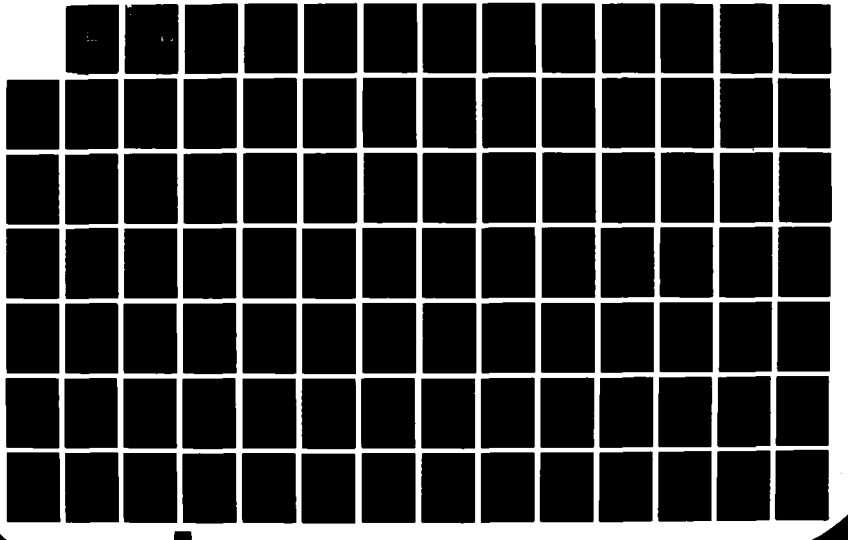
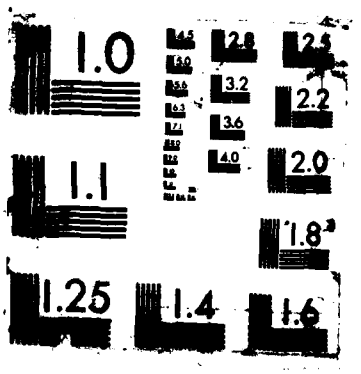
AD-A187 276 DEVELOPMENT OF A GRAPHICS BASED TWO-ATTRIBUTE UTILITY ASSESSMENT PROGRAM W.. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH E H WANNER DEC 86 1/3

UNCLASSIFIED AFIT/CI/NR-87-73T F/G 12/4 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/CI/NR 87-73T
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): Development Of A Graphics Based Two-Attribute Utility Assessment Program With Application To Mission Planning
5. TYPE OF REPORT & PERIOD COVERED: THESIS/DISSERTATION
6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Eleonore H. Wanner
8. CONTRACT OR GRANT NUMBER(s):

9. PERFORMING ORGANIZATION NAME AND ADDRESS: AFIT STUDENT AT: Rensselaer Polytechnic Institute
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:

11. CONTROLLING OFFICE NAME AND ADDRESS: AFIT/NR, WPAFB OH 45433-6583
12. REPORT DATE: December 1986
13. NUMBER OF PAGES: 183

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
15. SECURITY CLASS. (of this report): UNCLASSIFIED
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

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-1

LYNN E. WOLAVER 7 Aug 87
Dean for Research and Professional Development
AFIT/NR

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DEVELOPMENT OF A GRAPHICS BASED TWO-ATTRIBUTE UTILITY ASSESSMENT PROGRAM WITH APPLICATION TO MISSION PLANNING

by

Eleonore H. Wanner

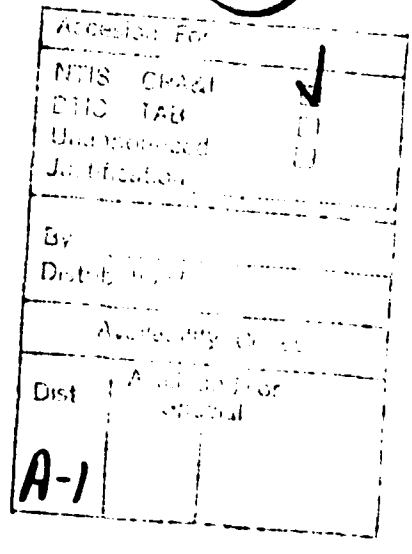
A Project Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE



Approved:

James M. Tien

Dr. James M. Tien
Thesis Advisor

Rensselaer Polytechnic Institute
Troy, New York

December 1986

CONTENTS

LIST OF FIGURES

FOREWORD

This project was suggested by Dr. James M. Tien, Acting Chairman, ECSE Dept, and I would like to express my appreciation for his suggestion. If not for him, I would never have learned so much in so short a time. His criticisms and support as my advisor have been invaluable.

Sincere thanks is also due to the people at Rome Air Development Center (RADC) who helped me in both research and programming efforts. I would particularly like to thank Mr. Yale Smith, Lt Russ Gilbertson, and Capt Peter Priest of the Decision Aids Section; Mr. Ron Blackall and Mr. Zen Pryk of the Surveillance Division; Lt Greg Bandeau of the Accounting Division; and the entire Technical Library staff.

I am grateful to Major Rand Case and Major Robert Stan for agreeing to participate in my endeavors to assess their utilities; and finally, I owe my husband volumes of thanks for his support and help, not only in the preparation of this manuscript, but throughout the school program.

ABSTRACT

A multi-attribute utility assessment program is developed which can handle consequences composed of two attributes. It handles four particular cases of utility independence properties: additive independence, mutual utility independence, utility independence in one direction, and no independence properties. A two-attribute consequence space was chosen particularly for the ability to represent the multi-attribute utility function graphically in three dimensions, from one of eight possible viewpoints. The utilities of two Air Force mission planners were assessed to determine the feasibility of using probability of arrival and probability of target destruction as factors for mission success. Based on the results, it appears as though multi-attribute utility theory would be a definite help to Air Force mission planning.

1. INTRODUCTION

Utility theory is a relatively new methodology for decision making. Relatively new, that is, when compared to probability theory from which it gained its foundation. The basic tenets of utility theory have been around for some fifty years, and it has been applied to a number of various decision making problems. In many cases, it has been grouped with other techniques for decision making under the general heading of decision analysis.

Utility theory provides the means for characterizing a decision maker's preferences over the range of some attribute. Usually these preference values are represented on a scale from 0 to 1. The ideas behind utility theory have been extended to decision making problems in which more than one attribute is used to characterize possible alternatives, and this extension is termed multi-attribute utility theory.

Stated simply, multi-attribute utility theory attempts to produce a map of a decision maker's preference values (utility) for various alternatives of a problem in a systematic fashion. These alternatives are generally represented by more than one attribute (if only one attribute is used, simple utility theory, and not multi-attribute utility theory, may be used). This map, or utility function can then help the decision maker in

determining the utility of other potential outcomes of the problem.

It is not an easy task for a decision maker to make choices between alternatives characterized by many attributes, or when the attribute values vary only slightly between alternatives. However, depending on the utility independence properties existing between attributes, it may be possible to decompose the problem into the assessment of utility functions for individual attribute values, which can later be combined in an appropriate fashion. After combining the parts, the overall utility function for different possible alternatives can be used to choose the best possible outcome by maximizing the expected utility. It has been shown that by choosing the alternative which produces the greatest expected utility, consistent decision making is upheld. (Keeney 1976, p. 131)

The advantages of utility theory and multi-attribute utility follow directly from their implementations. First, they provide a structured framework for a decision maker to operate under, and decompose a problem into manageable subproblems. They also provide quantitative maps of a decision maker's qualitative preferences, thereby lending objectivity to solutions which might otherwise have been totally subjective in nature. Finally, these theories can be used to compare non-similar items.

After explaining the terminology and assumptions used in multi-attribute utility theory, we will describe a computer program we developed to handle problems whose alternatives can be characterized by two attributes. Though simple in form, its potential usefulness is demonstrated by its application to a decision making task in tactical Air Force mission planning.

2. MULTI-ATTRIBUTE UTILITY THEORY

Multi-attribute utility theory is quite powerful when properly used. It can provide an extremely structured framework for solving difficult choice problems involving many attributes. In order to exploit its usefullness, an understanding of the terminology and methodology behind it is necessary.

2.1 Terminology and Assumptions

The terminology used in this peper is consistent with that of Keeney and Raiffa in their book titled, Decisions with Multiple Objectives: Preferences and Value Tradeoffs (1976). Assumptions are taken from the class notes of Dr. James M. Tien. The explanations are fairly cursory since we are not interested in proving the theory, merely in introducing the concepts which are used later in the computer program.

In decision making, it is frequently the case that preferences are expressed concerning various outcomes. In cases where there is no preference between outcomes, indifference expresses this relation. A statement of the form:

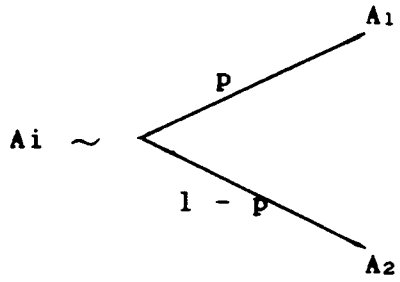
$$A \succ B$$

is read as A is preferred to B. Similarly,

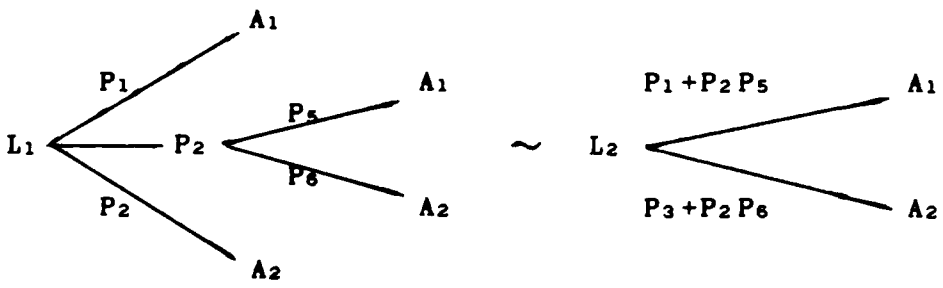
$$A \sim B$$

is read as A is indifferent to B. These relations are transitive.

A common representation of a choice in utility theory is called a lottery, and it has the form:



where p is the probability of consequence A_1 and (1 - p) is the probability of consequence A_2. Indifferent lotteries allow simplifying compound lotteries into a simple lottery:



Each consequence, Ai, is indifferent to some lottery involving the most preferred (A*) and least preferred (A_o) consequence. That is, there exists a Q_i such that:

$$Ai \sim [Q_i A_i,\ (0)A_2,\ 0(A_3),\ldots,(1 - Q_i)A_n]$$

Ai is defined as the certainty equivalent of lottery $\widetilde{Ai}$, since it can be interpreted as the selling price of a lottery, given that you own it. In any lottery, the

consequence Ai, may be replaced by $\widetilde{Ai}$ without altering the preference of that lottery:

$[P_1A_1, P_2A_2, \ldots, PiAi, \ldots PnAn] \quad [P_1A_1, P_2A_2, \ldots, Pi\widetilde{Ai}, \ldots PnAn].$

Both outcomes above are indifferent to:

$$[QA_1, (1-Q)A_n],$$

where

$$Q = P_1Q_1 + P_2Q_2 + P_3Q_3 \ldots + PnQn,$$

and $Q_1 = 1$ and $Qn = 0$. We can then say, for any two lotteries, L and L' described as:

$$L = [QA_1, (1-Q)An] \qquad L' = [Q'A_1, (1-Q)'An]$$

that $L \succsim L'$ iff $Q > Q'$. Since Ai can be substituted for the lottery, $Ai \succsim Aj$ iff $Qi > Qj$. Since we want the utility of Ai to be greater than the utility of Aj, we let:

$$U(Ai) = Qi.$$

Now, for any lottery, L, the utility is:

$$U(L) = Q = \sum_{i=1}^{n} PiQi.$$

Since utilities are indicators of preference, not absolute measurements, they are generally normalized to the range 0 to 1. This equates to a utility of 0 for the least preferred alternative (A_o) and a utility of 1 for the most preferred alternative (A^*). It is also often assumed that

monotonicity (the function increases or decreases in one direction) is characteristic of utility functions.

Increasing monotonicity is usually the case for attributes like money, happiness, beauty, etc., where more is generally better. Negative attributes such as expense, pollution, and crime, on the other hand, can ordinarily be characterized by monotonically decreasing functions. Keeney (1976) shows that transformation between monotonically increasing and monotonically decreasing functions is possible simply by changing the attributes being measured. For instance, response time measured in minutes has a decreasing utility function (shorter response times are better), but it can be changed to minutes saved in response time, where the more minutes saved, the better. (Keeney 1976, p. 141)

This brief overview of terms and assumptions used in multi-attribute utility theory is sufficient to allow the reader to follow the material in the next section, which covers assessment procedures for determining preferences. A deeper treatment of these topics can be found in the Keeney and Raiffa text mentioned previously.

2.2 Utility Assessment

One of the more critical areas of multi-attribute utility theory is obtaining information from a decision maker to produce his utility function. Some decision makers

are unfamiliar or uncomfortable with the ideas of utility theory or perhaps even probability. Some may exhibit inconsistency in their decision making processes without being aware of it. It is therefore imperative that methods exist to phrase the problem in terms meaningful to the decision maker.

According to Johnson and Huber (1977), common methodologies which have been used can be grouped into five mutually exclusive categories: ranking methods, category methods, direct methods, gamble methods, and indifference methods. (Johnson 1977, pp. 312 - 315) A general description of each method is given below, based on their assessment of the methods.

2.2.1 Ranking Methods

Ranking methods are intuitive to many decision makers. Given a number of alternatives, the decision maker assigns an order to them such that the least preferred alternative is ranked lowest, and the most preferred alternative is highest. This method works best when the alternatives are few in number. It is difficult to apply to alternatives composed of more than one attribute. Because the levels of each attribute may vary, it is not always easy to assign a preference in multi-attribute problems.

2.2.2 Category Methods

These methods only allow approximations of worth since alternative preference values are restricted to pre-specified ranges or categories. While less exact than direct methods (see 2.2.3), it is more likely easier on the decision maker to group alternatives in this manner.

2.2.3 Direct Methods

Direct methods have the decision maker assign worths (or utilities) directly to the alternatives. This is an extremely difficult procedure when the alternatives are characterized by more than one attribute. If it were not, decision makers would need no help in determining which alternative to choose.

2.2.4 Indifference Methods

These methods are can be viewed as extending gamble methods (see 2.2.5) to the multi-attribute case, without risk. They require choosing a level of an attribute to make a certain indifference relation hold. For example, the decision maker is asked, "What level of dollars will make you indifferent to the outcomes below?", where the outcomes are expressed in terms of the ordered pair (dollars, hours):

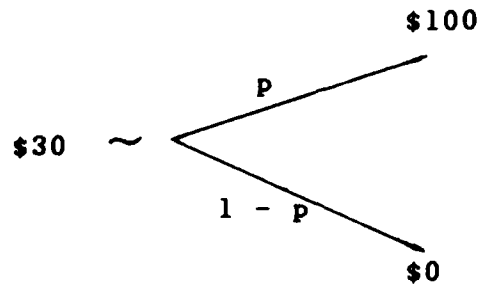
$$(? , 24 \text{ hours}) \sim (\$25 , 50 \text{ hours})$$

This technique is employed in the computer program to determine scaling constants between the two attributes.

2.2.5 Gamble Methods

These assessment methods require a decision maker to either assign probability values to a lottery so as to equate its worth with a certain given outcome, or given a lottery with fixed probabilities, determining a certainty equivalent for the lottery. Since these methods are also the ones employed by the multi-attribute utility program developed, they are considered in more detail below.

As an example of the first type, with variable probabilities, suppose a decision maker is offered a choice between \$30 and the following lottery (with dollars as the attribute measure):

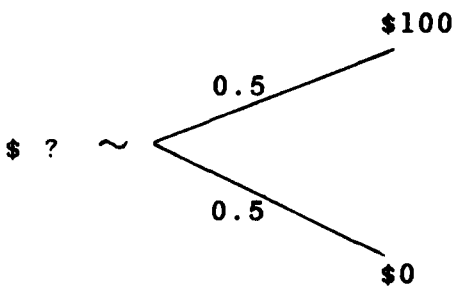


He must then choose a value for p such that the indifference relation is valid. If this value is, say, 0.6, then for consistency, this value for p must mean that the utility of receiving \$30 for certain is equal to a 0.6 chance at \$100 and a 0.4 chance of receiving \$0:

$$U(\$30) = 0.6(\$100) + 0.4(\$0)$$

Since U($100) is 1 and U($0) is 0, the utility of $30 is obviously 0.6. Continuing in this fashion, the utilities of various consequences can be assessed.

This type of gamble is extremely similar to, but not as easy on the decision maker as the second type, keeping the value of p fixed. To illustrate, suppose a decision maker is offered a lottery between receiving $100 with probability p, and probability (1 - p) of receiving $0. To make the decision easiest, the value for p is fixed at 0.5. This way, a decision maker can look at the lottery as a fifty-fifty chance between $100 and $0:

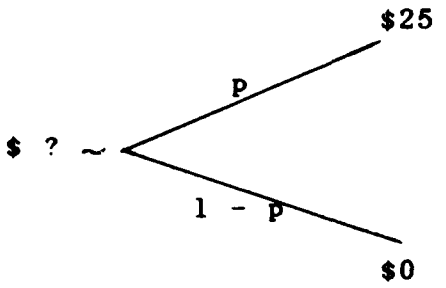


He is then asked what amount of dollars he would have to receive for certain to make him indifferent to the lottery (i,e., willing to "sell" it). Perhaps he would give up the lottery for $25. Using the formula for utility determination:

$$U(\$25) = (0.5)U(\$100) + (0.5)U(\$0)$$

$$U(\$25) = (0.5)\ (1)\ + (0.5)\ (0) = 0.5$$

To continue, the next lottery offered could be:

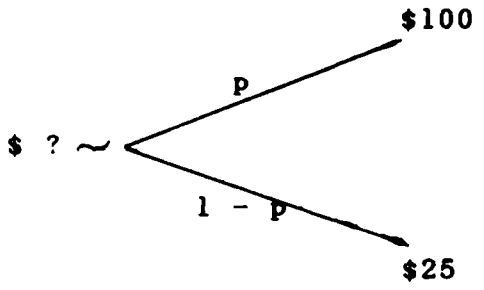


If the decision maker indicates $5 would be his certainty equivalent, the utility of $5 would be:

$$U(\$5) = (0.5)U(\$25) + (0.5)U(\$0)$$

$$U(\$5) = (0.5)\ \ 0.5\ \ + (0.5)\ (0) = 0.25$$

Once more:



If he answered $55, his utility would be:

$$U(\$55) = (0.5)U(\$100) + (0.5)U(\$25)$$

$$U(\$55) = (0.5)\ \ \ 1\ \ \ + (0.5)\ 0.5 = 0.75$$

Eventually, after enough trials, sufficient data points would have been collected to develop the utility curve.

The utility curve is rarely exactly smooth or regular due to possible slight inconsistencies. The shape of the utility curves tells us something about the decision maker. If the curve falls along the expected value line (the decision maker has specified his certainty equivalent to lie exactly midway between the extremes offered), then he is

exhibiting risk neutrality. If the curve falls above the expected value line, he is risk averse; below it, he is risk prone. These characterizations change between decision makers and even for the same decision maker, depending on their current asset positions and the ranges of the attributes.

By structuring the assessment procedure in this fashion, the decision maker has a better chance of expressing his utility in a consistent manner. If certain utility independence properties hold (discussed in the next section), this method of assessing utility can be extended to the multi-attribute case.

