 76-5, Department of Industrial and Operations Engineering, The University of Michigan, September 1976. To appear in Operational Research Quarterly.

5. Keeney, R.L., "A Decision Analysis with Multiple Objectives: the Mexico City Airport," Bell Journal of Economics and Management Science, Vol. 4, pp. 101-117 (1973).

6. Keeney, R.L., "Multiplicative Utility Functions," Operations Research, Vol. 22, pp. 24-34 (1974).

7. Keeney, R.L., and H. Raiffa, Decisions With Multiple Objectives, Wiley, New York, 1976.

8. Keeney, R.L., and A. Sicherman, "Assessing and Analyzing Preferences Concerning Multiple Objectives: An Interactive Computer Program," Behavioral Science, Vol. 21, pp. 173-182 (1976).

9. Pearson, E.S., and J.W. Tukey, "Approximate Means and Standard Deviations Based on Distances Between Percentage Points of Frequency Curves," Biometrika, Vol. 52, pp. 533-546 (1965).

10. Perry, C., and I.D. Greig, "Estimating the Mean and Variance of Subjective Distributions in PERT and Decision Analysis," Management Science, Vol. 21, pp. 1477-1480 (1975).

11. Pratt, J.W., "Risk Aversion in the Small and in the Large," Econometrica, Vol. 32, pp. 122-136 (1964).

12. Pratt, J.W., H. Raiffa, and R. Schlaifer, "The Foundations of Decision Under Uncertainty: An Elementary Exposition," Journal of the American Statistical Association, Vol. 59, pp. 353-375 (1964).

13. Sicherman, A., "An Interactive Computer Program for Assessing and Using Multiattribute Utility Functions," Technical Report No. 111, Operations Research Center, Massachusetts Institute of Technology, June 1975.

14. Spetzler, C.S., and C.-A.S. Stael von Holstein, "Probability Encoding in Decision Analysis," Management Science, Vol. 22, pp. 340-358 (1975).

APPENDIX A

CALCULATOR PROGRAM

The calculator program described in this appendix was written for the Hewlett-Packard HP-25 calculator. A virtually identical program should run on any calculator with an automatic stack.

Constant risk aversion. The first segment of the calculator program, stored in program steps 1 to 23, calculates the certainty equivalent for any two-fork single attribute lottery if preferences are constantly risk averse and increasing in the attribute and if the risk aversion coefficient is specified. This program can also be used to find the certainty equivalent for a two-fork lottery assuming constant risk aversion and decreasing preferences. To do this reverse the signs on the attribute values for the two forks before entering them and also reverse the sign on the certainty equivalent calculated by the program.

Constant proportional risk aversion. The second segment of the program, stored in program steps 25 to 48, calculates the certainty equivalent for any two-fork single attribute lottery if preferences are constant proportionally risk averse and increasing in the attribute and if the risk aversion coefficient is specified. (For this case all attribute values must be positive.)

The equations for all three cases handled by the program are given in (7).

Program uses. The program can be used for two different tasks. First, a utility function can be found that fits an assessed certainty equivalent for a specified lottery. This is done by trial-and-error as discussed in Section 2.1. To do this the lottery is input, the type of utility function to be fit is selected and different values of the risk aversion coefficient are tried until the calculated certainty equivalent for the lottery equals the assessed one.

The second use of the program is to plot a specified utility function. To do this enter as the attribute values on the two forks of the lottery the extremes of the range over which the utility function is to be determined. Then use the calculator program to find the certainty equivalent of the lottery for different probabilities of obtaining the more desirable fork. Then, of course,

$$\begin{aligned} &u(\text{certainty equivalent}) \\ &\quad = pu\,(\text{most desirable fork}) \\ &\qquad + (1-p)u\,(\text{least desirable fork}). \end{aligned}$$

If the utility function is scaled so that u (most desirable fork) = 1 and u (least desirable fork) = 0, then u (certainty equivalent) = p. By varying p the utilities of as many points as desired can be found.

Example problems. Instructions for using the calculator program and a listing are given at the end of this appendix. To check that the program has been properly entered the following three examples may be used:

i) Constant risk aversion with increasing preferences

$$\left.\begin{array}{l} p' = 0.5 \\ x' = 18 \\ x'' = 16 \\ c\ \ = 0.1 \end{array}\right\} \quad \begin{array}{c} \text{certainty equivalent} \\ = 16.95 \end{array}$$

ii) Constant risk aversion with decreasing preferences.

$$\left.\begin{array}{l} p' = 0.5 \\ x' = 16 \\ x'' = 18 \\ c\ \ = 0.1 \end{array}\right\} \quad \begin{array}{c} \text{certainty equivalent} \\ = 17.05 \end{array}$$

iii) Constant proportional risk aversion

$$\left.\begin{array}{l} p' = 0.5 \\ x' = 18 \\ x'' = 16 \\ c\ \ = 1.70 \end{array}\right\} \quad \begin{array}{c} \text{certainty equivalent} \\ = 16.95 \end{array}$$

HP-25 Program Form

Title: Utility Assessment — Page 1 of 2

Programmer:

STEP	INSTRUCTIONS	INPUT DATA/UNITS	KEYS				OUTPUT DATA/UNITS
1	Store program						
2	Store probability	p'	STO	0			
3	Store more						
	preferred outcome	x'	STO	1			
4	Store less						
	preferred outcome	x''	STO	2			
5	Select type of						
	utility function						
	a) to select $-e^{-cx}$		GTO	0	0		
	b) to select x^{1-c}		GTO	2	5		
6	Input risk						
	aversion coefficient	c	R/S				
7	Display certainty						
	equivalent						certainty equivalent
8	If desired,						
	display c		$x \gtrless y$				c

HP-25 Program Form

Title: Utility Assessment — Page 2 of 2

Switch to PRGM mode, press [f] [PRGM], then key in the program.