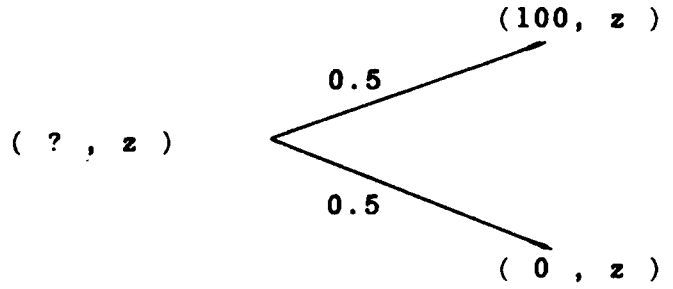
2.3 Utility Independence Properties

If it can be established that certain strong utility independence properties exist between attributes, the process of assessing a multi-attribute utility function is greatly simplified. This is because marginal utility curves for each single attribute can be combined in ways consistent with the utility independence properties, and the assessment procedure can be done for the attributes individually. The four possible combinations of utility independence properties are called: mutual utility independence, additive independence, utility independence in one direction, and no utility independence. Again, proofs of these properties can be found in Keeney and Raiffa's

Decision Making with Multiple Objectives (1976). The intent here is to summarize the information as it applies to the computer program developed. This is also why the discussion is limited to the two attribute case; each property can however, be extended to problems whose consequences can be characterized by more than two attributes.

2.3.1 Mutual Utility Independence

This form of utility independence is valid if the utility values for one of the attributes stay constant over the entire range of the other's values. For example, if the following lottery were offered (with z representing a specific value for the second attribute):



then no matter what z level is specified, the value for the first attribute would not change. This implies that the first attribute is utility independent of the second. It would also have to be valid in the reverse case (i.e., that for any specific value for the first attribute, the value for the second attribute would remain constant) for the attributes to be mutually utility independent. When this property holds, it becomes possible to combine the utilities

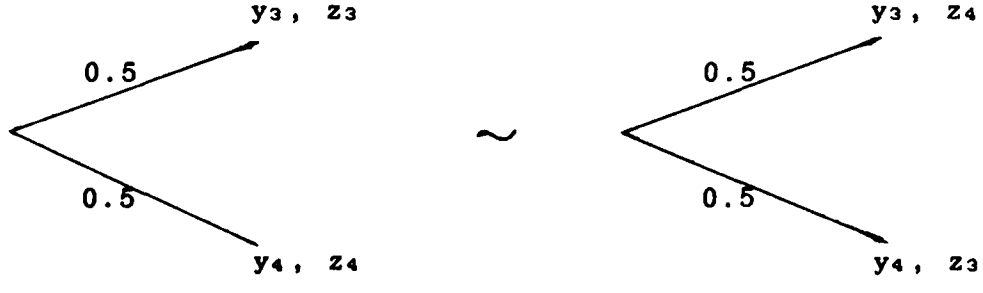
of the attributes to develop the overall utility function. If y is the first attribute and z is the second, then:

$$U(y,z) = k_Y U_Y(y) + k_Z U_Z(z) + K_{YZ} U_Y(y) U_Z(z),$$

where the k constants are used to maintain consistent scaling.

2.3.2 Additive Independence

Additive Independence is the strongest form of utility independence. It is also probably the least likely relationship existing between attributes. Additive independence in the two attribute case specifically entails that preference values for one attribute are constant over the entire range of the second attribute and vice versa (mutually utility independent). In addition, the decision maker must be indifferent to the following two lotteries (with y as the first attribute and z as the second and specific values of either are indicated by subscripts):



It is also a condition that (y_3, z_3) not be indifferent to either (y_3, z_4) or (y_4, z_3). When these conditions are met, it is possible to combine the utilities of the attributes in the following manner:

$$U(y,z) = k_Y U_Y(y) + k_Z U_Z(z)$$

where the k's again represent scaling factors.

2.3.3 Utility Independence in One Direction

It may be possible, in the two attribute case, for one attribute (y), to be utility independent of the other (z), but (z), may not be utility independent of (y). If this is true, it is said that the attributes are utility independent in the y direction. The direction of utility independence may vary, that is if z is utility independent of y, the attributes are utility independent in the z direction. (Note that if y is utility independent of z and z is utility independent of y, then mutual utility independence holds). If the attributes are such that z is utility independent of y, then:

$$U(y,z) = U(y,z_o)[1 - U(y_o,z)] + U(y,z_i)U(y_o,z),$$

and for y utility independent of z:

$$U(y,z) = U(y_o,z)[1 - U(y,z_o)] + U(y_i,z)U(y,z_o).$$

These formulas also provide close approximations when neither attribute is utility independent of the other. (Keeney 1976, p. 253)

2.3.4 No Utility Independence

It may be possible that none of the previously discussed strong utility independence properties hold. In other words, none of the previously mentioned properties can be exploited to easily assess a decision maker's utility

function. The implications here are more complex than in the other cases. In the worst situation, a decision maker has to resort to assigning utilities directly to various outcomes, and the process in painful, difficult and time consuming. The gamble method used throughout the computer program offers little support to a decision maker if no utility independence properties hold and direct assessment must be used. However, there is a possibility the attributes can be transformed into other attributes which may exhibit some strong utility independence properties. It may also be possible to decompose the attributes over a subset of their ranges to get the same result. At least, in the latter case, direct assessment of utility may only have to be done for fewer outcomes.

2.4 Utility Function Determination

After determining which utility independence properties hold, the decision maker's utility over various outcomes is obtained by assessing the appropriate utility or marginal utility curves for the attributes and combining them according to the formulas provided previously. The utility function itself can be obtained in two ways.

2.4.1 Curve Fitting Approach

Once a decision maker has provided sufficient data points, a curve can be fit through these points by various curve

fitting algorithms. Traditionally, because of time and difficulty and the ability to exploit properties of consistency and monotonicity of the utility functions, this approach has not received wide-spread use. Instead, what is more common is pre-specification of the form of the utility function.

2.4.2 Pre-Specified Form

A good discussion of this technique comes from Spetzler (1968) (who credits Pratt 1965, pp. 4-16), he shows that " a mathematical form of the utility function is assumed prior to plotting of the data. The fit of this form is then tested." (Spetzler 1968, p. 284) Obviously, for a program to consider all possible forms of utility functions is unreasonable and disadvantages of this approach include having no reasonable match for the data points. According to Spetzler (1968), another disadvantage is prejudgement of behavior patterns, but he says this can be overcome by varying the form of the function and increasing the degrees of freedom.

Spetzler (1968) also reiterates Pratt's (1965) findings that in choosing utility forms, certain theoretical considerations give indications of form as follows:

Let U(x) = utility value of x over the range in consideration.

r(x) = $-U''(x)/U'(x)$ = measure of local risk aversion as shown by Pratt. Then, if the following hold:

1) $r(x) \geq 0$,

2) the form should be continuous and twice differentiable, and

3) $r(x)$ should be constant or monotonically decreasing i.e., $r'(x) \leq 0$ over the range of x, (Spetzler 1968, p. 290)

then several common utility function forms can be used.

Common Utility Function Forms.

As multi-attribute utility theory has been applied to various decision analytic tasks -- nuclear power plant site selection (Keeney 1975), utility company power system engineering evaluation (Johnson 1977), etc -- various curves have been found to be representative of a decision maker's utility function. These functions are contained in Figure 2.1 and taken from Bolinger (1978), Stimpson (1981), and Keeney (1976).

These curves can represent several decision making patterns, indicating risk neutrality, risk aversion (constant or decreasing) and risk proneness. By assessing a decision maker's preferences for various levels of attributes, one of the pre-specified curves which best matches his curve can be used as an approximation of utility function. The amount of error allowed in fitting these pre-specified curves needs to be chosen by trading off the time to find which specific form of a curve best matches the decision maker's data, and how closely each of the specific curves match. The choice of error allowed should not be so

restrictive as to preclude a match, and because of this, consideration must be given to decision making inconsistencies.

Now that some of the underlying theory for multi-attribute utility assessment has been discussed, the next section discusses a computer graphics program which implements the two-attribute utility case.

General Equations for Common Utility Curves

$$U(x) = b_o + b_1 x$$

$$U(x) = b_o + b_1 \, x$$

$$U(x) = b_o + b_1 x^2$$

$$U(x) = b_o + b_1 \ln x$$

$$U(x) = b_o + b_1 e^x$$

$$U(x) = b_o + b_1 x + b_2 x^3$$

$$U(x) = b_o + b_1 (-e^{-cx})$$

$$U(x) = b_o + b_1 (-e^{-ax} - be^{-cx})$$

$$U(x) = b_o - b_1 e^{-cx}$$

$$U(x) = \log(x + b)$$

Figure 2.1 General Equations for Common Utility Curves

3. A TWO-ATTRIBUTE UTILITY GRAPHICS PROGRAM

This section covers the implementation details of the utility graphics program. It also provides a detailed example session which steps the reader through the utility assessment process performed by the program, providing representative figures from a computer monitor screen.

3.1 Software and Hardware Specifications

The two-attribute utility program developed to runs on a VAX 11/780 under the VMS operating system. It is written in Pascal, but makes routine use of external procedures for graphics programming. These external procedures belong to a graphics package developed by Precision Visuals, Inc., called DI-3000 software (the extended version of DI-3000 software is required to run the program). The program also uses a sub-package of the DI-3000 software called the Contouring Package. The graphics terminal used to run the program on is the Tektronix 4105, with a Tektronix 4695 hard copy jet printer attached.

The program is designed to interactively query the decision maker for the attributes of the alternatives, their ranges, and their units of measure. From there, it presents lotteries for the decision maker to give certainty equivalents to, so that the utility independence properties of the attributes can be determined. Depending on which utility independence properties hold, appropriate questions are asked to develop utility functions for the individual

attributes, which are then combined to form the overall utility function. It is then presented graphically to the user from eight possible viewpoints. Examples of a sample session are now presented.

3.2 Example Session

In this example session, suppose a decision maker is faced with a choice of receiving leisure time or money next week. His attributes characterizing the consequences would be time for the first attribute , measured in hours; and money for the second attribute, measured in dollars. The range of time he is offered is from 0 to 100 hours, and the range of money he is offered ranges from $200 to $300. The program is used to assess his utility for all possible (hours, dollars) combinations so that if offered a specific outcome, his preference would be evident.

3.2.1 Additive and Mutual Utility Independence

To start with, the program introduces itself and explains its function (fig. 3.1). It then asks the decision maker for the problem specifics (fig. 3.2). Then, it presents the decision maker with ten lotteries (five for each attribute), at the quartile points of the other attribute's range (fig. 3.3 - 3.5). Based on the decision maker's responses, one of the four utility independence properties would hold. This is determined by comparing all

Figure 3.1 Program Introduction

This program is a generic multi-attribute utility theory based methodology for assisting a decision maker. From interacting with the decision maker, it will determine the decision maker's utility function over two user specified attributes. From there, it will present the utility function graphically, at the angle of rotation desired by the decision maker.

Please enter any character followed by a return to continue:

Figure 3.2 Attribute Information

```
   Please enter your first attribute.  If the attribute is over
10 characters long, please abbreviate it to be 10 characters.

attribute name: TIME

What units is this attribute measured in?: HOURS

  Please enter the least and most preferred values this attribute
can be -- (to the nearest whole number).

least preferred: 0
most preferred: 100

   Please enter your second attribute.  Again, please insure it
does not exceed 10 characters in length.

attribute name: MONEY

What units is this attribute measured in?: DOLLARS

  Please enter the least and most preferred values this attribute
can have -- (to the nearest whole number).

least preferred: 200
most preferred: 300
```

Figure 3.3 Lottery Directions

You will now be shown a series of lotteries for which you must enter your certainty equivalents:

Hit any key followed by return to continue:

Figure 3.4 Lottery for Time at Money

In terms of the (HOURS , DOLLARS) outcome,
please enter your value for the question mark: 30

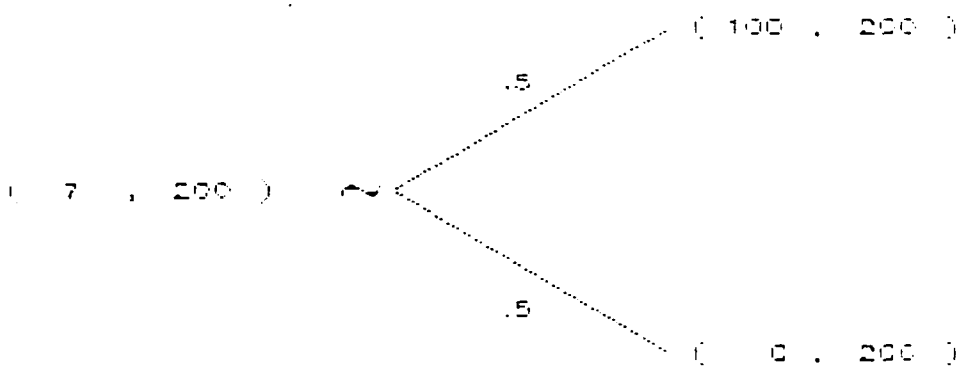
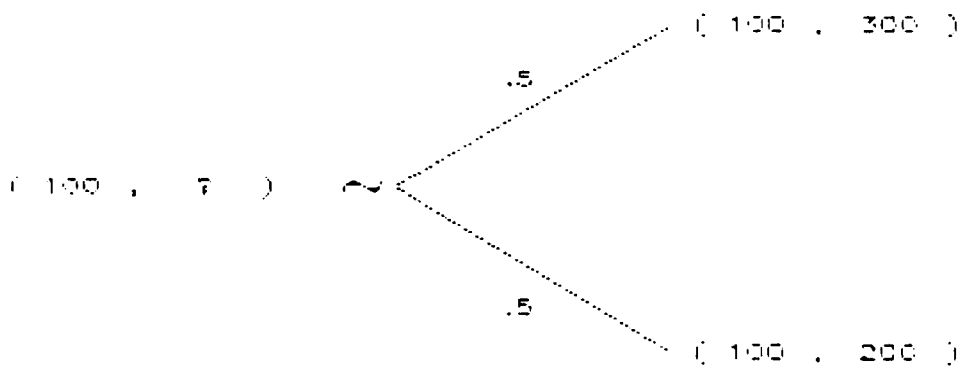


Figure 3.5 Lottery for Money at Time

In terms of the (HOURS , DOLLARS) outcome,
please enter your value for the question mark: 230



the responses for the time attribute certainty equivalents with each other and all the responses for the money certainty equivalents with each other.

When giving responses to the lotteries, it is possible that the decision maker can be slightly inconsistent. If the program detects that all responses for a particular attribute fall within 5% of the average value for the responses, it makes a consistency check before assigning any utility independence properties. It queries the decision maker to see if perhaps a single value could replace the responses (fig. 3.6 - 3.7). The previous responses are shown so that the decision maker can then leave them as is, or change the attribute values in question to a common one. Once this choice has been made, utility independence properties are determined.

If all the time attribute utility values are equal to each other and all the dollar attribute utility values are equal to each other, then at least, mutual utility independence holds, and a choice between two lotteries is offered (fig. 3.8 - 3.9). Depending on the decision maker's response (choice 1, or either of 2 or 3), a distinction is made between additive or mutual utility independence (fig. 3.10 and 3.11). Remember that additive independence exists if mutual utility independence exists and the decision maker indicates indifference between the two lotteries (i.e., choice 1).

Figure 3.6 Possible Inconsistency Indicated

In terms of the (HOURS , DOLLARS) outcome, since
your responses for TIME all fall within 5 percent of the average,
please carefully reconsider your previous answers. You will
need to determine if the attribute values should be changed to a
common value, and if so, which one. (NEXT SCREEN)

Hit any key followed by return to continue:

Figure 3.7 Previous Responses for Time C.E.'s

```
(  30,  200) ~ [0.5( 100,  200) ; 0.5(   0,  200)]

(  30,  250) ~ [0.5( 100,  250) ; 0.5(   0,  250)]

(  29,  225) ~ [0.5( 100,  225) ; 0.5(   0,  225)]

(  30,  300) ~ [0.5( 100,  300) ; 0.5(   0,  300)]

(  30,  275) ~ [0.5( 100,  275) ; 0.5(   0,  275)]

Please enter the number corresponding to your choice.
  1) Change previous answers to a common one.
  2) Leave answers as is.
Choice: 1
What value would you like to change it to?: 30
```

Figure 3.8 Lottery Specifying Mutual Utility Independence

With respect to the lotteries below, in terms of the
(HOURS , DOLLARS) outcome, please indicate
which statement corresponds to your feelings about this lottery:
1) I am indifferent between the lotteries.
2) I prefer the lottery on the left.
3) I prefer the lottery on the right.
Choice: 3

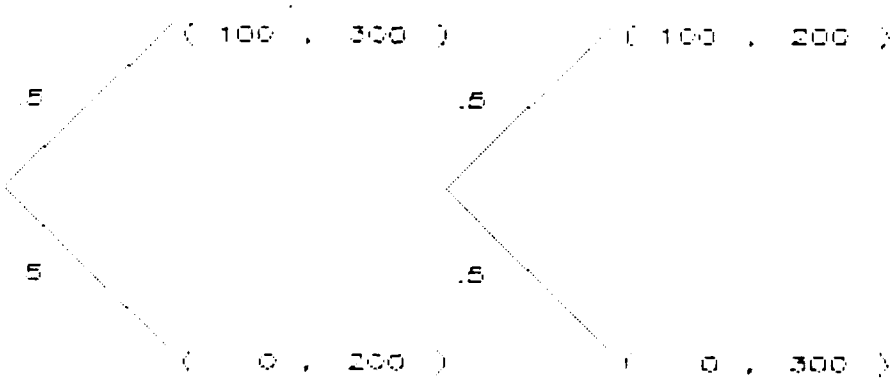


Figure 3.9 Lottery Specifying Additive Independence

With respect to the lotteries below, in terms of the
(HOURS , DOLLARS) outcome, please indicate
which statement corresponds to your feelings about this lottery:
1) I am indifferent between the lotteries.
2) I prefer the lottery on the left.
3) I prefer the lottery on the right.
Choice: 1

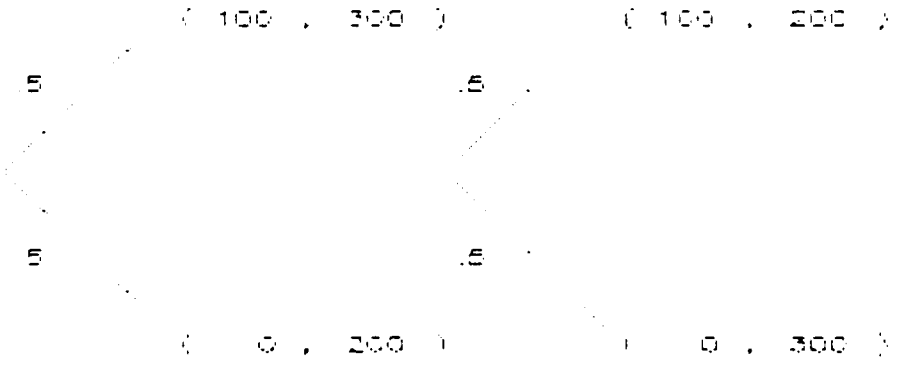


Figure 3.10 Mutual Utility Independence Indicated

Since your responses indicate that the
attributes are mutually utility independent,
your utility function can easily be developed by assessing
the marginal utility curve for each attribute. To do this,
you will be shown additional lotteries for which you must
enter your certainty equivalents.

Hit any key followed by return to continue:

Figure 3.11 Additive Independence Indicated

Since your responses indicate that the
attributes are additive independent,
your utility function can easily be developed by assessing
the marginal utility curve for each attribute. To do this,
you will be shown additional lotteries for which you must
enter your certainty equivalents.

Hit any key followed by return to continue:

The distinction between additive and mutual utility independence is important, because it dictates the form of the utility function used. However, marginal utilities of the two attributes are used in both cases, so the procedures for determining each attribute's individual utility curve and associated scaling constants are the same.

In this program, only one pre-specified form of utility function is used for simplicity. It is also a flexible, commonly used function. (Sheridan 1977, p. 392) This function is of the form:

$$U(x) = b(1 - e^{-cx}).$$

This form requires only a single response from the decision maker to develop the utility function for time (fig. 3.12) and a single response to develop the utility function for money (fig. 3.13). It is also a fairly realistic function to apply except in cases where the decision maker's certainty equivalents lie too close to the most and least preferred consequence. If this is the case, the iterative optimization procedure for finding values for the b and c variables does not converge to the specified maximum error allowed by the program (.0001 units) within a reasonable time (35 iterations), and the decision maker is informed that his utility can not be assessed by the utility program (fig. 3.14).

Figure 3.12 C.E. for Time Marginal Utility Curve Determination

Please enter your certainty equivalent for the "?": 30

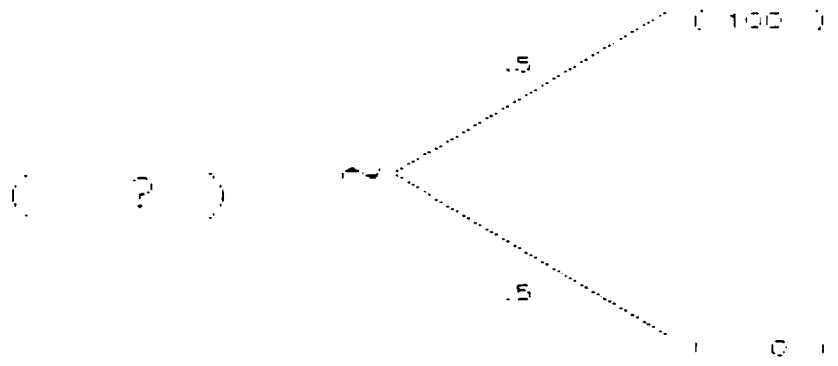


Figure 3.13 C.E. for Money Marginal Utility Curve Determination

Please enter your certainty equivalent for the "?": 230

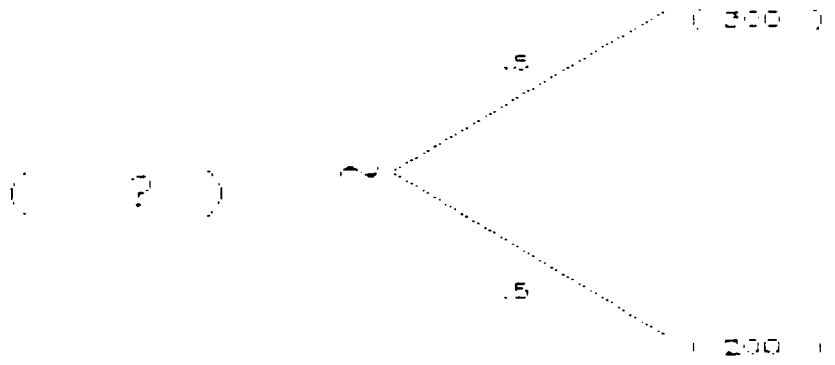


Figure 3.14 Program Inability to Assess Utility

```
   The prespecified curve used to approximate your utility
curve doesn't match your values well enough to be used as
an accurate representation.  Unfortunately, this program
can't be used to assess your utility. SORRY!


Hit any key followed by return to exit back to system level.
```

If, on the other hand, the form can be used for each attributes utility curve, it becomes necessary to determine their scaling constants. One necessary condition is that

$$k_Y + k_Z + k_{YZ} = 1.$$

Here, k_Y is the scaling constant for the first attribute (generically termed y) and k_Z is the scaling constant for the second attribute (generically termed z). In the additive independence case, k_{YZ} is 0 so that

$$k_Y + k_Z = 1.$$

We need to put k_Z in terms of k_Y or k_Y in terms of k_Z to remove one unknown in the equation. This is done by asking the decision maker to specify a preferred outcome, either the (most preferred of attribute y, least preferred of attribute z) or the (least preferred of attribute y, most preferred of attribute z). Depending on the decision maker's choice, if the first outcome is preferred, a value for attribute y is sought to complete the indifference relation. (fig. 3.15) (If the second outcome were preferred, a value for attribute z would be sought, and if the decision maker were indifferent between the two, the utility values would already be available).

From this indifference value, it is possible to establish the relationship between k_Y and k_Z (see Keeney 1976, pp. 278 - 281 for details), such as $k_Z = .28k_Y$ or $k_Y = .36k_Z$, etc. Finally, the last unknown must be solved for.

Figure 3.15 Finding Scaling Relationship

```
In terms of the (HOURS      , DOLLARS    ) outcome,
Please enter your preference.

     (       100,        200)   or   (         0,        300)

1) I prefer the outcome on the left.
2) I prefer the outcome on the right.
3) I am indifferent between outcomes.
Choice: 1



What amount of HOURS       would make you indifferent to
the outcomes below?

(  ? ,        200) ~ (         0 ,       300)
amount of HOURS      : 50
```

This requires the decision maker to specify an outcome such that it has a utility of 0.5.

The process of choosing this outcome is not handled very elegantly in this program. Many possible combinations of the two attributes exist which form a surface whose utility value is 0.5. Ideally, a convergence technique to help the decision maker focus on a particular outcome is preferable. However, as it stands now, the decision maker is faced with a direct assessment problem (fig. 3.16). This results in a solution from the following equations (if, for example, k_Y were $.36k_Z$):

$k_Y = .36k_Z$,

$k_{YZ} = 1 - .36k_Z - k_Z$, and

$0.5 = .36k_Z(U_Y) + k_Z(U_Z) + (1 - .36k_Z - k_Z)(U_Y)(U_Z)$.

Since U_Y and U_Z are known because of their utility curves, the only unknown is k_Z. Once k_Z is found, k_Y is found, and then k_{YZ}. Keep in mind here that if additive independence holds, $k_{YZ} = 0$.

Now the program fills an array with utility values for possible outcomes of attribute y and z, and asks the decision maker from which viewpoint he prefers to see the utility graph (fig. 3.17 - 3.18). Then, depending on the response, it draws one of eight graphs (here reflecting the results of the example session answers for the mutual utility independent case) (fig. 3.19 - 3.26). The other possibilities for utility independence is utility

Figure 3.16 Indiference Point Selection

```
In terms of the (HOURS      , DOLLARS    ) outcome,
what values of HOURS       (?) and DOLLARS     (#)
will make you indifferent between the c.e. and the lottery?
Value of HOURS      : 30
Value of DOLLARS    : 230
```

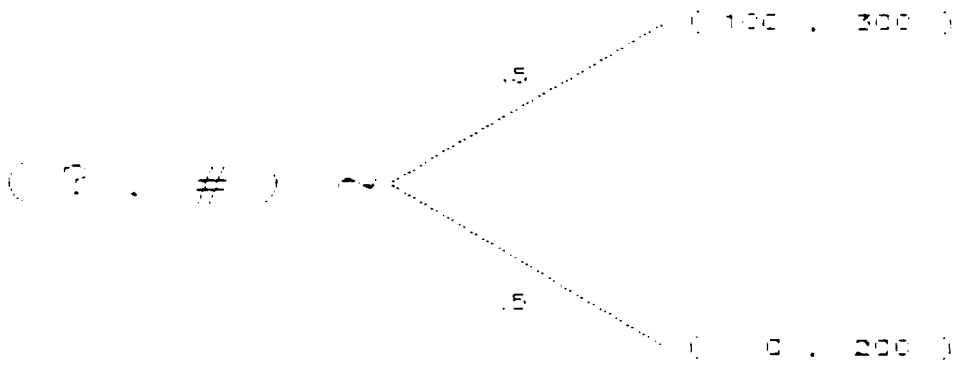


Figure 3.17 Representation of Viewpoints

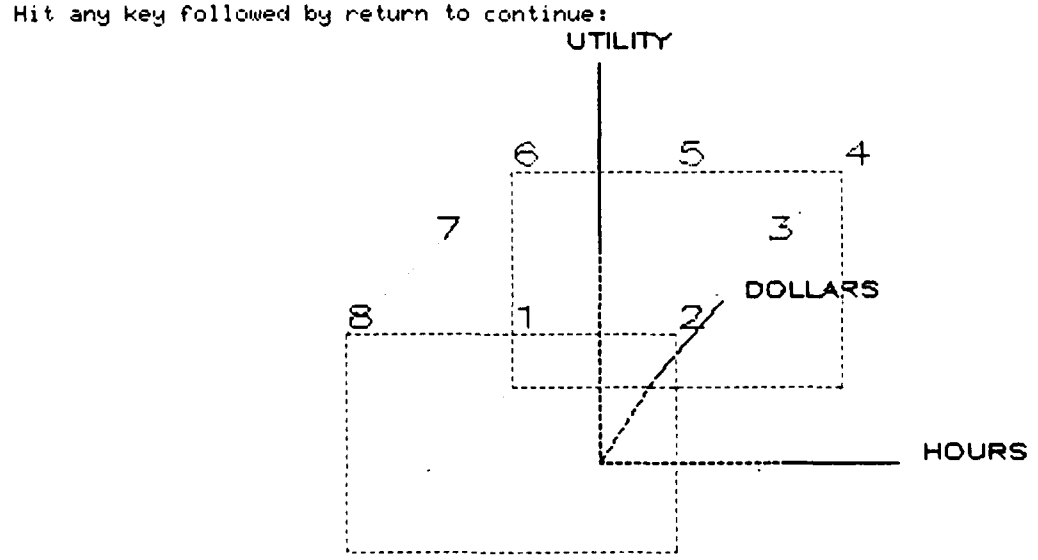


Figure 3.18 Viewpoints Offered

```
Please indicate your choice of viewpoint for the utility graph.
1)front        3)right        5)back        7)left
2)front-right  4)back-right   6)back-left   8)front-left  9)quit
choice:
```

Figure 3.19 Viewpoint 1 (Example Session)

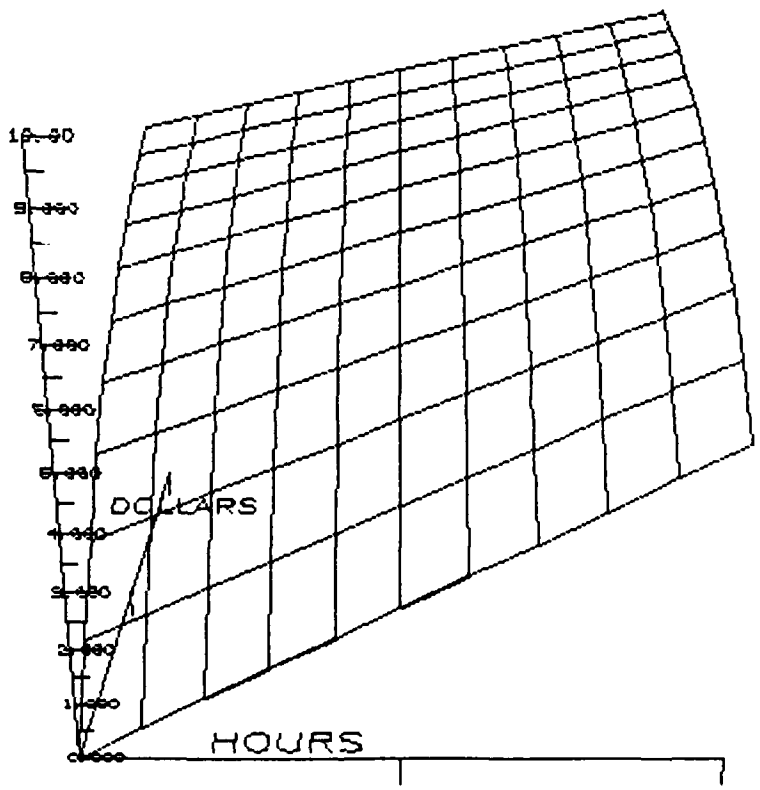


Figure 3.20 Viewpoint 2 (Example Session)

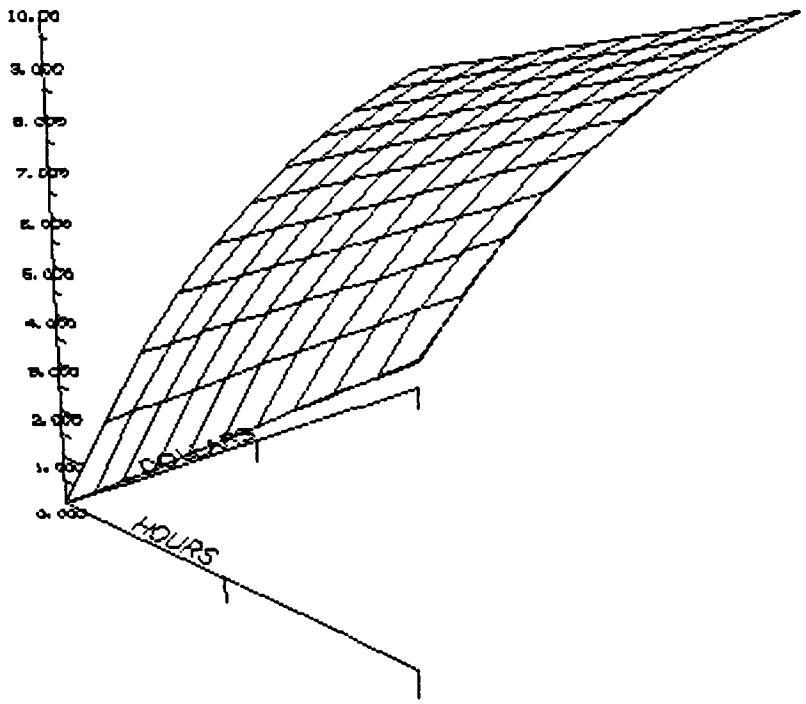


Figure 3.21 Viewpoint 3 (Example Session)

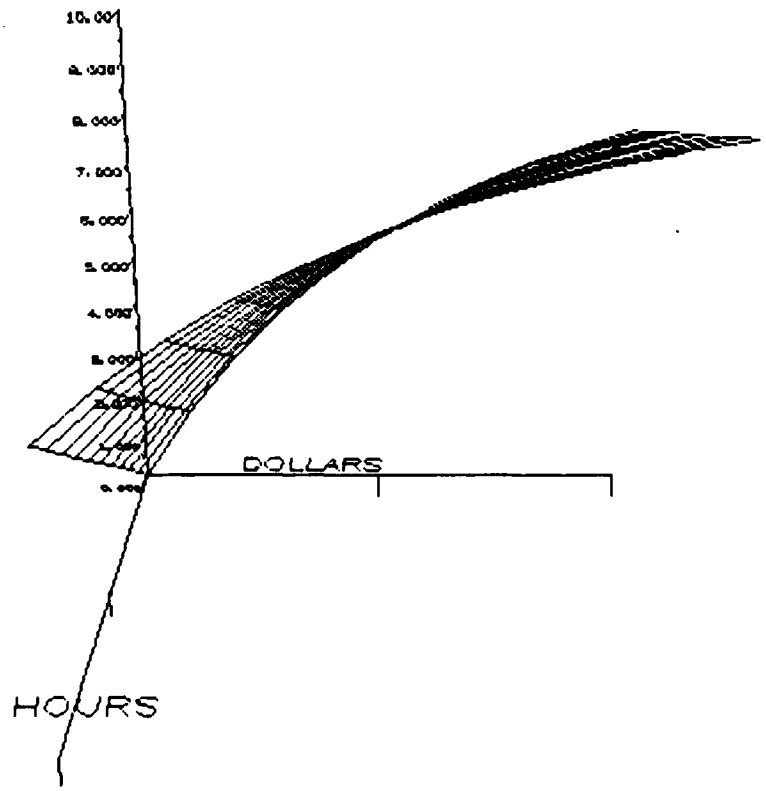


Figure 3.22 Viewpoint 4 (Example Session)

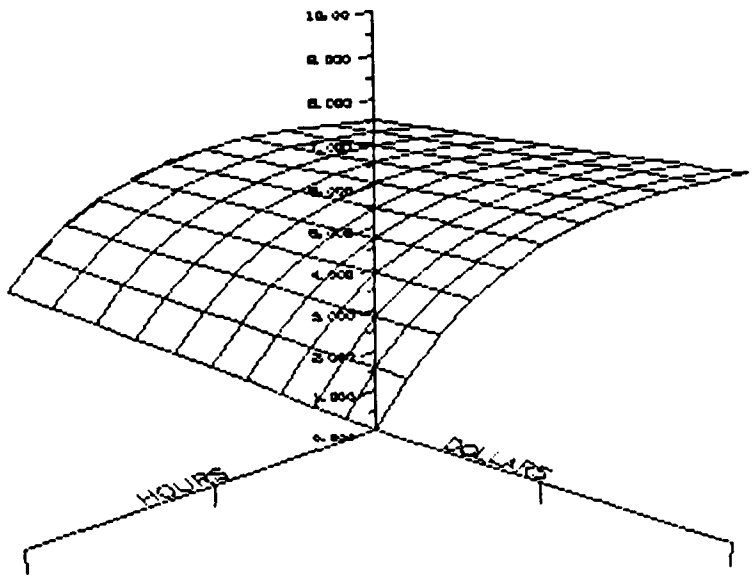
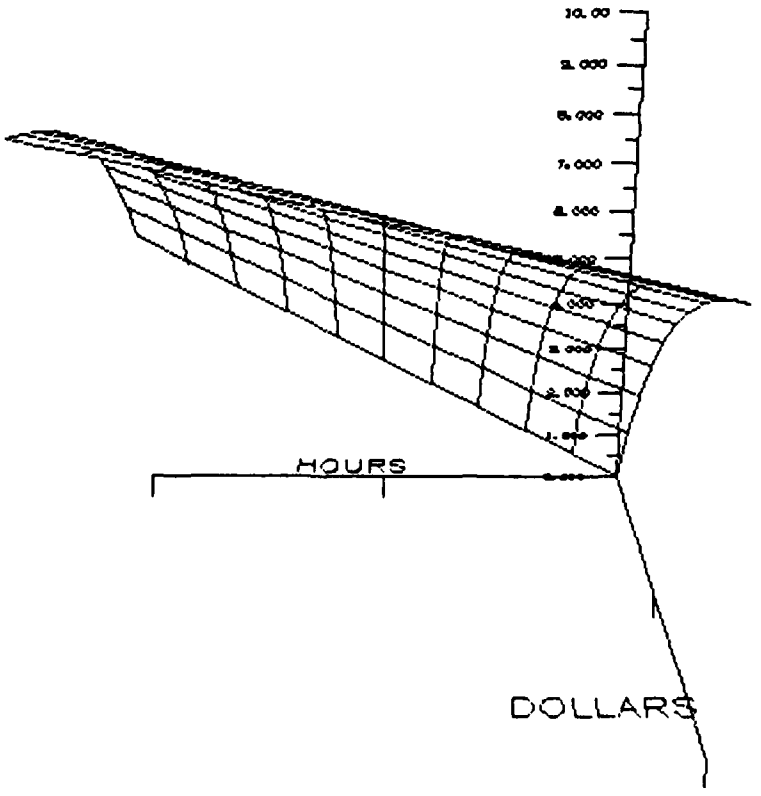


Figure 3.23 Viewpoint 5 (Example Session)