DISPLAY LINE	DISPLAY CODE	KEY ENTRY	X	Y	Z	T	COMMENTS
00			c				
01	31	↑	c	c			
02	32	CHS	−c	c			
03	15 07	g e^x	e^{-c}	c			
04	23 04	STO 4	e^{-c}	c			
05	24 01	RCL 1	x'	e^{-c}	c		
06	14 03	f y^x	$e^{-cx'}$	c			
07	01	1	1	$e^{-cx'}$	c		
08	23 03	STO 3	1	$e^{-cx'}$	c		
09	22	R↓	$e^{-cx'}$	c		1	
10	24 00	RCL 0	p'	$e^{-cx'}$	c		
11	23 41 03	STO−3	p'	$e^{-cx'}$	c		
12	61	×	$p'e^{-cx'}$	c			
13	24 04	RCL 4	e^{-c}	$p'e^{-cx'}$	c		
14	24 02	RCL 2	x''	e^{-c}	$p'e^{-cx'}$	c	
15	14 03	f y^x	$e^{-cx''}$	$p'e^{-cx'}$	c	c	
16	24 03	RCL 3	p''	$e^{-cx''}$	$p'e^{-cx'}$	c	
17	61	×	$p''e^{-cx''}$	$p'e^{-cx'}$	c	c	
18	51	+	↓+	c	c	c	
19	14 07	f ln	ln [↓]	c	c	c	
20	21	x⇄y	c	ln [ ]	c	c	
21	32	CHS	−c	ln [ ]	c	c	
22	71	÷	−ln[ ]/c	c	c	c	X = CE of lottery
23	13 00	GTO 00					Y = c
24	15 74	g NOP					
25	31	↑	c	c			
26	31	↑	c	c	c		
27	01	1	1	c	c		
28	23 03	STO 3	1	c	c		
29	21	x⇄y	c	1	c		
30	41	−	1−c	c			
31	23 05	STO 5	1−c	c			
32	24 01	RCL 1	x'	1−c	c		
33	21	x⇄y	1−c	x'	c		
34	14 03	f y^x	x'^{1-c}	c			
35	24 00	RCL 0	p'	x'^{1-c}	c		
36	23 41 03	STO−3	p'	x'^{1-c}	c		
37	61	×	$p'x'^{1-c}$	c			
38	24 02	RCL 2	x''	$p'x'^{1-c}$	c		
39	24 05	RCL 5	1−c	x''	$p'x'^{1-c}$	c	
40	14 03	f y^x	x''^{1-c}	$p'x'^{1-c}$	c	c	
41	24 03	RCL 3	p''	x''^{1-c}	$p'x'^{1-c}$	c	
42	61	×	$p''x''^{1-c}$	$p'x'^{1-c}$	c	c	
43	51	+	↓+	c	c	c	
44	24 05	RCL 5	1−c	[ ]	c	c	
45	15 22	g 1/x	1/[1−c]	[ ]	c	c	
46	14 03	f y^x	$[\ ]^{1/[1-c]}$	c	c	c	X = CE of lottery
47	74	R/S					Y = c
48	13 25	GTO 25					
49							

REGISTERS

- R_0: p'
- R_1: x' (more preferred)
- R_2: x'' (less preferred)
- R_3: p'' *
- R_4: e^{-c} *
- R_5: 1−c *
- R_6:
- R_7:

* calculated by program

APPENDIX B

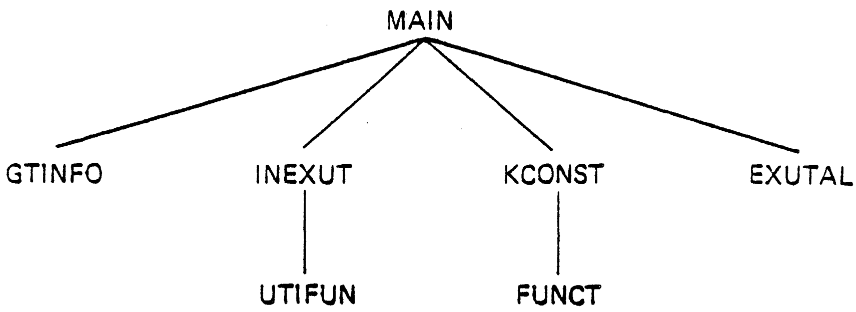
DESCRIPTION OF COMPUTER PROGRAM

This appendix describes the computer program MULAT which aids in carrying out multiobjective decision analysis. This program is interactive and written in Level F PL/I. It uses no system dependent features and should run on any computer system which supports this language. The program consists of a MAIN procedure and six subprocedures. The calling hierarchy and program organization are shown in Figure 4.

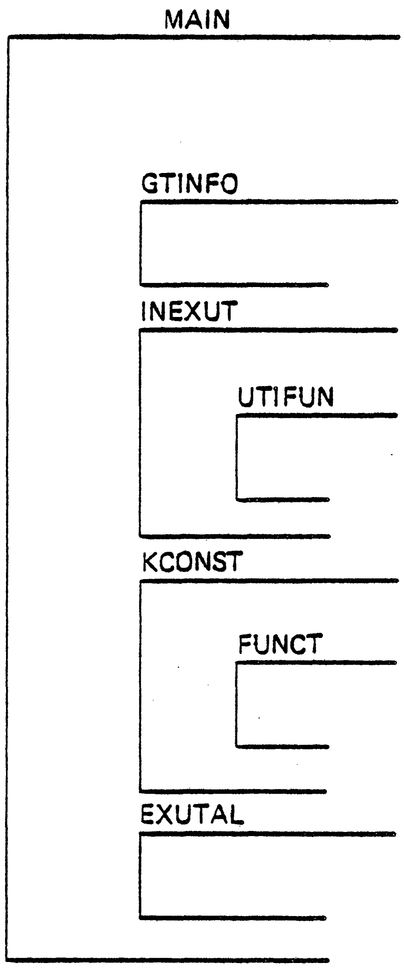
General description. The program handles decision problems with the structure shown in Figure 5. That is, single stage multiattribute problems with continuous probability distributions can be analyzed. A maximum of twenty attributes and twenty alternatives can be accomodated. The allowed types of utility functions and probability distributions are discussed in Section 1. Briefly, mutual utility independence of the attributes is assumed along with constant or constant proportional risk aversion for each attribute. In addition, for each alternative the attributes are assumed mutually probabilistically independent. The Pearson-Tukey approximation is used to calculate the required expected utilities.

As discussed in Section 2, the decision problem data can be stored in a data file for future use. Also, changes can be made to an existing problem's data in order to carry out sensitivity analyses.