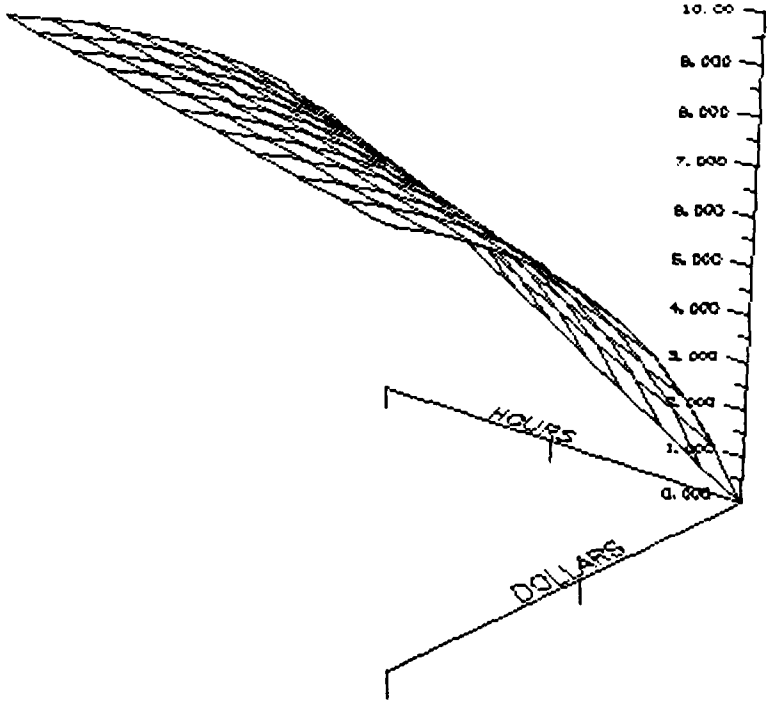


Figure 3.24 Viewpoint 6 (Example Session)

Figure 3.25 Viewpoint 7 (Example Session)

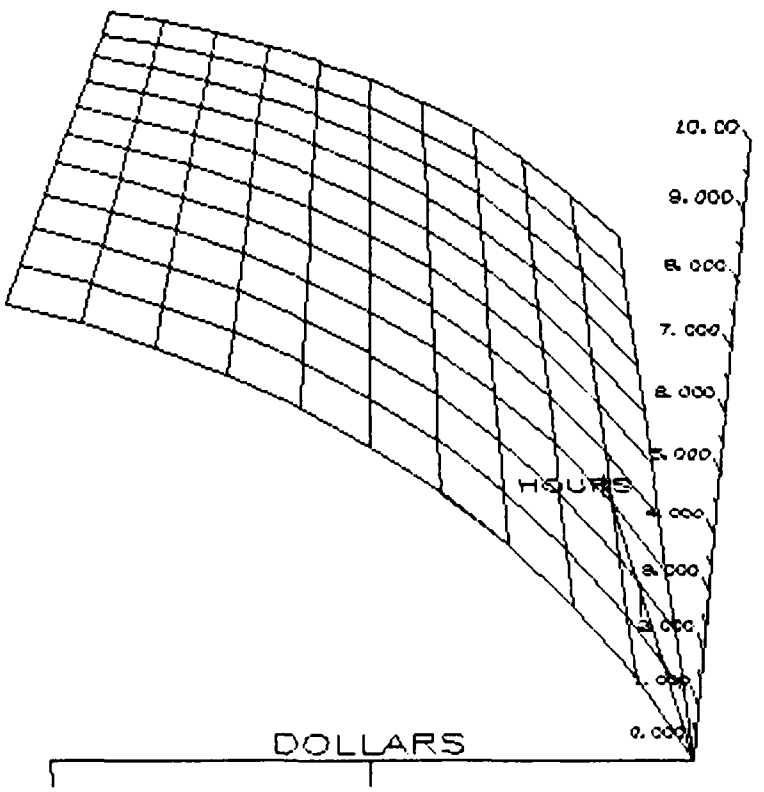
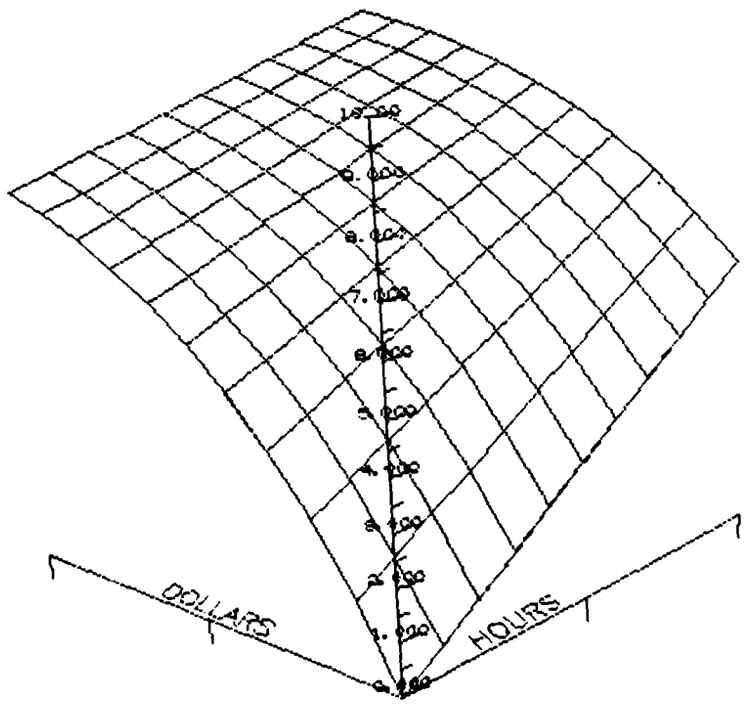


Figure 3.26 Viewpoint 8 (Example Session)



independence in one direction (either direction), or no utility independence for the attributes.

3.2.2 Utility Independence in One Direction

If the responses for only one of the two attributes are all equivalent (say for dollars) then the utility program informs the decision maker (fig. 3.27) and obtains the values necessary to develop the three marginal utility curves (fig. 3.28 - 3.30). These curves are then combined in the manner previously indicated (section 2.3.3) for attributes utility independent in one direction. If none of the previously mentioned properties exist, the fourth case, of no utility independence between attributes, is chosen.

3.2.3 No Utility Independence

This case is indicated by non-equivalence of the certainty equivalents obtained by asking the first ten lotteries. Keeney (1976), however, has indicated it may be possible to avoid taking a direct assessment approach immediately by either transforming the problem attributes into ones which do exhibit a form of utility independence, or by decomposing the attribute ranges into subsets over which utility independence properties hold (fig. 3.31). (Keeney 1976)

If the decision maker chooses the first option, he is expected to provide the transformed attributes and rerun the

Figure 3.27 Utility Independence One Way Indicated

```
Since your responses indicate that DOLLARS     are
utility independent of HOURS       ,
your utility function can easily be assessed by developing
three marginal utility curves for the attributes.  To do this,
you will be shown additional lotteries for which you must
enter your certainty equivalents.


Hit any key followed by return to continue:
```

Figure 3.28 C.E. for Money Marginal Utility Curve Determination

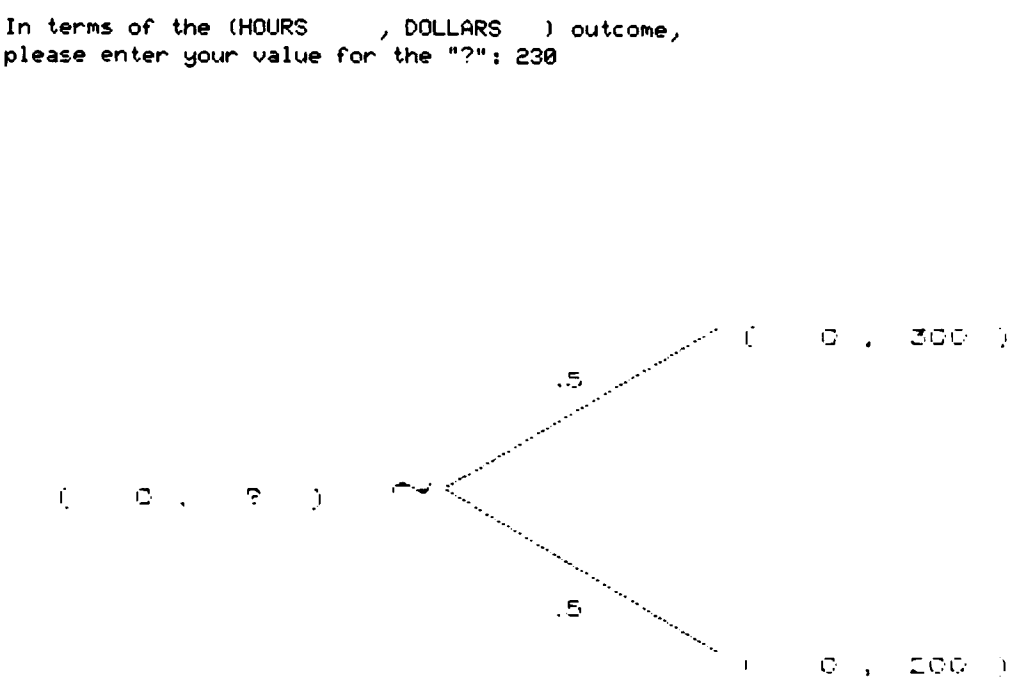


Figure 3.29 C.E. for Time Marginal Utility Curve Determination (at $Money_{min}$)

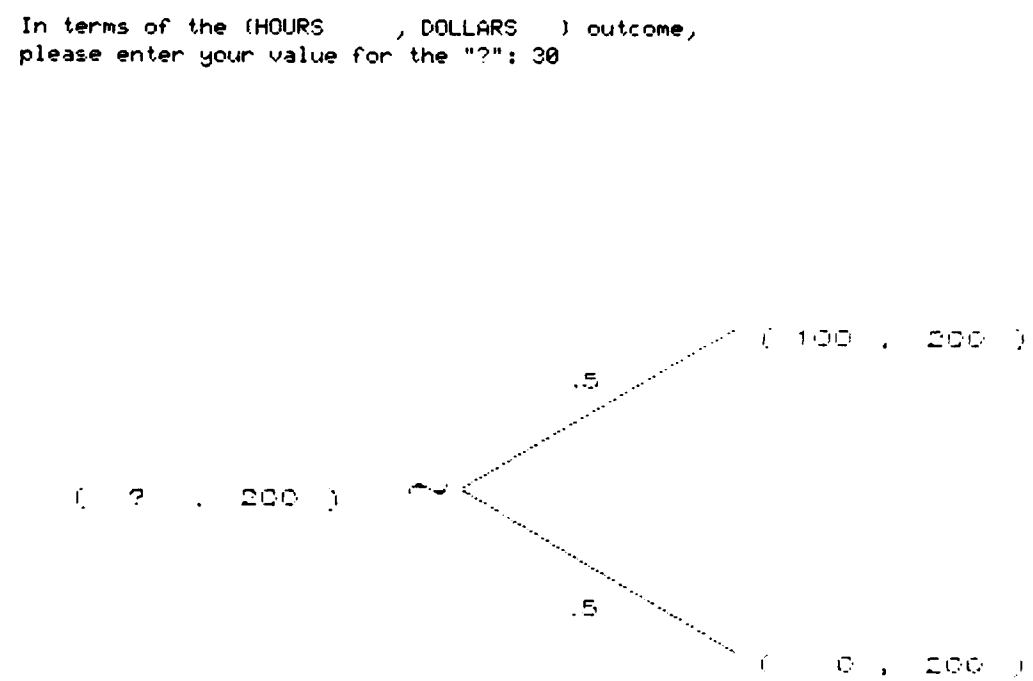


Figure 3.30 C.E. for Time Marginal Utility Curve Determination (at $Money_{max}$)

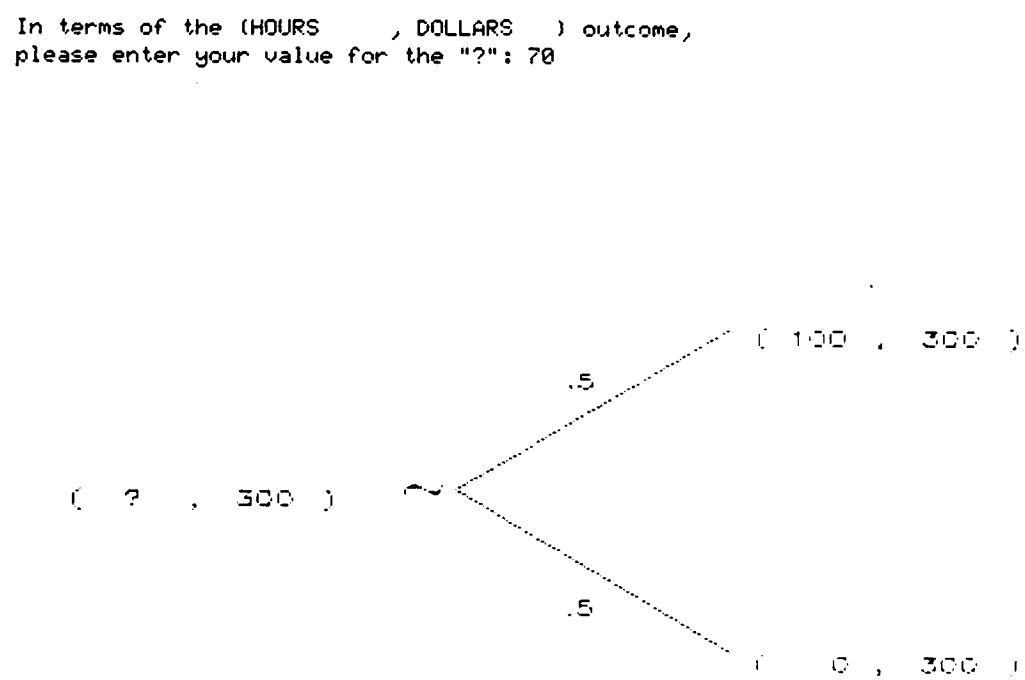


Figure 3.31 No Utility Independence Indicated

```
Since your responses indicate that the attributes are not at all
utility independent, there are three possible approaches to
assessing utility values for the attributes. The first approach
requires you to rerun the utility program using transformed
attributes which may exhibit utility independence. The second
approach requires decomposing the attribute ranges and
then assessing the utility function over these decomposed
ranges, which may be utility independent. Further guidance on
this option is provided when you choose it. Finally, the
last approach requires you to indicate your utilities directly
for different combinations of attributes. Since this is the
most difficult option to perform, it is recommended you try
one of the other two if possible.


Please indicate which option you want.
1) Use transformed attributes.
2) Decompose the attribute ranges.
3) Direct assessment.
Choice: 2
```

program (the program automatically restarts). If he chooses to decompose the ranges, an explanatory paragraph gives an example of how to do this (fig. 3.32) and he is offered a chance to view his previous responses to the initial ten lotteries (the answers are of the same form as those presented during the consistency check). The utility program is then restarted. Should the decision maker choose the direct assessment option, he is warned about its difficulty and told how to proceed (fig. 3.33). After that, he is shown thirty six consequences over the attribute ranges for which he must give his utility directly (fig. 3.34 - 3.39). In both the utility independent in one direction and no utility independence cases, the same viewpoints and graphs of the utility function are offered.

To show the applicability of even a simple two-attribute utility program to a prototypical problem, we were assisted by two Air Force officers from the 24th Air Division, Griffiss AFB. They allowed us to assess their utilities for tactical mission planning outcomes and the sessions are recorded in the next section.

Figure 3.32 Decomposition of Ranges Chosen

```
   Decomposing the attribute ranges should be done if there
is indication of utility independence over a subset of the range.
This is indicated by having a constant certainty equivalent for
an attribute even when the other attribute's value varies.  For
example, if your certainty equivalent for attribute one (ranging
from, say, 0 to 100) is always constant (say, 60) over a subrange
(say, less than 350) of attribute two (ranging from, say, 200 to
400), then a good place to subset the range would be from 200 to
350 and 350 to 400.  This way, over the 200 to 350 range, some
utility independence properties will hold.

Would you like to see your original answers again?
1) yes
2) no
Choice: 2

Please indicate which operation you want to perform.
1) Return to options menu.
2) Perform the analysis over a subset of ranges.
Choice: 1
```

Figure 3.33 Direct Assessment Chosen

```
    This procedure is difficult to perform because it is te-
dious, and you must have a good feel for your preferences. You
will be asked to assign a number from 0 to 1.0 to a variety of
outcomes.  Please think carefully about your choices.


Hit any key followed by return to continue:
```

Figure 3.34 Direct Assessment Points 1-6

Based on the scale below, please enter the number you feel indicates the worth of the following outcomes. The number entered may be real or integer between and including 0 through 1.0.

```
(TIME      , MONEY     )
(          0,        200) : 0.0
(         20,        200) : 0.15
(         40,        200) : 0.3
(         60,        200) : 0.4
(         80,        200) : 0.45
(        100,        200) : 0.5
```

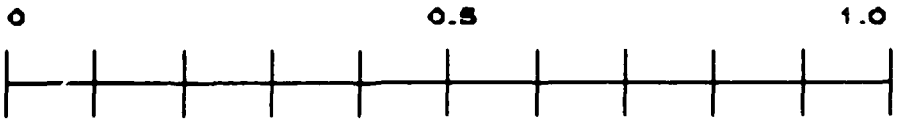


Figure 3.35 Direct Assessment Points 7-12

(TIME	, MONEY	)	
(	0,	220)	: 0.15
(	20,	220)	: 0.3
(	40,	220)	: 0.45
(	60,	220)	: 0.55
(	80,	220)	: 0.6
(	100,	220)	: 0.65

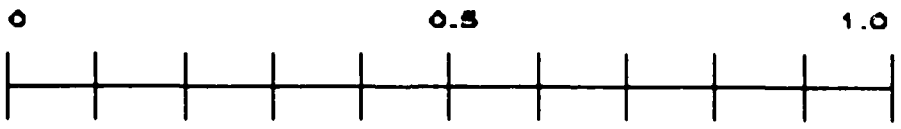


Figure 3.36 Direct Assessment Points 13-18

```
(TIME      , MONEY     )
(         0,         240) : 0.3
(        20,         240) : 0.45
(        40,         240) : 0.6
(        60,         240) : 0.7
(        80,         240) : 0.75
(       100,         240) : 0.8
```

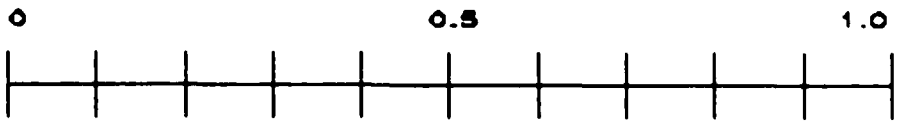


Figure 3.37 Direct Assessment Points 19-24

(TIME	, MONEY	)	
(	0,	260)	: 0.4
(	20,	260)	: 0.55
(	40,	260)	: 0.7
(	60,	260)	: 0.8
(	80,	260)	: 0.85
(	100,	260)	: 0.9

Figrue 3.38 Direct Assessment Points 25-30

(TIME	, MONEY	)	
(	0,	280)	: 0.45
(	20,	280)	: 0.6
(	40,	280)	: 0.75
(	60,	280)	: 0.85
(	80,	280)	: 0.9
(	100,	280)	: 0.95

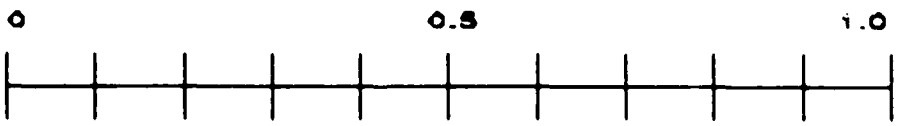
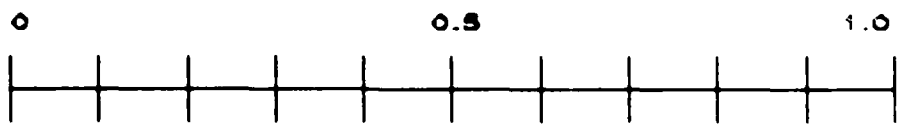


Figure 3.39 Direct Assessment Points 31-36

(TIME	, MONEY	)	
(	0,	300)	: 0.5
(	20,	300)	: 0.65
(	40,	300)	: 0.8
(	60,	300)	: 0.9
(	80,	300)	: 0.95
(	100,	300)	: 1.0



4. APPLICATION TO MISSION PLANNING

This section describes an attempt to determine the utility graphs of two Air Force mission planners. The process is described via figures of the computer screen taken during the utility assessment session. A five step approach is followed, in conjuction with running the program, to perform the utility assessment.