The program provides error recovery for the following types of errors:



a) CALLING HEIRARCHY



b) PROGRAM ORGANIZATION

Figure 4

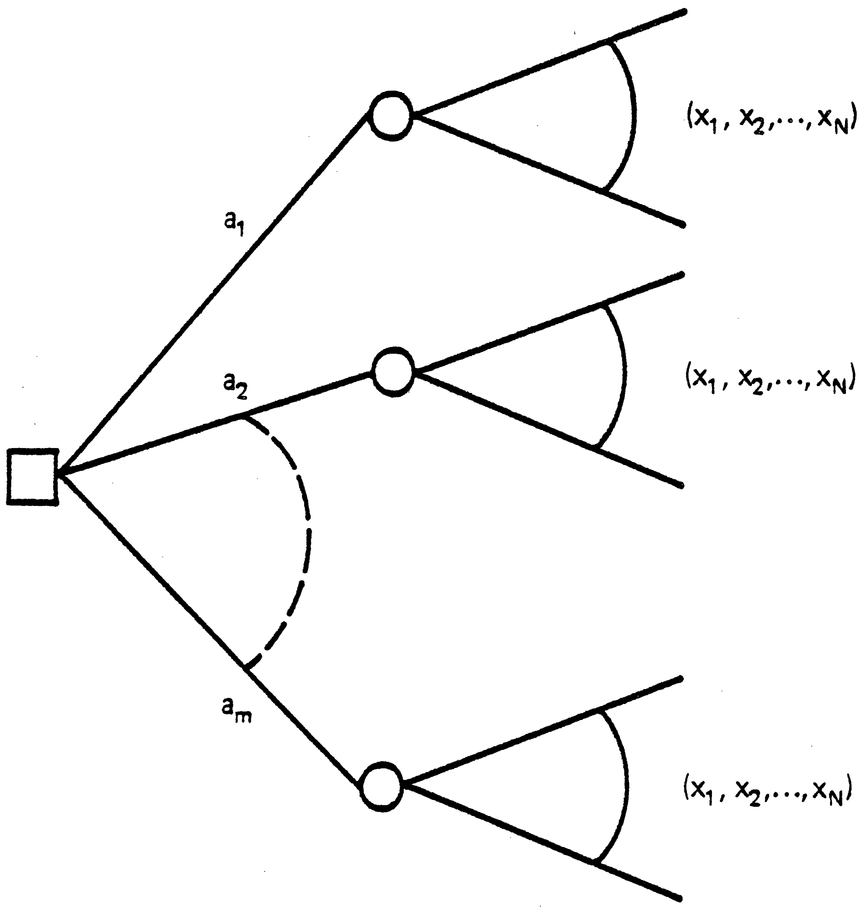


Figure 5. DECISION PROBLEM

- attribute or alternative number specified is greater than the total number previously specified
- a fractile is outside the range previously specified
- more than 500 iterations of the solution algorithm are needed to calculate the scaling constant k.

The remainder of this Appendix describes the functions of each procedure in MULAT. A listing of the program is furnished in Appendix C.

PROCEDURE NAME: MAIN

Procedure call: none

Parameters: none

Description: This is the main program for the multiattribute decision analysis.

Functions:

- Opens files.
- Calls GTINFO to ask user for problem data.
- Calls INEXUT to compute individual attribute expected utilities for each alternative.
- Calls KCONST to compute the value of scaling constant k.
- Calls EXUTAL to compute the expected utility for each alternative
- Prints out the analysis results.
- Changes data for sensitivity analysis, if desired, and repeats sequence of subprocedure calls needed to calculate expected utilities.
- Prints final problem data into a file before terminating execution.

Subprocedures called:

GTINFO, INEXUT, KCONST, EXUTAL

PROCEDURE NAME: GTINFO

Procedure call:

```
CALL GTINFO(NU_AT,NU_ALT,TYPE,C,XW,XB,XL,XM,XH,KI)
```

Input parameters: none

Output parameters:

NU_AT:	number of attributes
NU_ALT:	number of alternatives
TYPE:	risk types for attributes (vector)
C:	risk constants for attributes (vector)
XW:	lowest values of attribute ranges (vector)
XB:	highest values of attribute ranges (vector)
XL:	0.05-fractiles (two-dimensional array)
XM:	0.50-fractiles (two-dimensional array)
XH:	0.95-fractiles (two-dimensional array)
KI:	attribute scaling constants (vector)

Description: This procedure obtains input data interactively from user.

Functions:

- Asks user for source of data (either data file or terminal)
- Obtains data from specified source.

Subprocedures called: none

PROCEDURE NAME: INEXUT

Procedure call:

CALL INEXUT(NU_AT,NU_ALT,TYPE,C,XW,XB,XL,XM,XH,EXUTI)

Input parameters:

NU_AT: number of attributes

NU_ALT: number of alternatives

TYPE: risk types for attributes (vector)

C: risk constants for attributes (vector)

XW: lowest values of attribute ranges (vector)

XB: highest values of attribute ranges (vector)

XL: 0.05-fractiles (two-dimensional array)

XM: 0.50-fractiles (two-dimensional array)

XH: 0.95-fractiles (two-dimensional array)

EXUTI: expected utilities for each alternative/attribute combination (two-dimensional array)

Description: This procedure calculates the expected utility for each attribute for each alternative.

Functions:

- For each alternative
  - For each attribute
    - For each fractile
      - Call subprocedure UTIFUN to compute single attribute value.
- For each alternative
  - For each attribute

- Compute single attribute expected utility using Pearson-Tukey approximation.

Subprocedures called: UTIFUN

PROCEDURE NAME: UTIFUN

Procedure call:

UTIFUN (TY,C,X,Y1,Y2)

Input parameters:

TY: risk type

C: risk constant

X: attribute value for which utility is desired

Y1: lowest value of attribute range

Y2: highest value of attribute range

Output parameters: none

Description: This function calculates the single attribute utility for a specified value of the attribute.

Functions:

- Identifies the risk type of the utility function.
- Computes the utility for the specified attribute value.

Subprocedures called: none

PROCEDURE NAME: KCONST

Procedure call:

CALL KCONST (NU_AT,KON,KI)

Input parameters:

NU_AT: number of attributes

KI: attribute scaling constants (vector)

Output parameters:

KON: value of scaling constant k.