4.1 Problem Discussion

The idea for applying the utility program to mission planning arose from some research work being done for Rome Air Development Center (RADC) at Griffiss AFB. There, in the Decision Aids Section, a decision aid is being developed to determine mission success based on a mission's probability of arrival (Pa) to a target, and its probability of destruction (Pd) of the target. The Pd for a mission is readily available from the Joint Munitions Effectiveness Manual, but the Pa is a fairly subjective assessment arrived at from intelligence estimates of enemy threat. Since the problem was characterized in this fashion, we decided it was an ideal one for the utility program.

The two officers whose utilities were assessed were Major Rand Case and Major Robert Stan. Both men had prior experience in the mission planning area and are currently assigned to the 24th Air Division, Live Exercise Division at

Griffiss AFB. To assess their utilities, the five step process outlined by Keeney (Keeney 1976, p. 261) was used.

4.2 Five Step Assessment Procedure

This procedure gives a systematic approach for assessing utility, but there is no guarantee that following it results in a correctly assessed utility function. The chances of having a correctly assessed function increase if an experienced analyst elicits the information from the decision maker or the decision maker is sufficiently informed and comfortable with the ideas behind multi-attribute utility theory. Since both officers were somewhat familiar with the concept, and the computer program provided the format to follow, the author hopes that my inexperience as an analyst did not totally prejudice the process. Following the five step process, we proceeded as follows.

4.2.1 Introduce Terminology and Ideas

Both officers were present at this part. We stressed that it was not the kind of a problem where either of their preferences were right or wrong, merely an indication of how they felt about the problem at hand. We then asked if it was reasonable to characterize the problem of assigning missions as being dependent on the two attributes Pa and Pd. Both men felt it could be broken down into consequences of this nature, but that while Pd was easily obtainable, Pa was

still a fairly ethereal concept. Both agreed, however, that if Pa could be determined, it would indicate, along with Pd, the mission's worth.

Since both men were familiar with the concept of utility, we only had to familiarize them with the specific terminology of the computer program as outlined in section 2.1. We also ran through an example session of the program so they could see how it operated. We then asked Major Stan to sit aside while we worked with Major Case.

4.2.2 Utility Assessment for Mission Planner 1

To find the ranges of Pa and Pd over which to assess his utility, we asked him if there were some value of Pa and Pd below which he would never assign a mission. He said it depended on the mission, and we asked him to think of a specific type. He said fire suppression. His determination was that no mission should be sent without at least a 45% chance of arriving and a 25% chance of destroying the target (fig 4.1). This meant the consequence -- in terms of (Pa, Pd) -- for the least preferred was (45, 25) and most preferred was (100, 100). The next step was performed by the utility program.

Identify Relevant Independence Assumptions.

Major Case answered the initial ten lotteries (fig. 4.2 - 4.11). During this process, both he and Major Stan had some problems. Initially, both officers wanted to enter the

Figure 4.1 Attribute Information

```
   Please enter your first attribute.  If the attribute is over
10 characters long, please abbreviate it to be 10 characters.

attribute name: PARRIVE

What units is this attribute measured in?: PA

  Please enter the least and most preferred values this attribute
can be -- (to the nearest whole number).

least preferred: 45
most preferred: 100

   Please enter your second attribute.  Again, please insure it
does not exceed 10 characters in length.

attribute name: PDESTROY

What units is this attribute measured in?: PD

  Please enter the least and most preferred values this attribute
can have -- (to the nearest whole number).

least preferred: 25
most preferred: 100
```

Figure 4.2 Lottery for Pa at Pd_{min} (MP 1)

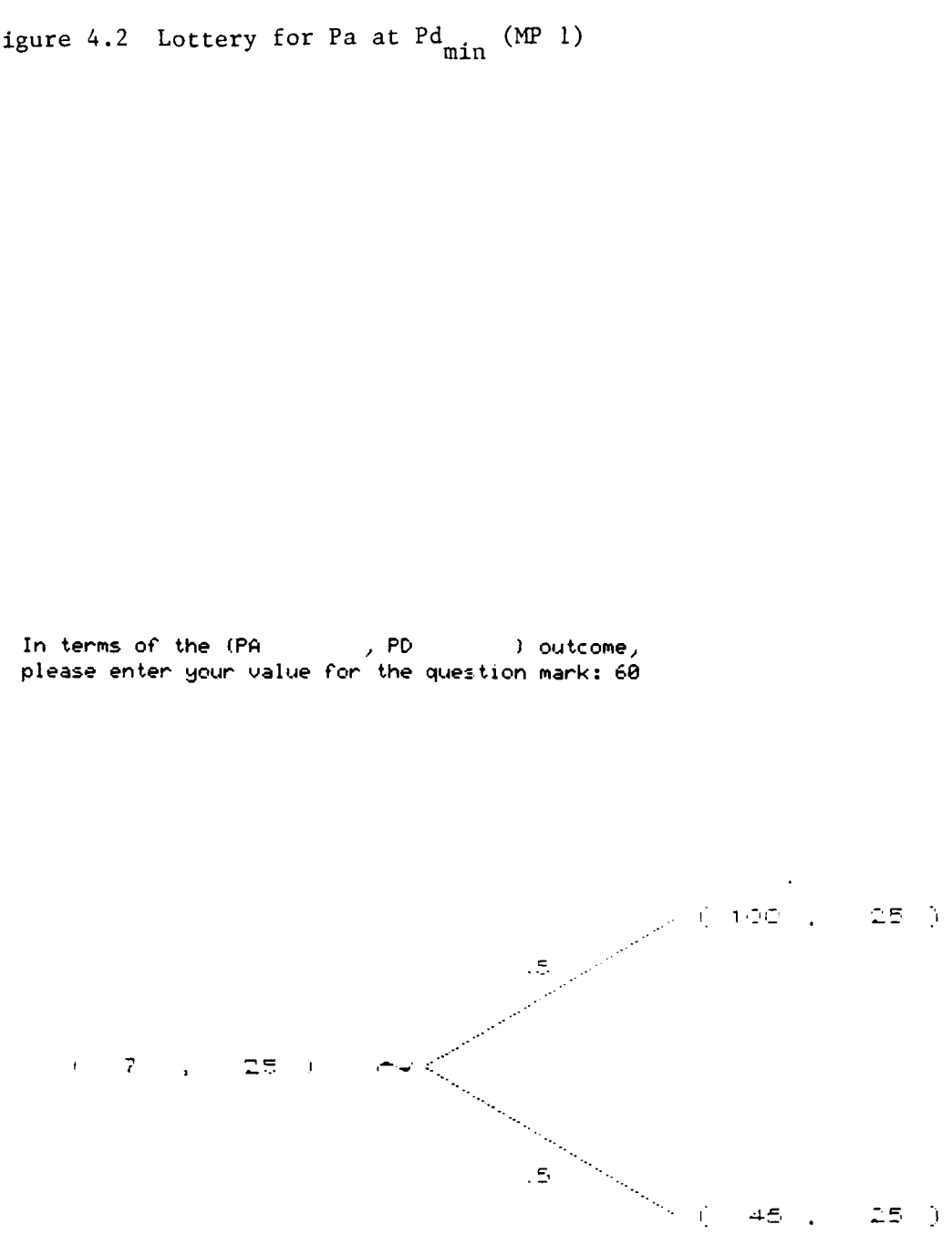


Figure 4.3 Lottery for Pa at $Pd_{.50}$ (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 75

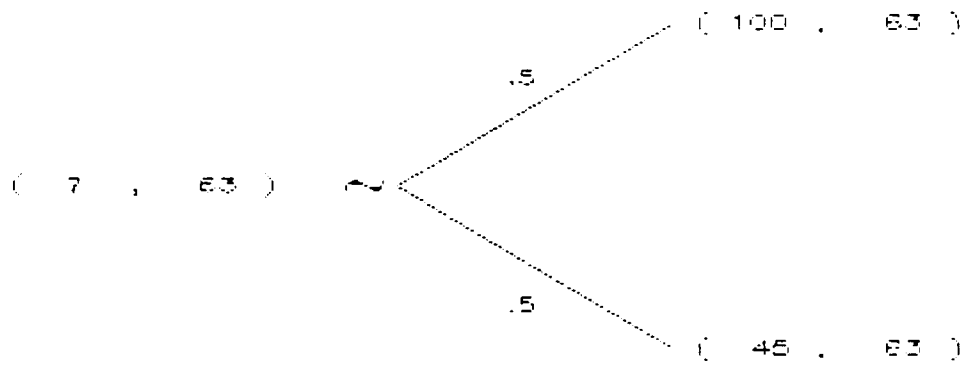


Figure 4.4 Lottery for Pd at Pa_{max} (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 50

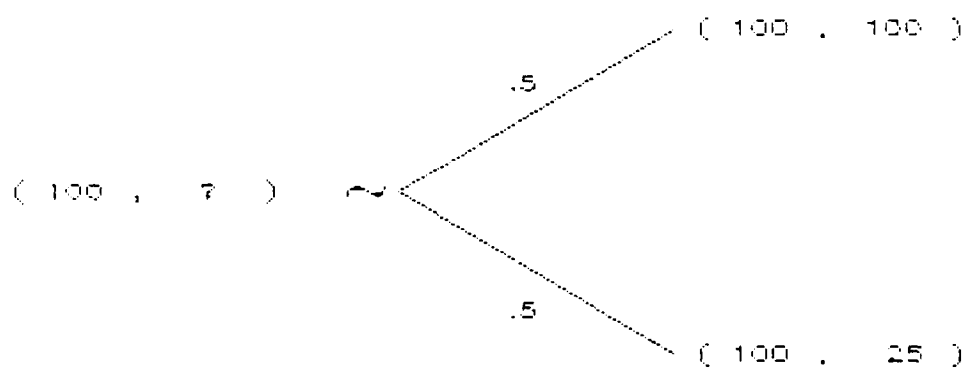


Figure 4.5 Lottery for Pa at $Pd_{.25}$ (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 55

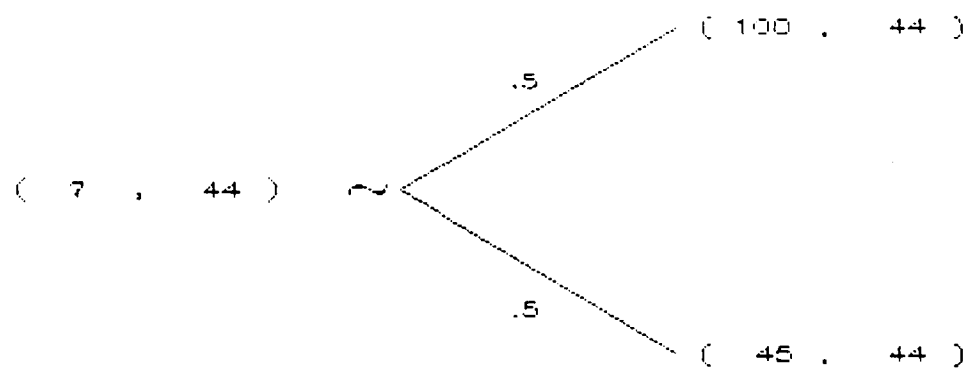


Figure 4.6 Lottery for Pd at $Pa_{.50}$ (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 50

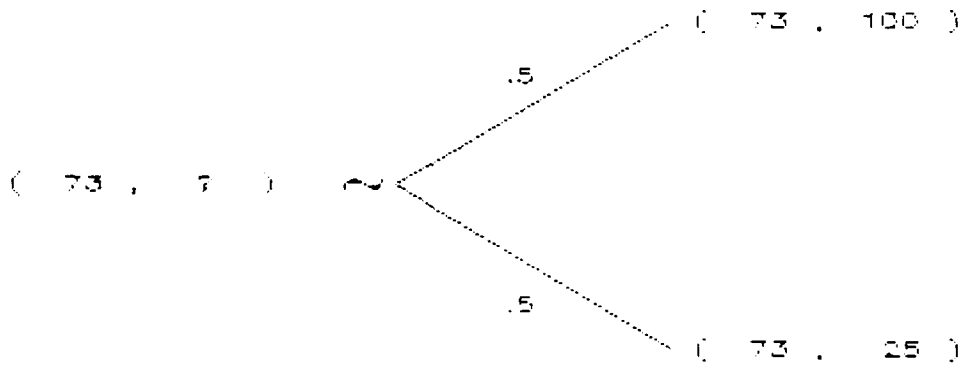


Figure 4.7 Lottery for Pd at $Pa_{.25}$ (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 50

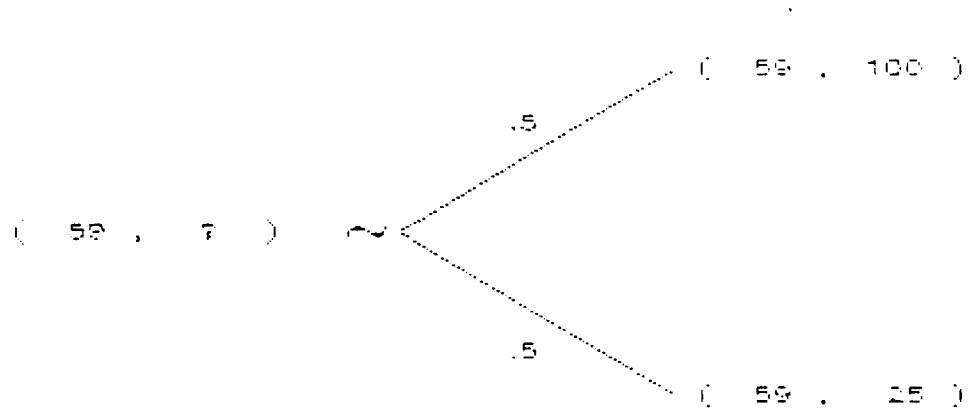


Figure 4.8 Lottery for Pa at Pd_{max} (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 60

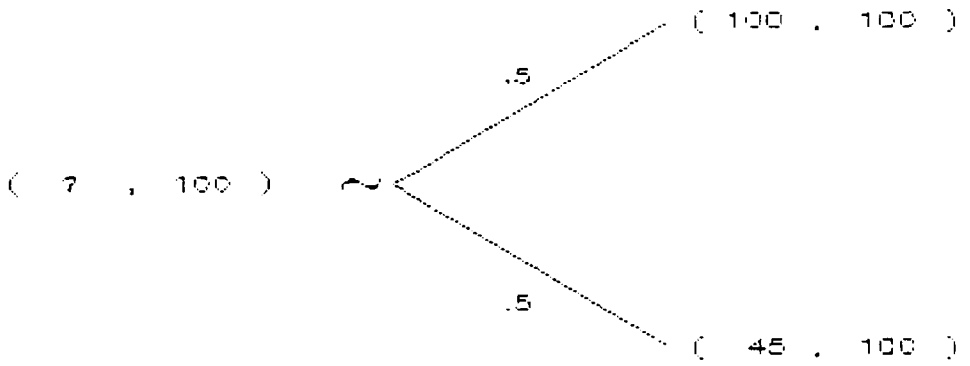


Figure 4.9 Lottery for Pd at Pa_{min} (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 50

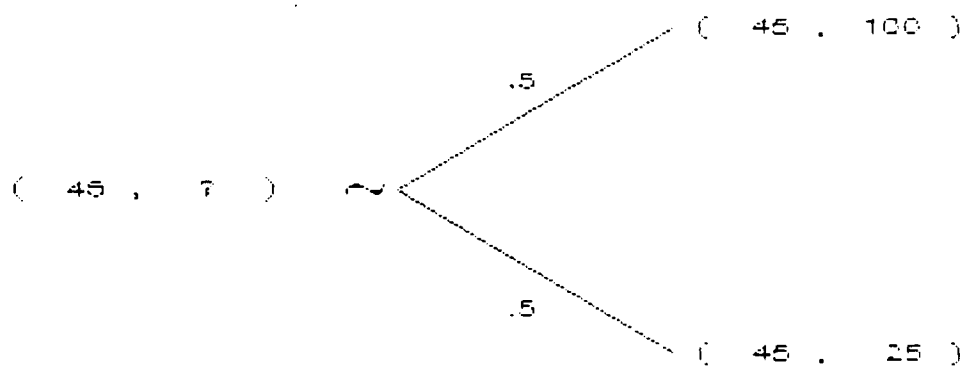


Figure 4.10 Lottery for Pa at $Pd_{.75}$ (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 70

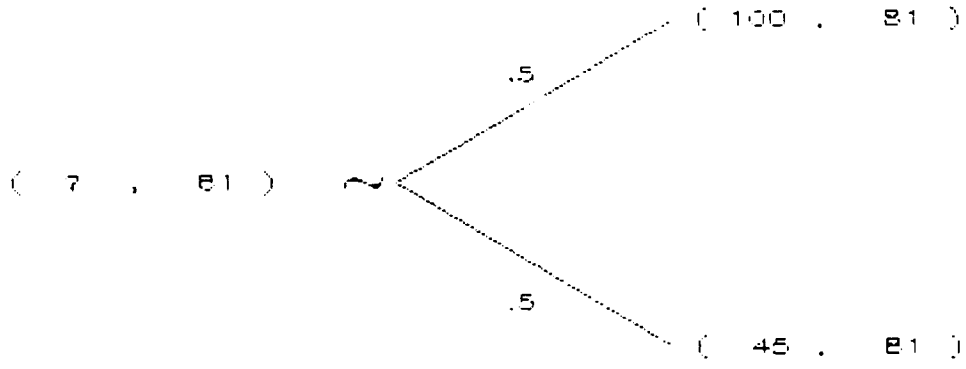
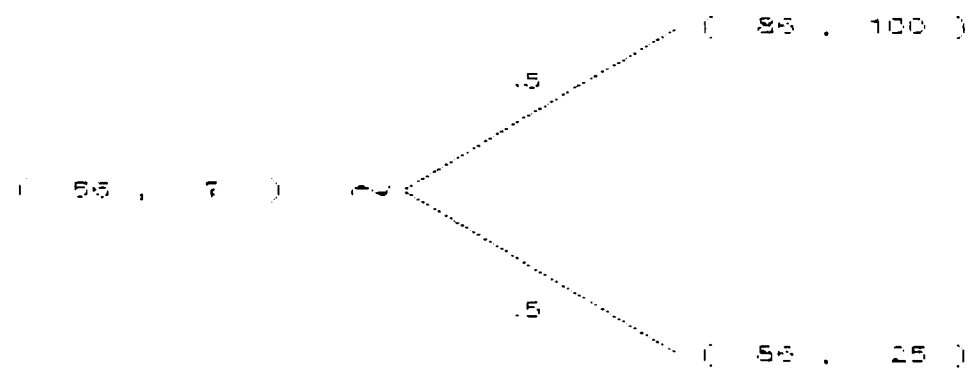


Figure 4.11 Lottery for Pd at $Pa_{.75}$ (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 50



least preferred alternative as the certainty equivalent, because they said they'd already indicated it was the minimum acceptable amount. We had to reemphasize the nature of the lottery being offered. We told them the worst they could do was get the least preferred alternative, so how much more would they have to get for certain before they were willing to give up a 50% shot at the most preferred alternative. Stated this way, they quickly realized the intent of he lottery. Major Case's data indicated that Pd was utility independent of Pa (fig. 4.12).

Assess Conditional Utility Functions and Obtain Scaling Constants.

Three points were elicited to develop the conditional functions and scale the utility values (fig. 4.13 - 4.15). After this, the eight different views of the utility graph were plotted for him to see (fig. 4.16 - 4.23).

Consistency Checking.