Description: This procedure uses the method of bisection to find the solution to the equation

$$f(k) = \prod_{n=1}^{NU\_AT} [KON*KI(n) + 1] - (KON + 1) = 0$$

Functions:

- Computes sum of scaling constants to find region where k will lie.
- Computes value of f(k) at midpoint of region
- If $|f(k)| < 10^{-5}$ at the midpoint return.
- Otherwise, shrinks the region, calculates the value of f(k) at the midpoint of the new region and repeats the test.

Subprocedures called: FUNCT

PROCEDURE NAME: FUNCT

Procedure call:

FUNCT (X,NU_AT,KI)

Input parameters:

X: attribute value for which f(X) is desired

NU_AT: number of attributes

KI: attribute scaling constants (vector)

Output parameters: none

Description: This function computes the value of

$$f(X) = \prod_{n=1}^{NU\_AT} [X*KI(n) + 1] - (X + 1)$$

Subprocedures called: none

PROCEDURE NAME: EXUTAL

Procedure call:

CALL EXUTAL (EXUTI,KON,KI,NU_AT,NU_ALT,EX_UTI_AL)

Input parameters:

EXUTI: expected utilities for each attribute/alternative combination (two-dimensional array)

KON: scaling constant k

KI: attribute scaling constants (vector)

NU_AT: number of attributes

NU_ALT: number of alternatives

EX_UTI_AL: expected utilities for alternatives (vector)

Description: This procedure computes the expected utility for each alternative.