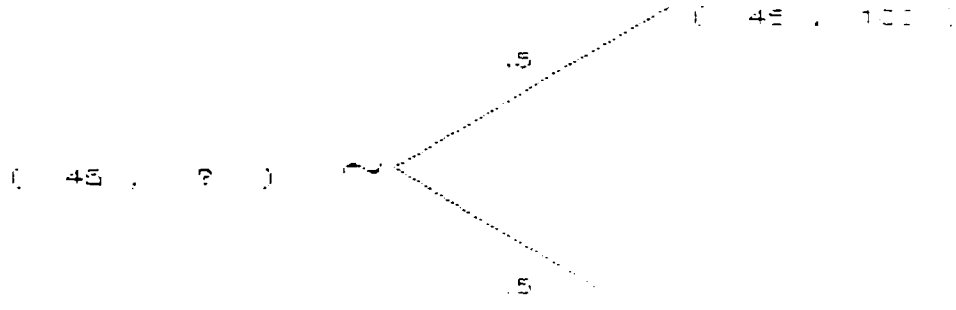
Major Case felt more comfortable being able to express his preferences in words (as did Major Stan). He was uncomfortable with some of his answers to the lotteries, but was fairly confident the graphs showed his intentions to give values for Pd without being influenced by Pa. Specific comments on the technique by Major Case are included in section 4.2.4.

Figure 4.12 Utility Independence One Way Indicated (MP 1)

```
Since your responses indicate that PD          are
utility independent of PA          ,
your utility function can easily be assessed by developing
three marginal utility curves for the attributes.  To do this,
you will be shown additional lotteries for which you must
enter your certainty equivalents.


Hit any key followed by return to continue:
```

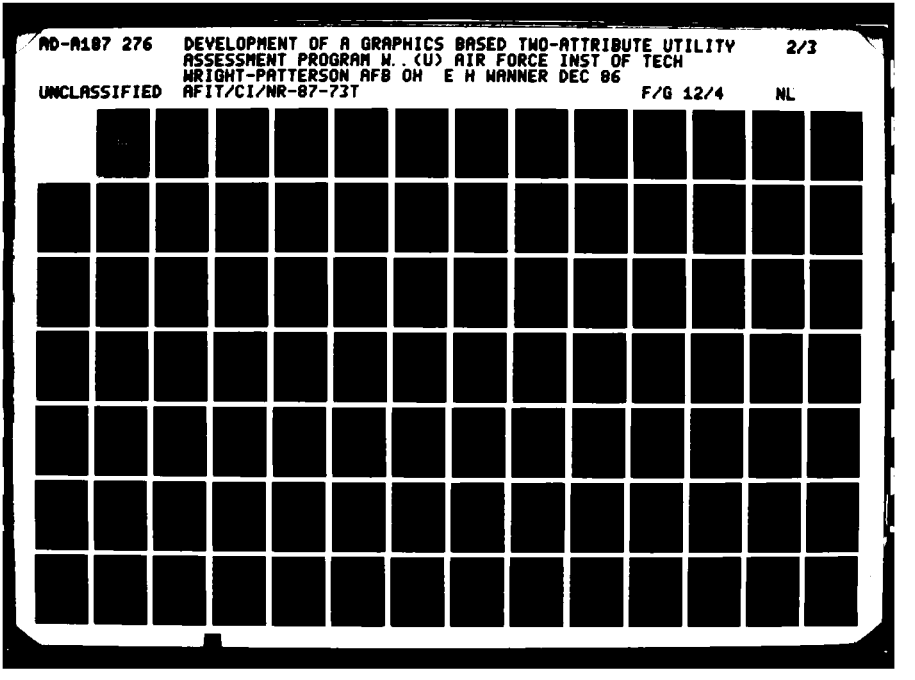
Figure 4.13 C.E. for Pd Marginal Utility Curve Determination (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the "?": 50



AD-A187 276 DEVELOPMENT OF A GRAPHICS BASED TWO-ATTRIBUTE UTILITY ASSESSMENT PROGRAM W..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH E H WANNER DEC 86 2/3

UNCLASSIFIED AFIT/CI/NR-87-73T F/G 12/4 NL



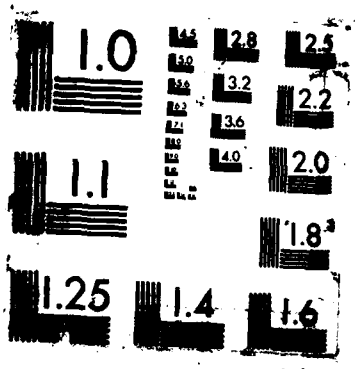

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

Figure 4.14 C.E. for Pa Marginal Utility Curve Determination (at Pd_{min}) (MP 1)

In terms of the (PA , PD) outcome,
please enter your value for the "?": 75

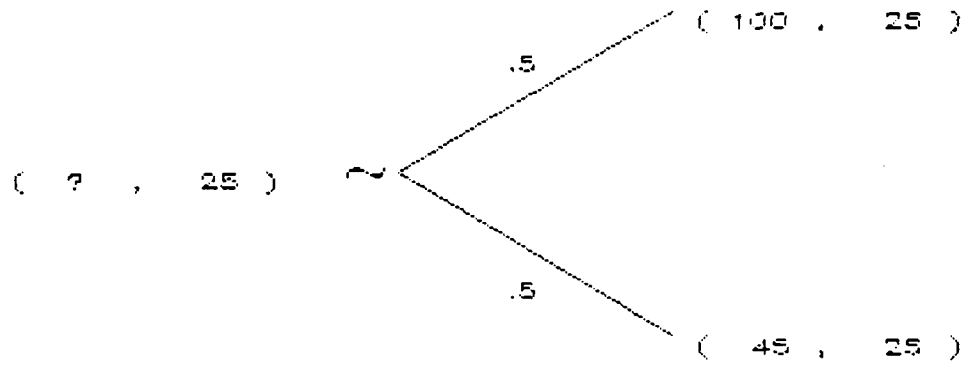


Figure 4.15 C.E. for Pa Marginal Utility Curve Determination (at Pd_{max}) (MP 1)

```
In terms of the (PA        , PD        ) outcome,
please enter your value for the "?": 50
```

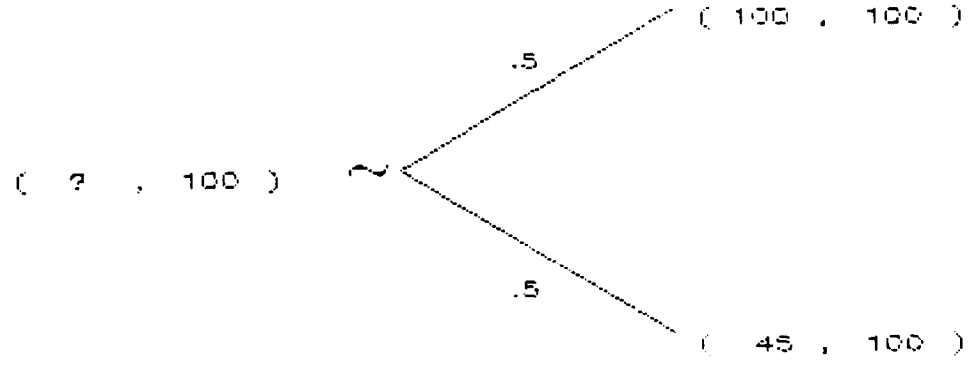


Figure 4.16 Viewpoint 1 (MP 1)

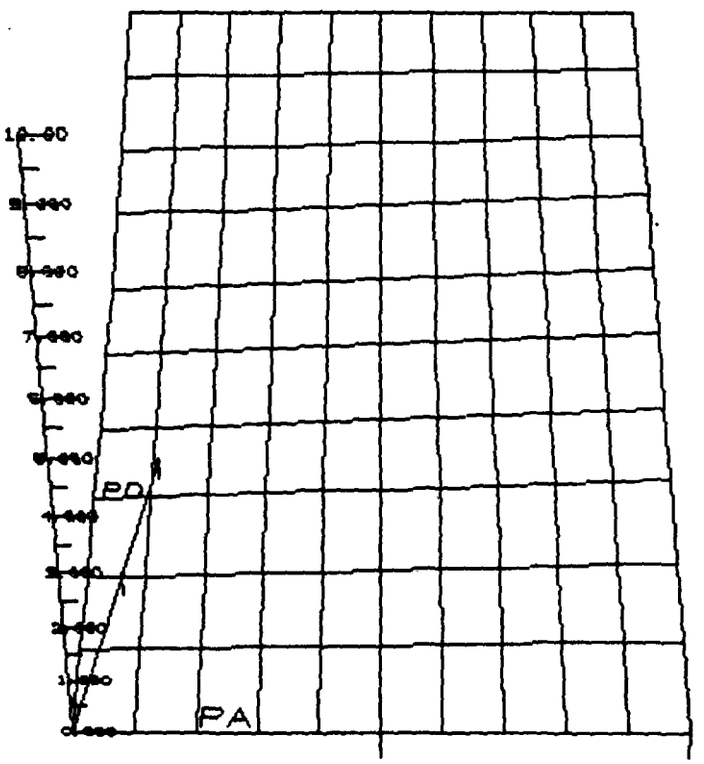


Figure 4.17 Viewpoint 2 (MP 1)

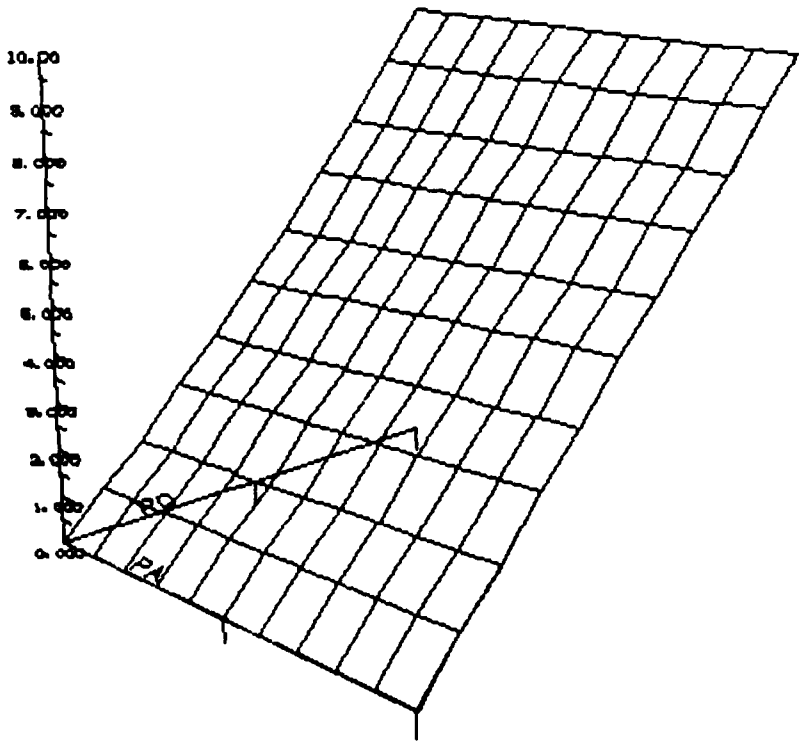


Figure 4.18 Viewpoint 3 (MP 1)

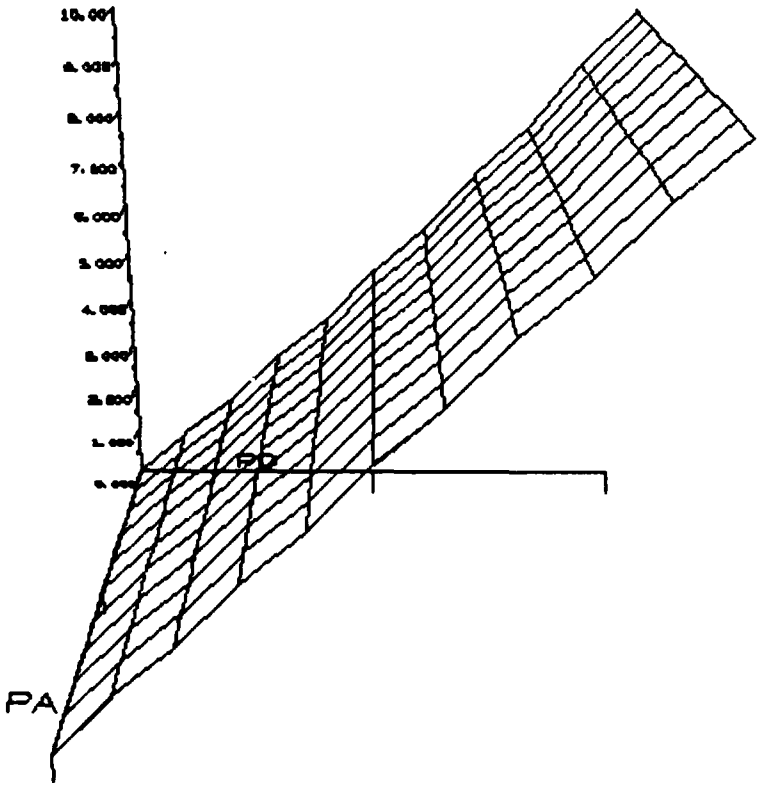


Figure 4.19 Viewpoint 4 (MP 1)

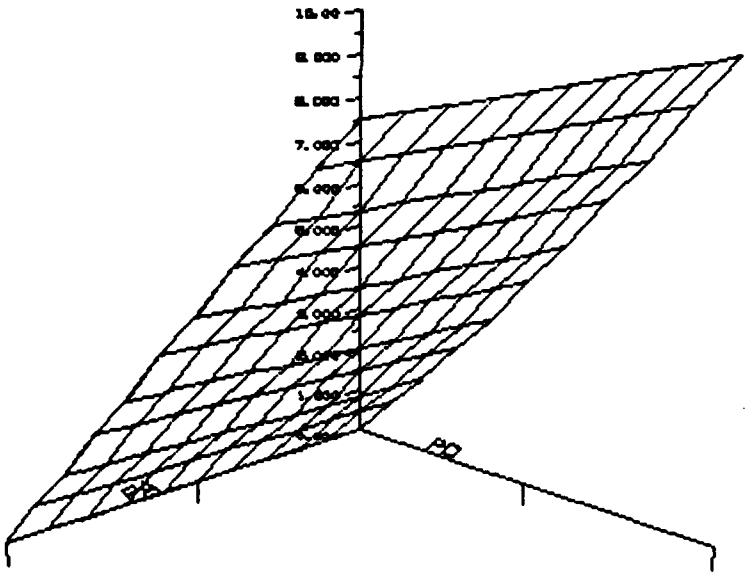


Figure 4.20 Viewpoint 5 (MP 1)

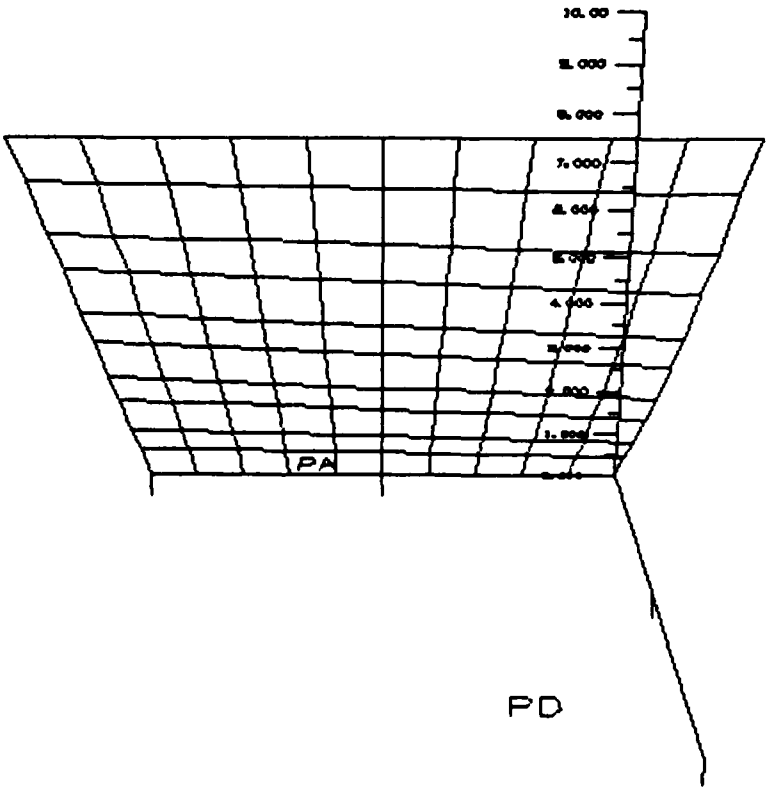


Figure 4.21 Viewpoint 6 (MP 1)

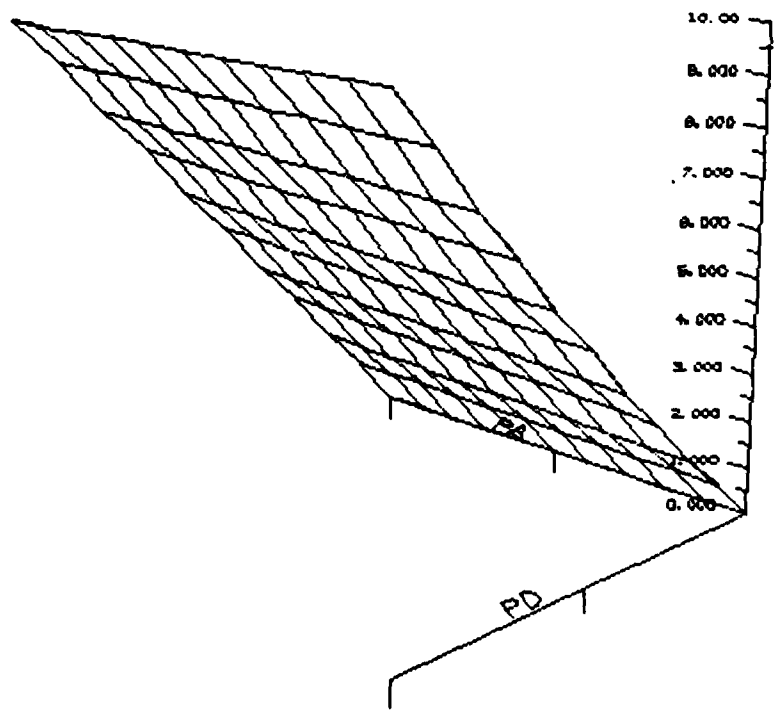


Figure 4.22 Viewpoint 7 (MP 1)

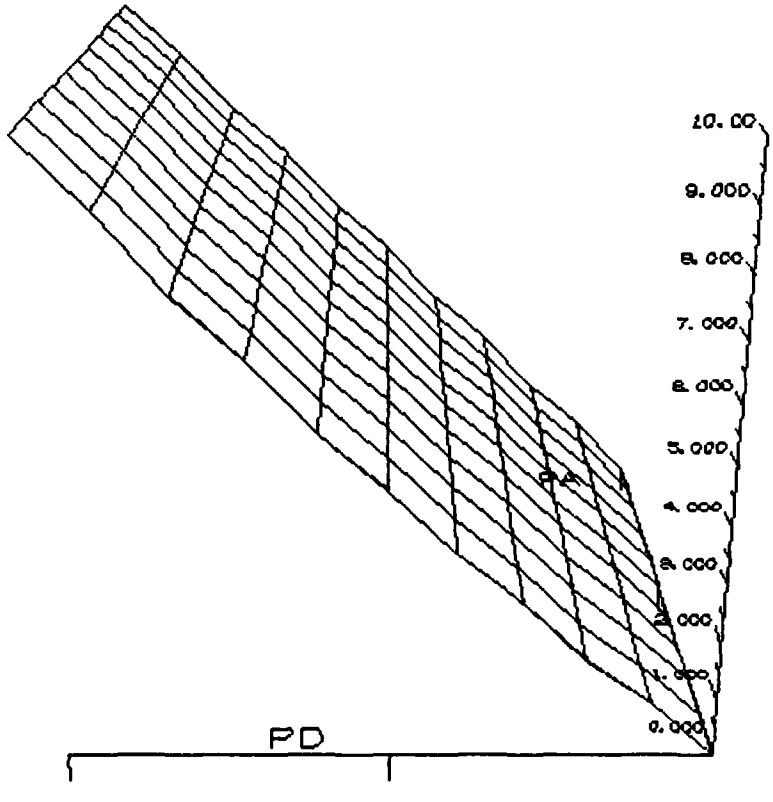
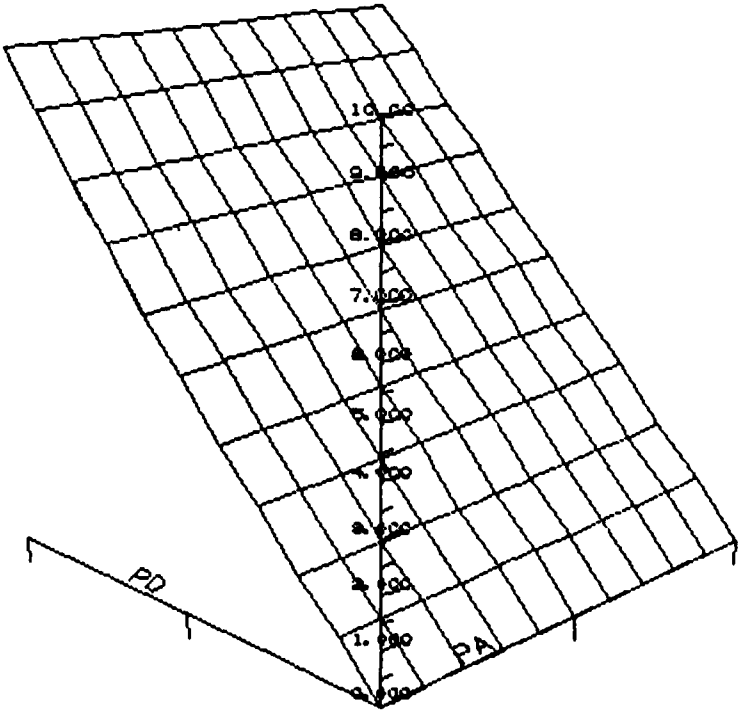


Figure 4.23 Viewpoint 8 (MP 1)



4.2.3 Utility Assessment for Mission Planner 2

Major Stan felt uncomfortable with the idea of setting limits for Pa and Pd without knowing the specific mission, as well. We asked him to think in terms of a specific mission type as well, which made him feel better. His least preferred consequences in terms of (Pa, Pd) was (60, 50) and most preferred was (100, 100), (fig. 4.24).

Identify Relevant Independence Assumptions.

Major Stan's answers to the first ten lotteries (fig. 4.25 - 4.34) showed a possible inconsistency in his certainty equivalents for Pd (fig. 4.35). When asked by the program if this was intentional after being shown his responses for the attribute certainty equivalents, he felt it was; since at higher values of Pa his feelings about Pd changed (fig. 4.16). This resulted in no utility independence properties being present for the attributes (fig 4.37). He chose to try to decompose the attributes, and after viewing his original responses (fig. 4.38 - 4.40) saw that Pa remained constant over all lower values of Pd and only went down once the most preferred consequence of Pd was reached. So, he indicated he'd like to perform the analysis over a subset of the ranges (fig. 4.41).

This time, his consequence space for Pd changed. The least preferred outcome remained constant, but he felt that his preferences for Pa would only remain constant while Pd was 90% or less. Therefore, his utility was assessed over

Figure 4.24 Attribute Information (1st try, MP 2)

```
   Please enter your first attribute.  If the attribute is over
10 characters long, please abbreviate it to be 10 characters.

attribute name: PARRIVE

What units is this attribute measured in?: PA

  Please enter the least and most preferred values this attribute
can be -- (to the nearest whole number).

least preferred: 60
most preferred: 100

   Please enter your second attribute.  Again, please insure it
does not exceed 10 characters in length.

attribute name: PDESTROY

What units is this attribute measured in?: PD

  Please enter the least and most preferred values this attribute
can have -- (to the nearest whole number).

least preferred: 50
most preferred: 100
```

Figure 4.25 Lottery for Pa at Pd_{min} (1st try, MP2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 80

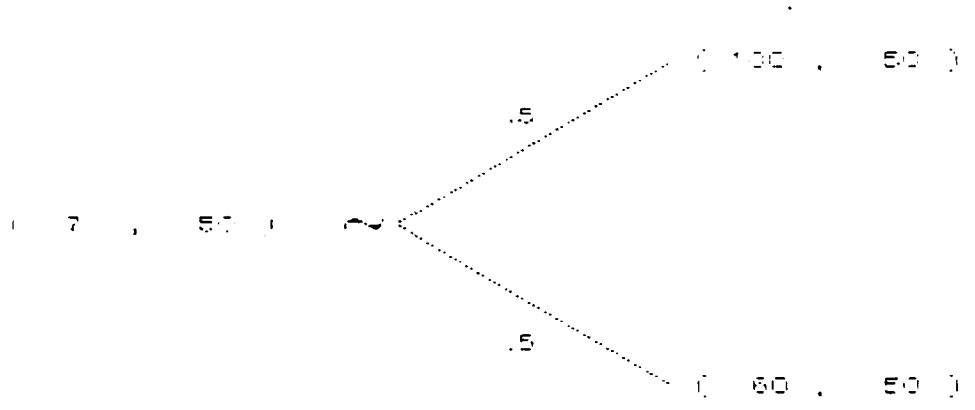


Figure 4.26 Lottery for Pa at $Pd_{.50}$ (1st try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 80

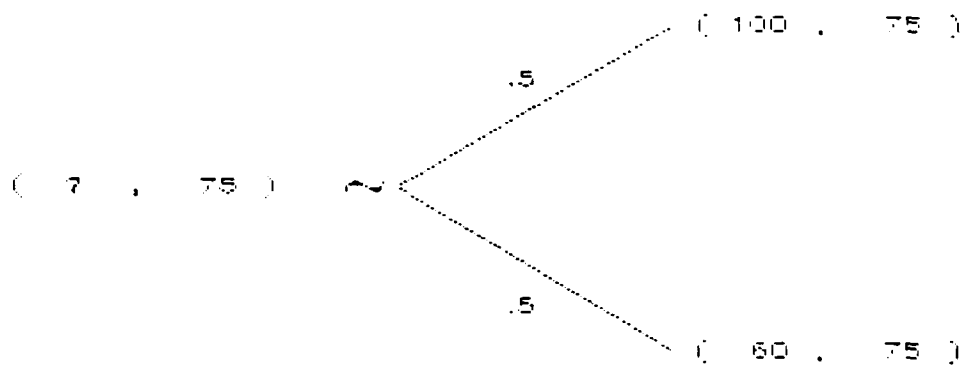


Figure 4.27 Lottery for Pd at Pa_{max} (1st try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 88

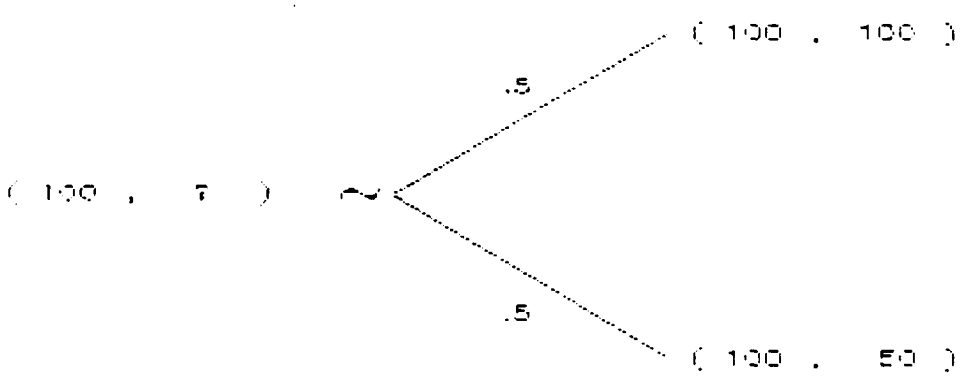


Figure 4.28 Lottery for Pa at $Pd_{.25}$ (1st try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 80

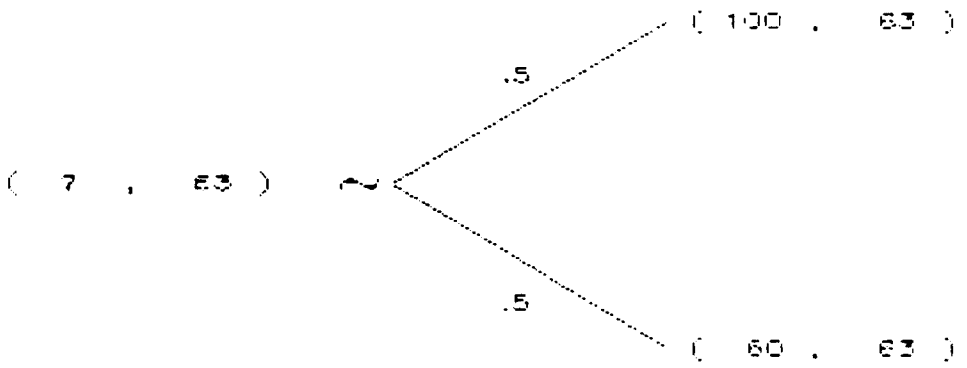


Figure 4.29 Lottery for Pd at $Pa_{.50}$ (1st try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 88

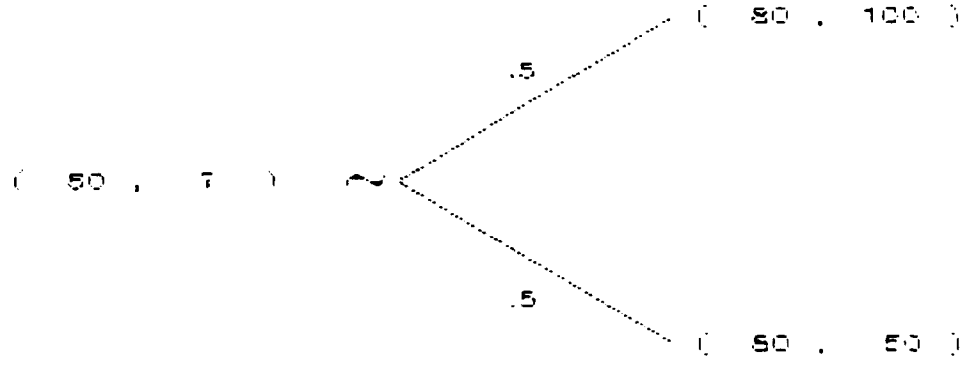


Figure 4.30 Lottery for Pd at $Pa_{.25}$ (1st try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 95

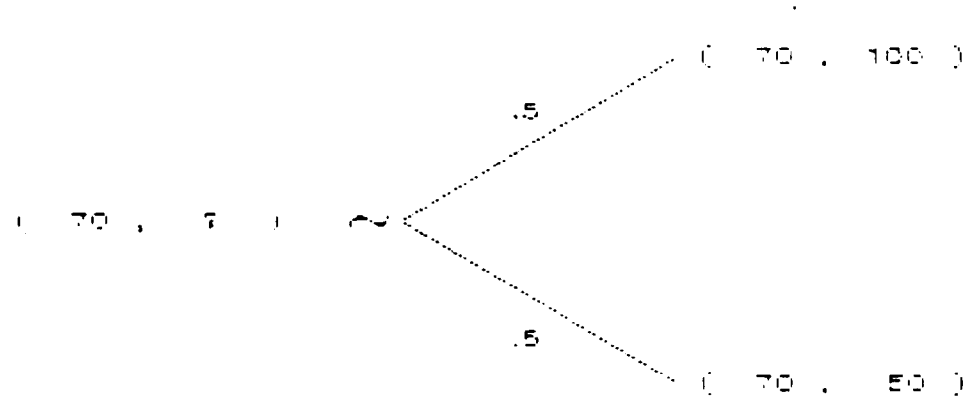


Figure 4.31 Lottery for Pa at Pd_{max} (1st try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 70

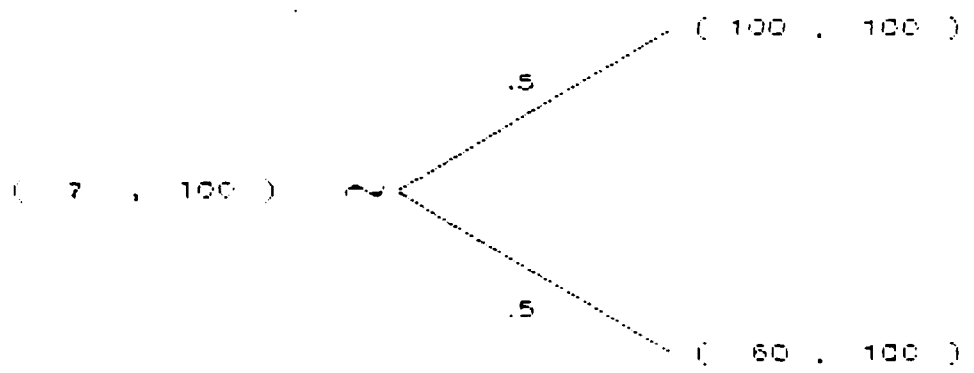


Figure 4.32 Lottery for Pd at Pa_{min} (1st try, MP 2)

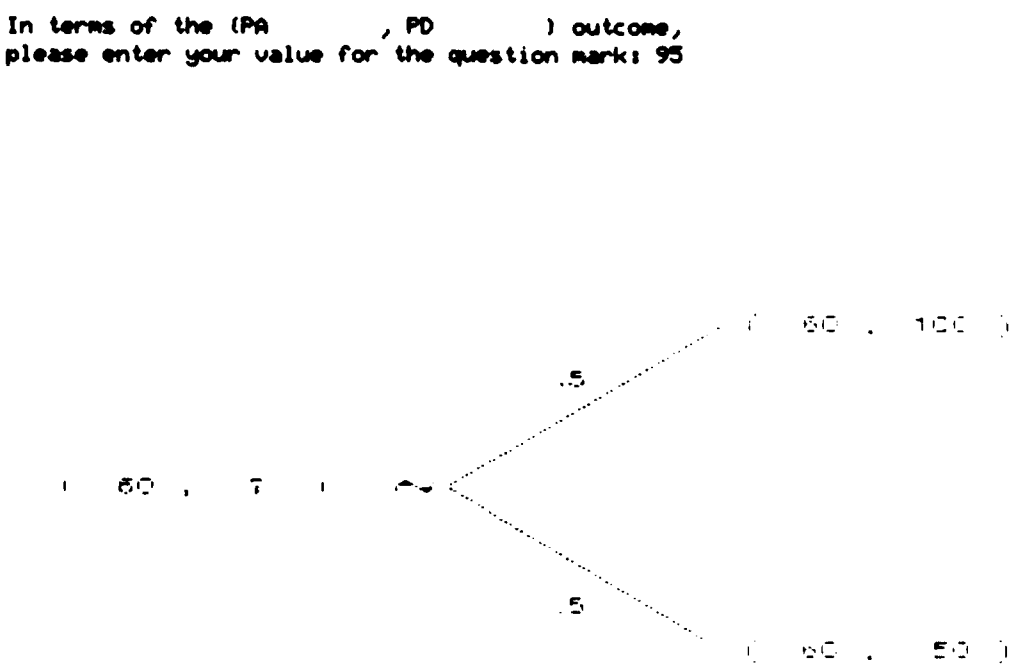


Figure 4.33 Lottery for Pa at $Pd_{.75}$ (1st try, MP 2)

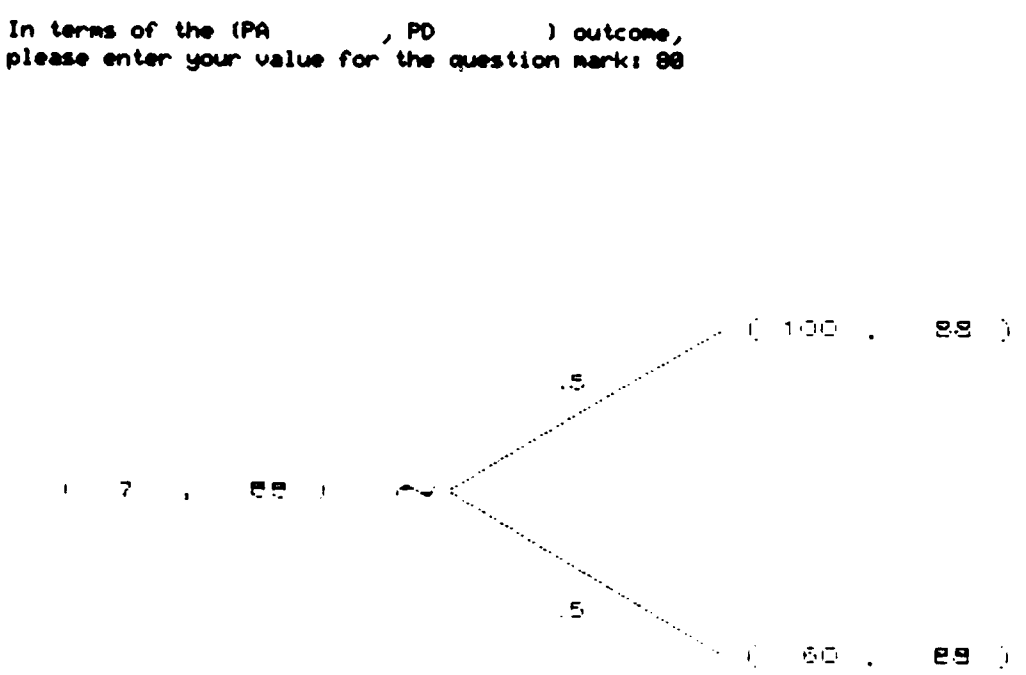


Figure 4.34 Lottery for Pd at $Pa_{.75}$ (1st try, MP 2)

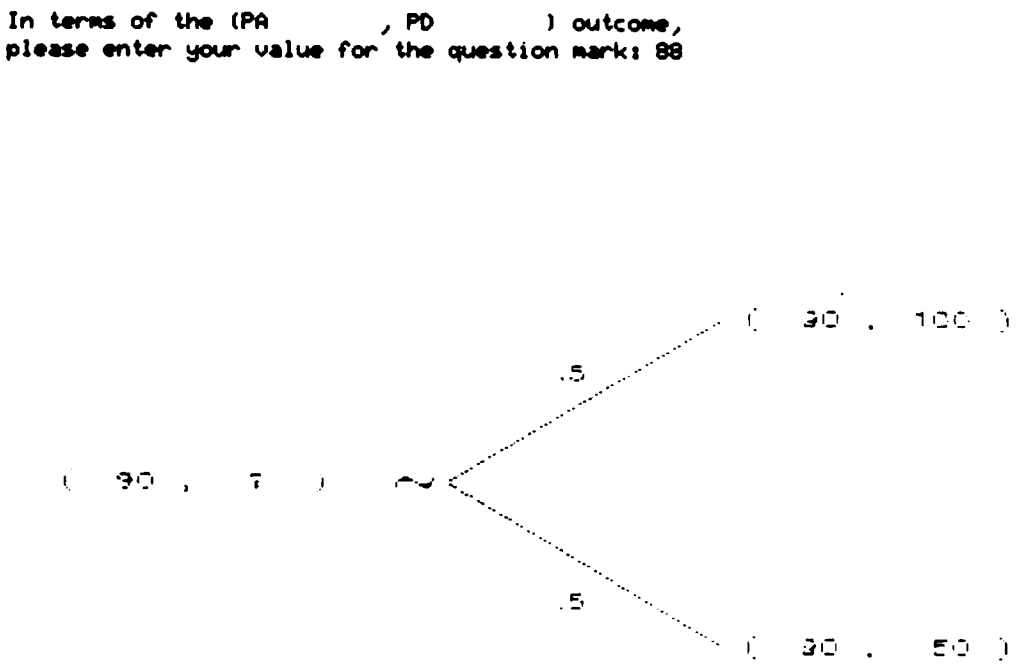


Figure 4.35 Possible Inconsistency Indicated (1st try, MP 2)