Subprocedures called: none

APPENDIX C

COMPUTER PROGRAM LISTING

```
%PROCESS ('NOATR,NOXREF');
 MULAT: PROCEDURE OPTIONS(MAIN);
 DCL  (K,J,NU_AT,NU_ALT)        FIXED BIN(31);
 DCL  (C(20),XW(20),XB(20))      FLOAT DECIMAL;
 DCL  (XL(20,20),XM(20,20),XH(20,20),KI(20))          FLOAT DECIMAL;
 DCL  (EXUTI(20,20),EX_UTI_AL(20))      FLOAT DECIMAL ;
 DCL KON   FLOAT DECIMAL ;
 DCL TYPE(3)    FIXED BIN(31);
 DCL  (SW,SW2,SW3,SW4,SW5,SW6)      FIXED BIN(31);
 DCL  (NEW_NU_AT,N,NEW,NU_CHAN,ATTR_NU,NU_CHAN_Q,ALT_NU,AT_NU,
      NU_CHAN_K,NU_CON)        FIXED BIN(31);
 DCL NEW_NU_ALT                         FIXED BIN(31);
 DCL L(20)                    LABEL;
 DCL Z                        FIXED BIN(31);
 /* ************************************************** */
 OPEN FILE(INP) INPUT;
 OPEN FILE(STORE) OUTPUT;
 OPEN FILE(DATA) TITLE('SCARDS') INPUT;
  ON CONVERSION BEGIN;
     PUT SKIP EDIT('*** ERROR: INVALID DATA, TRY AGAIN') (A);
     PUT SKIP;
     GO TO L(Z);
    END;

 /* ASK FOR INFORMATION */

 CALL GTINFO(NU_AT,NU_ALT,TYPE,C,XW,XB,XL,XM,XH,KI);

/* COMPUTE THE INDIVIDUAL EXPECTED UTILITY VALUES */

 INIPRO:
 CALL INEXUT(NU_AT,NU_ALT,TYPE,C,XW,XB,XL,XM,XH,EXUTI);

 /*COMPUTE THE SCALING CONSTANT K */

 CALL KCONST(NU_AT,KON,KI);

 /* COMPUTE THE EXPECTED UTILITY FOR EACH ALTERNATIVE */

 CALL EXUTAL(EXUTI,KON,KI,NU_AT,NU_ALT,EX_UTI_AL);







/* PRINT OUT THE ANALYSIS RESULTS SUMMARY */

 PUT SKIP(5)  EDIT('***** MULTIATTRIBUTE DECISION ANALYSIS *****')
                   (X(13),A);
 PUT SKIP(2)  EDIT('ATTRIBUTES :',NU_AT)(X(27),A,F(2));
 PUT SKIP(2)  EDIT('ALTERNATIVES :',NU_ALT)(X(27),A,F(2));
 PUT SKIP(2)  EDIT('- 0 - 0 - 0 - 0 - 0 -')(X(24),A);
 PUT SKIP(4)  EDIT('---- INFORMATION ABOUT UTILITY FUNCTION ')
                 (X(5),A);
 PUT SKIP EDIT('R A N G E S')(X(53),A);
 PUT SKIP EDIT('ATTRIBUTE','RISK TYPE','CONSTANT','LOWEST','HIGHEST')
                 (X(10),A,X(3),A,X(4),A,X(6),A,X(7),A);
 DO J=1 TO NU_AT;
  PUT SKIP EDIT(J,TYPE(J),C(J),XW(J),XB(J))
```

```
                    (X(12),F(2),X(11),F(2),X(4),E(13,4),X(3),F(10,2),
                       X(2),F(10,2));
         END;
        PUT SKIP(4) EDIT('---- SCALING CONSTANTS') (X(5),A);
         DO J=1 TO NU_AT;
          PUT SKIP EDIT(J,KI(J)) (X(18),F(2),X(5),F(5,4));
         END;
        PUT SKIP(2) EDIT('*** K = ',KON) (X(19),A,F(10,4));
        PUT SKIP(4) EDIT('----INFORMATION ABOUT PROBABILITIES') (X(5),A);
         DO K=1 TO NU_ALT;
            PUT SKIP(2) EDIT('ALTERNATIVE',K) (X(5),A,F(2));
            PUT SKIP EDIT('ATTRIBUTE','.05 FRACT','.50 FRACT','.95 FRACT')
                         (X(8),A,X(7),A,X(7),A,X(7),A);
            DO J=1 TO NU_AT;
               PUT SKIP EDIT(J,XL(K,J),XM(K,J),XH(K,J)) (X(12),F(2),
                             (3) (X(7),F(10,2)));
            END;
         END;
        PUT SKIP(4) EDIT('---- INDIVIDUAL ATTRIBUTE EXPECTED UTILITY VALUES')
                    (X(5),A);
         DO K=1 TO NU_ALT;
          PUT SKIP(2) EDIT('ALTERNATIVE:',K) (X(29),A,X(1),F(2));
          PUT SKIP(2) EDIT('ATTRIBUTE','EXPECTED UTILITY')
                           (X(20),A,X(10),A);
          DO J=1 TO NU_AT;
           PUT SKIP EDIT(J,EXUTI(K,J)) (X(23),F(2),X(18),F(5,4));
          END;
         END;
        PUT SKIP(4) EDIT('---- EXPECTED UTILITY FOR EACH ALTERNATIVE')
                         (X(5),A);
        PUT SKIP(2) EDIT('ALTERNATIVE','EXPECTED UTILITY') (X(16),A,X(6),A);
         DO K=1 TO NU_ALT;
          PUT SKIP EDIT(K,EX_UTI_AL(K)) (X(19),F(2),X(15),F(5,4));
         END;






        /* ************************************************************* */
        /* SENSITIVITY ANALYSIS */

       PUT SKIP(10) EDIT('ENTER 1 IF YOU WANT SENSITIVITY ANALYSIS, ZERO',
                         ' IF NOT') (A,A);
       PUT SKIP;
        Z=1;
        L(1):  GET LIST(SW);
         IF SW=0 THEN GO TO ENDPROC;



        /* ASK FOR CHANGES IN NUMBER OF ALTERNATIVES */

        PUT SKIP(2) EDIT('ENTER 1 IF YOU WANT TO ADD MORE ALTERNATIVES,') (A);
        PUT EDIT('ZERO IF NOT') (A);
        PUT SKIP;
        GET LIST(SW6);
        IF SW6=1
          THEN DO;
```

```
          PUT SKIP(2) EDIT('HOW MANY MORE ALTERNATIVES ?') (A);
          PUT SKIP;
          GET LIST(NEW_NU_ALT);
          /* ASK FOR NEW FRACTILES */
          DO N=1 TO NEW_NU_ALT;
            NEW=N+NU_ALT;
            DO J=1 TO NU_AT;
              PUT SKIP(2) EDIT('ENTER FRACTILES FOR NEW ALTERNATIVE:
                       NEW,' AND ATTRIBUTE: ',J) (A,F(2),A,F(2));
              PUT SKIP;
              VER9: GET LIST(XL(NEW,J),XM(NEW,J),XH(NEW,J));
              IF (XL(NEW,J)<XW(J)|XH(NEW,J)>XB(J))
                THEN DO;
                  CALL SYSERR('*** ERROR: FRACTILE OUT OF RANGE, TRY',
                              ' AGAIN');
                  GO TO VER9;
                END;
            END;
          END;
          NU_ALT=NU_ALT+NEW_NU_ALT;
          GO TO INIPRO;
        END;



     /* ASK FOR CHANGES IN NUMBER OF ATTRIBUTES */

     PUT SKIP(2) EDIT('ENTER 1 IF YOU WANT TO ADD MORE ATTRIBUTES, ZERO',
                      ' IF NOT') (A,A);
     PUT SKIP ;
      Z=2;
      L(2): GET LIST(SW2);
       IF SW2=1
         THEN DO;
           PUT SKIP(2) EDIT('HOW MANY MORE ATTRIBUTES?') (A);
           PUT SKIP;
      Z=3;
           L(3): GET LIST(NEW_NU_AT);
             DO N=1 TO NEW_NU_AT;
               NEW=N+NU_AT;
               /* ASK FOR NEW RISK CONDITIONS */
               PUT SKIP(2) EDIT('ENTER RISK TYPE, CONSTANT AND RANGES',
                        ' FOR NEW ATTRIBUTE:',NEW) (A,A,F(2));
               PUT SKIP;
       Z=4;
               L(4): GET LIST(TYPE(NEW),C(NEW),XW(NEW),XB(NEW));
             END;
          /* ASK FOR SCALING CONSTANTS */
          PUT SKIP(2) EDIT('NOW, PLEASE ENTER ALL THE SCALING CONSTANTS')
                           (A);
          PUT SKIP;
          DO N=1 TO (NEW_NU_AT+NU_AT);
            PUT SKIP(2) EDIT('ENTER SCALING CONSTANT NUMBER:',N)
                             (A,F(2));
            PUT SKIP;
       Z=6;
            L(6): GET LIST(KI(N));
          END;
          /* ASK FOR NEW FRACTILES */
          DO K=1 TO NU_ALT;
```

```
            DO N=1 TO NEW_NU_AT;
              NEW=N+NU_AT;
              PUT SKIP(2) EDIT('ENTER FRACTILES FOR ALTERNATIVE:',K,
                         ' AND NEW ATTRIBUTE:',NEW) (A,F(2),A,F(2));
              PUT SKIP;
        Z=5;
        L(5): ;
            VER3: GET LIST(XL(K,NEW),XM(K,NEW),XH(K,NEW));
          IF (XL(K,NEW)<XW(NEW)|XH(K,NEW)>XB(NEW))
            THEN DO;
              CALL SYSERR('*** ERROR: FRACTILE OUT OF RANGE, TRY AGAIN');
                GO TO VER3;
            END;
             END;
           END;
         /* RETURN TO THE MAIN PROCEDURE */
         NU_AT=NU_AT+NEW_NU_AT;
         GO TO HEAD;
        END;



      /* CHANGES IN RISK CONDITIONS AND RANGES */

       PUT SKIP(2) EDIT('IF YOU WANT TO CHANGE SOME OF THE FOLLOWING ',
              'VALUES * ') (A,A);
       PUT  SKIP;
       PUT EDIT('RISK TYPE, CONSTANT OR RANGES, ENTER 1, ZERO IF NOT') (A);
       PUT SKIP;
        Z=7;
        L(7): GET LIST(SW3);
         IF SW3=1
           THEN DO;
             PUT SKIP(2) EDIT('OK, FOR HOW MANY ATTRIBUTES?') (A);
             PUT SKIP;
        Z=8;
             L(8): GET LIST(NU_CHAN);
               DO J=1 TO NU_CHAN;
                 PUT SKIP(2) EDIT('CHANGE # ',J,': PLEASE ENTER THE ',
                     ' ATTRIBUTE NUMBER, RISK TYPE, CONSTANT ',
                     'AND RANGES') (A,F(1),A,A,A);
                 PUT SKIP;
        Z=9;
        L(9): ;
             VER8: GET LIST(ATTR_NU,TYPE(ATTR_NU),C(ATTR_NU),XW(ATTR_NU),
                         XB(ATTR_NU));
               IF (ATTR_NU>NU_AT)
                 THEN DO;
                   CALL SYSERR('*** ERROR: ATTRIBUTE OUT OF RANGE');
                   GO TO VER8;
                 END;
               END;
           END;



       /* ASK FOR CHANGES IN FRACTILES */

       PUT SKIP(2) EDIT('IF YOU WANT TO CHANGE SOME FRACTILE VALUES',
                    ' ENTER 1, ZERO IF NOT') (A,A);
```

```
PUT SKIP;
 Z=10;
 L(10): GET LIST(SW4);
  IF SW4=1
    THEN DO;
      PUT SKIP(2) EDIT('OK, HOW MANY CHANGES?') (A);
      PUT SKIP;
 Z=11;
      L(11): GET LIST(NU_CHAN_Q);
        DO J=1 TO NU_CHAN_Q;
          PUT SKIP(2) EDIT('CHANGE # ',J,': PLEASE ENTER',
               ' ALTERNATIVE #, ATTRIBUTE # AND FRACTILES')
               (A,F(2),A,A);
          PUT SKIP;
 Z=12;
 L(12): ;
         VER6: GET LIST(ALT_NU,AT_NU,XL(ALT_NU,AT_NU),XM(ALT_NU,
                     AT_NU),XH(ALT_NU,AT_NU));
        IF (ALT_NU>NU_ALT|AT_NU>NU_AT)
          THEN DO;
            CALL SYSERR('*** ERROR:ALTERNATIVE NUMBER OR ATTRIBUTE'
                ' NUMBER IS OUT OF RANGE, TRY AGAIN');
              GO TO VER6;
          END;
        IF (XL(ALT_NU,AT_NU)<XW(AT_NU)|XH(ALT_NU,AT_NU)>XB(AT_NU))
          THEN DO;
            CALL SYSERR('*** ERROR: FRACTILE OUT OF RANGE, TRY ',
                'AGAIN');
            GO TO VER6;
          END;
        END;
  END;



/* ASK FOR CHANGES IN SCALING CONSTANTS */

PUT SKIP(2) EDIT('IF YOU WANT TO CHANGE SOME OF THE SCALING ',
       'CONSTANTS') (A);
 PUT SKIP;
 PUT EDIT(', ENTER 1, ZERO IF NOT') (A);
PUT SKIP;
 Z=13;
 L(13): GET LIST(SW5);
  IF SW5=1
    THEN DO;
      PUT SKIP(2) EDIT('OK, HOW MANY CHANGES?') (A);
      PUT SKIP;
      GET LIST(NU_CHAN_K);
        DO J=1 TO NU_CHAN_K;
          PUT SKIP(2) EDIT('CHANGE # ',J,' ENTER NUMBER OF THE ',
                     'SCALING CONSTANT AND THE NEW VALUE')
                      (A,F(2),A,A);
             PUT SKIP;
 Z=14;
          L(14): GET LIST(NU_CON,KI(NU_CON));
        END;
  END;
```

```
/* RETURN TO THE MAIN PROCEDURE */

HEAD:
PUT SKIP(5) EDIT('*********************************************')
                    (X(13),A);
PUT SKIP;
PUT SKIP(2) EDIT('   SENSITIVITY ANALYSIS   ') (X(19),A);
PUT SKIP;
PUT SKIP(2) EDIT('*********************************************')
                   (X(13),A);
 PUT SKIP;
GO TO INIPRO;





/* SUBROUTINES SECTION   */

GTINFO: PROC(NU_AT,NU_ALT,TYPE,C,XW,XB,XL,XM,XH,KI);
DCL (NU_AT,NU_ALT,TYPE(3))              FIXED BIN(31);
DCL (C(20),XW(20),XB(20),XL(20,20),XM(20,20),XH(20,20),KI(20))
                                        FLOAT DECIMAL;
DCL (K,J)                               FIXED BIN(31);
DCL Z                               FIXED BIN(31);
DCL L(20)                           LABEL;
/* ************************************************************* */
/* ASK IF THE OPTION OF READING FROM FILE SHOULD BE USED */
 ON CONV BEGIN;
   PUT SKIP EDIT('*** ERROR: INVALID DATA, TRY AGAIN') (A);
   PUT SKIP;
   GO TO L(Z);
  END;
ON ENDFILE (INP) GO TO ALLDONE;
PUT SKIP EDIT('ENTER 1 IF YOU WANT TO READ FROM FILE, ZERO IF NOT')
              (A);
PUT SKIP;
 Z=15;
L(15): GET LIST(S);
  IF S=1
    THEN DO;
      GET FILE(INP) LIST(NU_AT,NU_ALT);
        DO J=1 TO NU_AT;
          GET FILE(INP) LIST(TYPE(J),C(J),XW(J),XB(J));
        END;
        DO K=1 TO NU_ALT;
          DO J=1 TO NU_AT;
            GET FILE(INP) LIST(XL(K,J),XM(K,J),XH(K,J));
          END;
        END;
        DO J=1 TO NU_AT;
          GET FILE(INP) LIST(KI(J));
        END;
      GO TO ALLDONE;
    END;
/* ASK FOR NUMBER OF ATTRIBUTES AND ALTERNATIVES */
PUT SKIP(5) EDIT('ENTER NUMBER OF ATTRIBUTES AND ALTERNATIVES')
                   (A);
PUT SKIP;
```

```
     Z=16;
    L(16): GET LIST(NU_AT,NU_ALT);
    /* ASK FOR RISK TYPE, CONSTANTS AND RANGES */
      DO J=1 TO NU_AT;
      PUT SKIP(2) EDIT('ENTER RISK TYPE, CONSTANT AND RANGES FOR',
                       ' ATTRIBUTE:',J) (A,A,F(2));
      PUT SKIP;
     Z=17;
       L(17): GET LIST(TYPE(J),C(J),XW(J),XB(J));
      END;
    /* ASK FOR SCALING CONSTANTS */
      DO J=1 TO NU_AT;
        PUT SKIP(2) EDIT('ENTER SCALING CONSTANT NUMBER:',J) (A,F(2));
         PUT SKIP;
     Z=18;
         L(18): GET LIST(KI(J));
      END;
    /* ASK FOR FRACTILES */
      DO K=1 TO NU_ALT;
        DO J=1 TO NU_AT;
          PUT SKIP(2) EDIT('ENTER FRACTILES FOR ALTERNATIVE:',K,' AND',
                           ' ATTRIBUTE:',J) (A,F(2),A,A,F(2));
        PUT SKIP;
     Z=19;
     L(19): ;
            VERIF: GET LIST(XL(K,J),XM(K,J),XH(K,J));
            IF (XL(K,J)<XW(J)|XH(K,J)>XB(J))
               THEN DO;
                 CALL SYSERR('*** ERROR: FRACTILE OUT OF RANGE, TRY AGAIN'
                  GO TO VERIF;
               END;
            END;
      END;
    ALLDONE:
   END GTINFO;




   INEXUT: PROC(NU_AT,NU_ALT,TYPE,C,XW,XB,XL,XM,XH,EXUTI);
    DCL (NU_AT,NU_ALT)     FIXED BIN(31);
    DCL (C(*),XW(*),XB(*))       FLOAT DECIMAL;
    DCL TYPE(*)        FIXED BIN(31);
   DCL EXUTI(20,20)              FLOAT DECIMAL;
    DCL (XL(20,20),XM(20,20),XH(20,20))       FLOAT DECIMAL;
   DCL UTIL (20,20,3)            FLOAT DECIMAL;
    DCL (K,J)                    FIXED BIN(31);
    DCL UTIFUN ENTRY(FIXED BIN(31),FLOAT DECIMAL,FLOAT DECIMAL,
                  FLOAT DECIMAL,FLOAT DECIMAL);
    DCL (YL,YM,YH,Y1,Y2,C1)                  FLOAT DECIMAL;
    DCL TY       FIXED BIN(31);
    /* ************************************************* */
    /* COMPUTE THE EXPECTED UTILITY VALUE FOR EACH ATR FOR EACH ALT */
    /* COMPUTE UTILITIES */
      DO K=1 TO NU_ALT ;
         DO J=1 TO NU_AT;
           TY=TYPE(J);
           C1=C(J);
           YL=XL(K,J);
```

```
              YM=XM(K,J);
              YH=XH(K,J);
              Y1=XW(J);
              Y2=XB(J);
               UTIL(K,J,1) = UTIFUN(TY,C1,YL,Y1,Y2);
               UTIL(K,J,2)  = UTIFUN(TY,C1,YM,Y1,Y2);
               UTIL(K,J,3)  = UTIFUN(TY,C1,YH,Y1,Y2);
           END;
        END;
    /* USE PEARSON-TUKEY FOR EXPECTED UTILITY VALUES */
      DO K=1 TO NU_ALT;
         DO J=1 TO NU_AT;
            EXUTI(K,J)=.63*UTIL(K,J,2)+.185*(UTIL(K,J,1)+UTIL(K,J,3));
         END;
      END;



   UTIFUN: PROC(TY,C,X,Y1,Y2)   RETURNS(FLOAT DECIMAL);
    DCL X                        FLOAT DECIMAL;
    DCL (Y1,Y2)                             FLOAT DECIMAL;
    DCL C                          FLOAT DECIMAL;
    DCL TY                       FIXED BIN(31);
    DCL (A,B,UT)                      FLOAT DECIMAL;
    /* ********************************************************** */
    /* SELECT AND COMPUTE THE APPROPIATE UTILITY FUNCTION */
      IF TY=1
         THEN IF C>0
           THEN DO;
                A=(-EXP(-C*X))-(-EXP(-C*Y1));
                B=(-EXP(-C*Y2))-(-EXP(-C*Y1));
                UT=A/B;
                RETURN(UT);
              END;
         ELSE IF C<0
                THEN DO;
                       A=(EXP(C*X))-(EXP(C*Y1));
                       B=(EXP(C*Y2))-(EXP(C*Y1));
                       UT=A/B;
                       RETURN(UT);
                     END;
                ELSE
                     DO;
                       UT=(X-Y1)/(Y2-Y1);
                       RETURN(UT);
                     END;
       IF TY=2
         THEN IF C>0
           THEN DO;
                A=(-EXP(C*X))-(-EXP(C*Y2));
                B=(-EXP(C*Y1))-(-EXP(C*Y2));
                UT=A/B;
                RETURN(UT);
              END;
           ELSE IF C<0
        THEN DO;
                A=(EXP(C*X))-(EXP(C*Y2));
                B=(EXP(C*Y1))-(EXP(C*Y2));
                UT =A/B;
                RETURN(UT);
```

```
                END;
              ELSE DO;
                  UT= (-X+Y2) / (-Y1+Y2) ;
                  RETURN (UT) ;
                END;
        IF TY=3
          THEN IF C>1
            THEN DO;
                  A= (-1/X ** (C-1) ) - (-1/Y1** (C-1) ) ;
                  B= (-1/Y2** (C-1) ) - (-1/Y1** (C-1) ) ;
                  UT=A/B;
                  RETURN (UT) ;
                END;
           ELSE IF C<1
             THEN DO;
                  A= (X** (1-C) ) - (Y1** (1-C) ) ;
                  B= (Y2** (1-C) ) - (Y1** (1-C) ) ;
                  UT=A/B;
                  RETURN (UT) ;
                END;
             ELSE IF C=1
               THEN DO;
                  A=LOG (X) -LOG (Y1) ;
                  B=LOG (Y2) -LOG (Y1) ;
                  UT=A/B;
                  RETURN (UT) ;
                END;
                ELSE IF C=0
                  THEN DO;
                    UT= (X-Y1) / (Y2-Y1) ;
                    RETURN (UT) ;
                  END;
    END UTIFUN;
    END INEXUT;





    KCONST: PROC (NU_AT,KON,KI) ;
     DCL  (I,J,NU_AT)                       FIXED BIN (31) ;
     DCL  KI (20)                           FLOAT DECIMAL;
     DCL  (F,XA)                      FLOAT DECIMAL;
     DCL FUNCT ENTRY (FLOAT DECIMAL,FIXED BIN (31) , (*)  FLOAT DECIMAL) ;
     DCL X1                FLOAT DECIMAL ;
     DCL X2                FLOAT DECIMAL ;
     DCL DELTA             FLOAT DECIMAL INIT(.00001) ;
     DCL KON        FLOAT DECIMAL;
     /* ************************************************************ */

     /* COMPUTE THE CONSTANT K BY BINARY SEARCH */

     /* CALCULATE SUM OF THE CONSTANTS */
     SUM=0;
       DO J=1 TO NU_AT;
         SUM=SUM+KI (J) ;
       END;

     I = 0 ;
     /* SUM OF THE CONSTANTS IS LESS THAN ONE */
```

```
IF SUM<1
  THEN DO;
    X1=0;
    X2=15;
    INITSTP:  F=FUNCT(X2,NU_AT,KI);
    IF F<0
      THEN DO;
        X2=X2+1;
       IF X2>300 THEN GO TO NOTFOUND;
        GO TO INITSTP;
      END;
 /* FIRST MIDDLE POINT */
      XA=(X2+X1)/2 ;
      LOOP:  F=FUNCT(XA,NU_AT,KI);
      IF ABS(F) < DELTA
        THEN DO;
          KON=XA;
          GO TO END_PROC;
         END;
       IF I>500 THEN GO TO NOTFOUND;
      I=I+1;
      IF F>0
        THEN DO;
          X2=XA;
          XA=(X2+X1)/2;
          GO TO LOOP;
        END;
        ELSE DO;
          X1=XA;
          XA=(X2+X1)/2;
         GO TO LOOP;
        END;
  END;
/* SUM OF THE CONSTANTS IS GREATER THAN ONE */
IF SUM>1
  THEN DO;
    X1=-1;
    X2=0;
    XA=(X2+X1)/2;
    LOOP2:  F=FUNCT(XA,NU_AT,KI);
    IF ABS(F)< DELTA
      THEN DO;
        KON=XA;
        GO TO END_PROC;
      END;
    IF I>500 THEN GO TO NOTFOUND;
    I=I+1;
    IF F>0
      THEN DO;
        X1=XA;
        XA=(X2+X1)/2;
        GO TO LOOP2;
      END;
      ELSE DO;
        X2=XA;
        XA=(X2+X1)/2;
        GO TO LOOP2;
      END;
```

```
          END;
       /* SUM OF THE CONSTANTS IS EQUAL TO ONE */
       IF SUM=1
         THEN DO;
            KON=0;
            GO TO END_PROC;
          END;
      FUNCT: PROC(X,NU_AT,KI)      RETURNS(FLOAT DECIMAL);
       DCL X                   FLOAT DECIMAL;
       DCL (J,NU_AT)           FIXED BIN(31);
      DCL KI(*)                FLOAT DECIMAL;
       DCL (PROD,FUN)          FLOAT DECIMAL;
       /* ********************************************************* */
       /* COMPUTE THE VALUE OF THE FUNCTION */
       PROD = 1;
         DO J=1 TO NU_AT;
            PROD=PROD*(X*KI(J)+1);
          END;
       FUN=PROD-(X+1);
       RETURN(FUN);
      END FUNCT;
       NOTFOUND:  PUT SKIP LIST('K VALUE NOT FOUND');
       END_PROC:
      END KCONST;
      EXUTAL: PROC(EXUTI,KON,KI,NU_AT,NU_ALT,EX_UTI_AL);
       DCL KI(*)                   FLOAT DECIMAL;
       DCL EXUTI(*,*)                       FLOAT DECIMAL;
       DCL (KON,PROD)                       FLOAT DECIMAL;
       DCL S                  FLOAT DECIMAL;
       DCL (K,J,NU_AT,NU_ALT)               FIXED BIN(31);
       DCL EX_UTI_AL(20)                    FLOAT DECIMAL;
       /* ********************************************************** */
       /* ADDITIVE CASE */
       IF KON=0
         THEN DO;
            DO K=1 TO NU_ALT;
               S=0;
               DO J=1 TO NU_AT;
                  S=S+KI(J)*EXUTI(K,J);
               END;
               EX_UTI_AL(K)=S;
            END;
            GO TO DONE;
          END;
       /* MULTIPLICATIVE CASE */
       /* COMPUTES EXPECTED UTILITIES FOR EACH ALTERNATIVE */
```

```
          DO K=1 TO NU_ALT;
            PROD=1;
            DO J=1 TO NU_AT;
              PROD=PROD*(KON*KI(J)*EXUTI(K,J)+1);
            END;
           EX_UTI_AL(K)=(PROD-1)/KON;
         END;
        DONE:
       END EXUTAL;





         SYSERR: PROC(MES,MES2);
          DCL MES           CHAR(*);
         DCL MES2           CHAR(*);
        /* ********************************************* */
          PUT SKIP EDIT(MES,MES2) (A,A);
          PUT SKIP;
         END SYSERR;





        /* FINAL PROCESS BEFORE FINISH, STORE DATA IN FILE */
       ENDPROC:
         PUT FILE(STORE) EDIT(NU_AT,NU_ALT) (X(1),F(2),X(1),F(2));
         DO J=1 TO NU_AT;
           PUT SKIP FILE(STORE) EDIT(TYPE(J),C(J),XW(J),XB(J))
                    (X(1),F(1),X(1),E(13,4),X(1),F(10,2),X(1),F(10,2));
         END;
         DO K=1 TO NU_ALT;
           DO J=1 TO NU_AT;
             PUT SKIP FILE(STORE) EDIT(XL(K,J),XM(K,J),XH(K,J))
                    ((3)(X(1),F(10,2)));
           END;
         END;
         DO J=1 TO NU_AT;
           PUT SKIP FILE(STORE) EDIT(KI(J),'   ') (X(3),F(5,4),A);
             PUT SKIP;
         END;
       CLOSE FILE(STORE);
        CLOSE FILE(INP);
        END MULAT;
END OF FILE
```