In terms of the (PA , PD) outcome, since
your responses for PDESTROY all fall within 5 percent of the average,
please carefully reconsider your previous answers. You will
need to determine if the attribute values should be changed to a
common value, and if so, which one. (NEXT SCREEN)

Hit any key followed by return to continue:

Figure 4.36 Previous Responses for Pd C.E.'s (1st try, MP 2)

```
( 100,   88) ~ [0.5( 100,  100) ; 0.5( 100,   50)]

(  80,   88) ~ [0.5(  80,  100) ; 0.5(  80,   50)]

(  70,   95) ~ [0.5(  70,  100) ; 0.5(  70,   50)]

(  60,   95) ~ [0.5(  60,  100) ; 0.5(  60,   50)]

(  90,   88) ~ [0.5(  90,  100) ; 0.5(  90,   50)]

Please enter the number corresponding to your choice.
  1) Change previous answers to a common one.
  2) Leave answers as is.
Choice: 2
```

Figure 4.37 No Utility Independence Indicated (1st try, MP 2)

```
Since your responses indicate that the attributes are not at all
utility independent, there are three possible approaches to
assessing utility values for the attributes. The first approach
requires you to rerun the utility program using transformed
attributes which may exhibit utility independence. The second
approach requires decomposing the attribute ranges and
then assessing the utility function over these decomposed
ranges, which may be utility independent. Further guidance on
this option is provided when you choose it. Finally, the
last approach requires you to indicate your utilities directly
for different combinations of attributes. Since this is the
most difficult option to perform, it is recommended you try
one of the other two if possible.


Please indicate which option you want.
1) Use transformed attributes.
2) Decompose the attribute ranges.
3) Direct assessment.
Choice: 2
```

Figure 4.38 Decomposition of Ranges Chosen (1st try, MP 2)

```
   Decomposing the attribute ranges should be done if there
is indication of utility independence over a subset of the range.
This is indicated by having a constant certainty equivalent for
an attribute even when the other attribute's value varies.  For
example, if your certainty equivalent for attribute one (ranging
from, say, 0 to 100) is always constant (say, 60) over a subrange
(say, less than 350) of attribute two (ranging from, say, 200 to
400), then a good place to subset the range would be from 200 to
350 and 350 to 400.  This way, over the 200 to 350 range, some
utility independence properties will hold.

Would you like to see your original answers again?
1) yes
2) no
Choice: 1
```

Figure 4.39 Previous Responses for Pa C.E.'s (1st try, MP 2)

```
These are the choices made for PARRIVE
by varying PDESTROY   in terms of the
(PA        , PD        ) outcome:

(  30,   50) ~ [0.5( 100,   50) ; 0.5(  60,   50)]

(  80,   75) ~ [0.5( 100,   75) ; 0.5(  60,   75)]

(  80,   63) ~ [0.5( 100,   63) ; 0.5(  60,   63)]

(  70,  100) ~ [0.5( 100,  100) ; 0.5(  60,  100)]

(  80,   88) ~ [0.5( 100,   88) ; 0.5(  60,   88)]

Hit any key followed by return to continue:
```

Figure 4.40 Previous Responses for Pd C.E.'s (1st try, MP 2)

```
These are the choices made for PDESTROY
by varying PARRIVE    in terms of the
(PA        , PD        ) outcome:

( 100,   88) ~ [0.5( 100,  100) ; 0.5( 100,   50)]

(  80,   88) ~ [0.5(  80,  100) ; 0.5(  80,   50)]

(  70,   95) ~ [0.5(  70,  100) ; 0.5(  70,   50)]

(  60,   95) ~ [0.5(  60,  100) ; 0.5(  60,   50)]

(  90,   88) ~ [0.5(  90,  100) ; 0.5(  90,   50)]

Hit any key followed by return to continue:
```

Figure 4.41 Decomposition of Ranges Rechosen (1st try, MP 2)

```
Please indicate which operation you want to perform.
1) Return to options menu.
2) Perform the analysis over a subset of ranges.
Choice: 2
```

this lesser range (fig. 4.42). Now his responses to the ten lotteries (fig. 4.43 - 4.52) indicated that Pa was utility independent of Pd (fig. 4.53).

Assess Conditional Utility Functions and Obtain Scaling Constants.

After answering the questions used to build the conditional utility curves (fig. 4.54 - 4.56), his graphs were presented for him to see (fig. 4.57 - 4.64). He was curious about how they could be used in an actual problem. We explained how they could be compared between mission planners and actual mission success to see which was the best representation of utility, or how they could be continually reassessed over time, depending on changing conditions, and the progression of a decision maker's preferences over time could be traced. Finally, the graph represents information which will tell a decision maker immediately if he prefers an outcome of (70, 80) or (75, 75).

Consistency Checking.

Major Stan felt good about the program's ability to reflect his preferences. However, it was difficult to ask specific questions about preferring particular outcomes which were close together on the utility graph because of a perspective distortion. This is a major critique item in the next section.

Figure 4.42 Attribute Information (2nd try, MP 2)

```
   Please enter your first attribute.  If the attribute is over
10 characters long, please abbreviate it to be 10 characters.

attribute name: PARRIVE

What units is this attribute measured in?: PA

  Please enter the least and most preferred values this attribute
can be -- (to the nearest whole number).

least preferred: 60
most preferred: 100

   Please enter your second attribute.  Again, please insure it
does not exceed 10 characters in length.

attribute name: PDESTROY

What units is this attribute measured in?: PD

  Please enter the least and most preferred values this attribute
can have -- (to the nearest whole number).

least preferred: 50
most preferred: 90
Hit any key followed by return to continue:
```

Figure 4.43 Lottery for Pa at Pd_{min} (2nd try, MP 2)

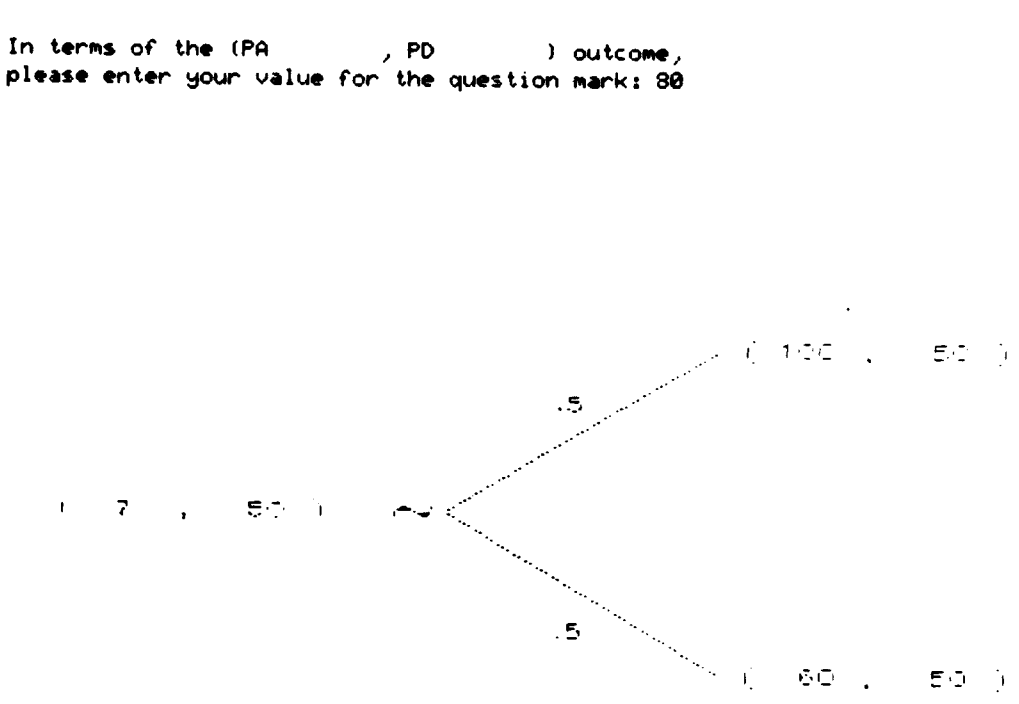


Figure 4.44 Lottery for Pa at Pd_{min} (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 80

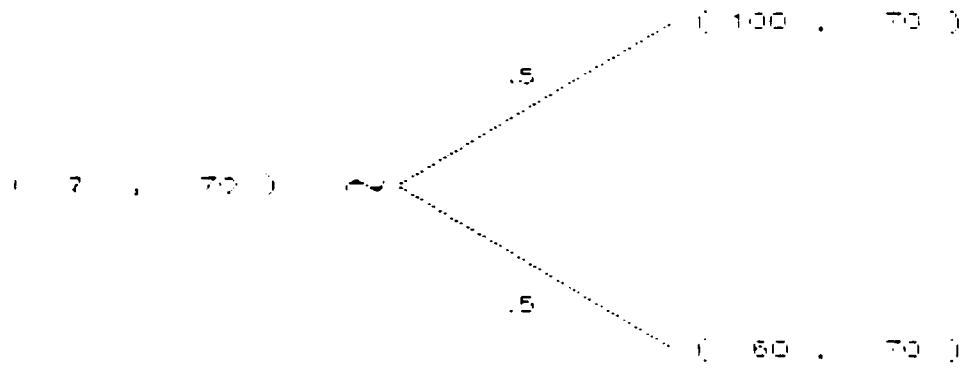


Figure 4.45 Lottery for Pd at Pa_{max} (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 78

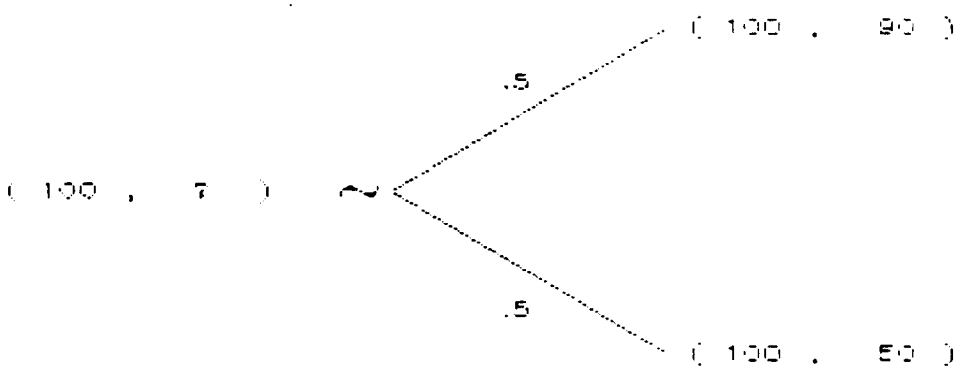


Figure 4.46 Lottery for Pa at $Pd_{.25}$ (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 80

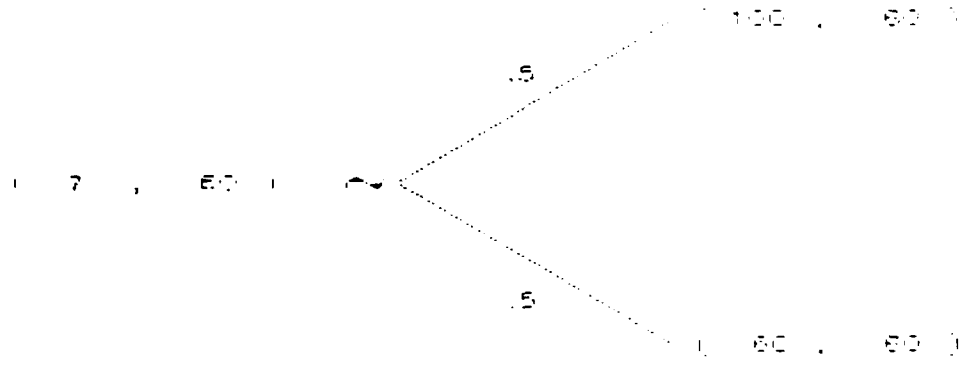


Figure 4.47 Lottery for Pd at $Pa_{.50}$ (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 78

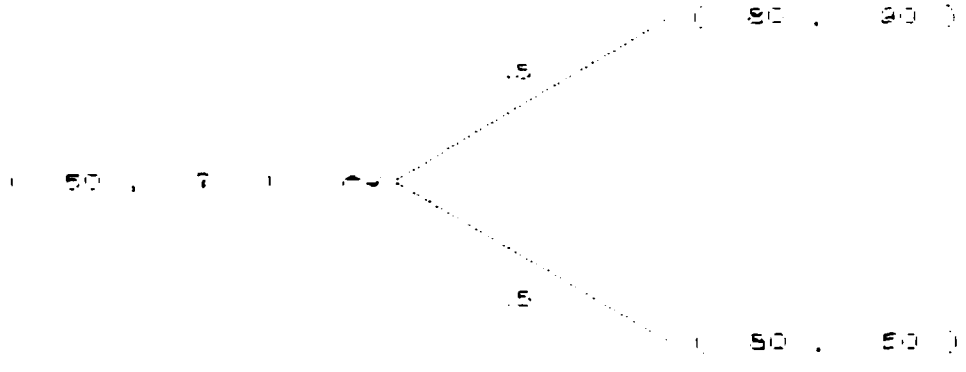


Figure 4.48 Lottery for Pd at $Pa_{.25}$ (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 88

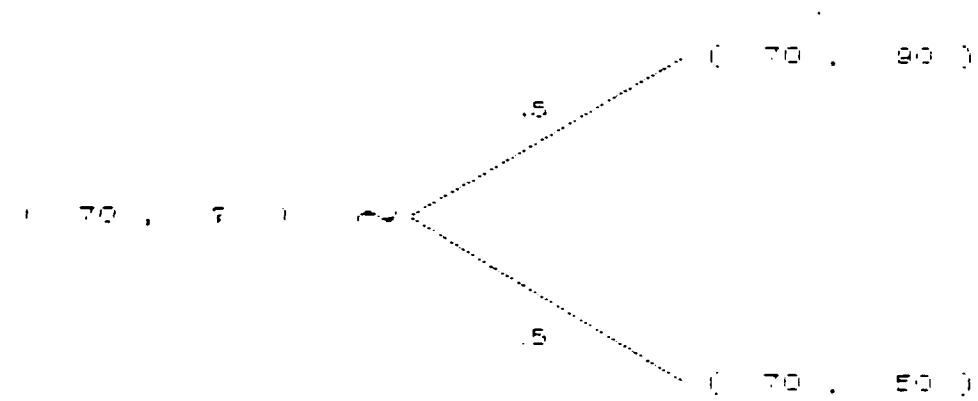


Figure 4.49 Lottery for Pa at Pd_{max} (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 80

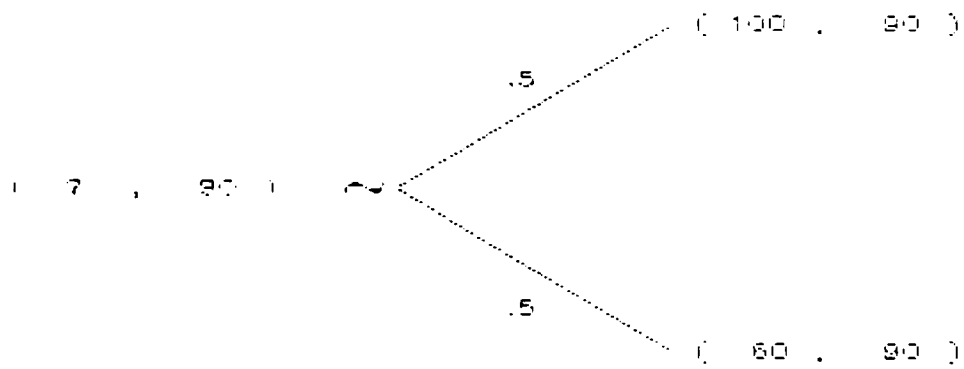


Figure 4.50 Lottery for Pd at Pa_{min} (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 85

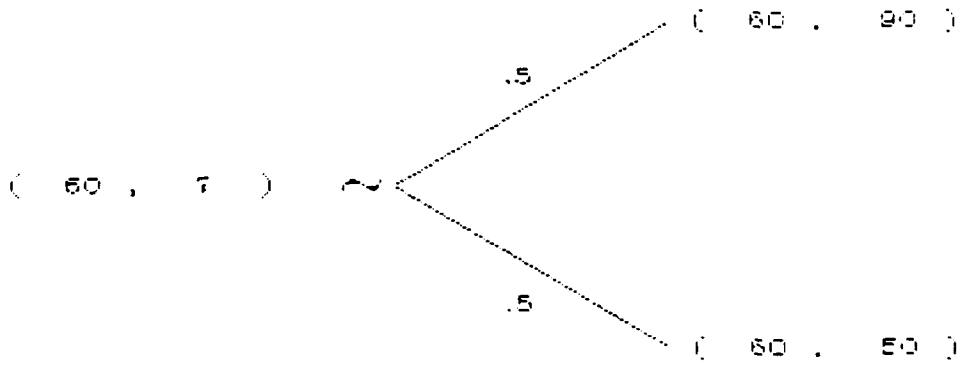


Figure 4.51 Lottery for Pa at $Pd_{.75}$ (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 80

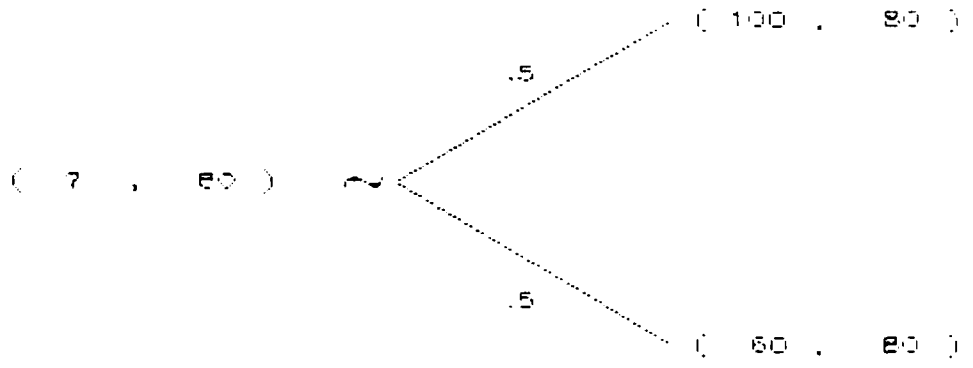


Figure 4.52 Lottery for Pd at $Pa_{.75}$ (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the question mark: 78

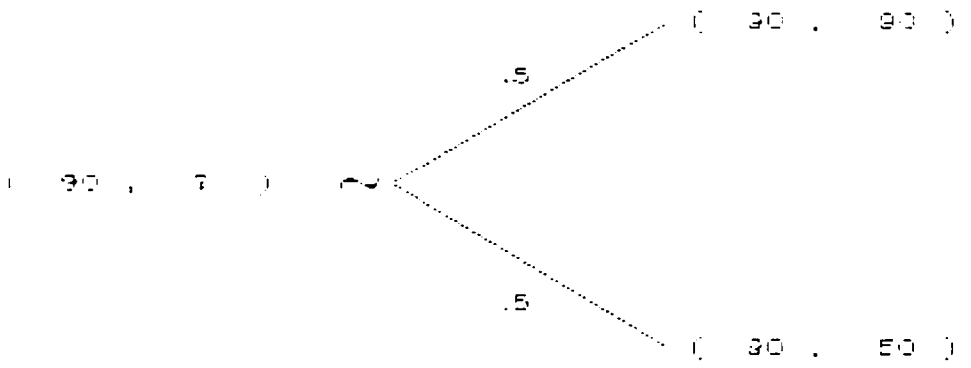


Figure 4.53 Utility Independence One Way Indicated (2nd try, MP 2)

```
Since your responses indicate that PA          are
utility independent of PD          ,
your utility function can easily be assessed by developing
three marginal utility curves for the attributes.  To do this,
you will be shown additional lotteries for which you must
enter your certainty equivalents.


Hit any key followed by return to continue:
```

Figure 4.54 C.E. for Pd Marginal Utility Curve Determination (at Pa_{min}) (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the "?": 80

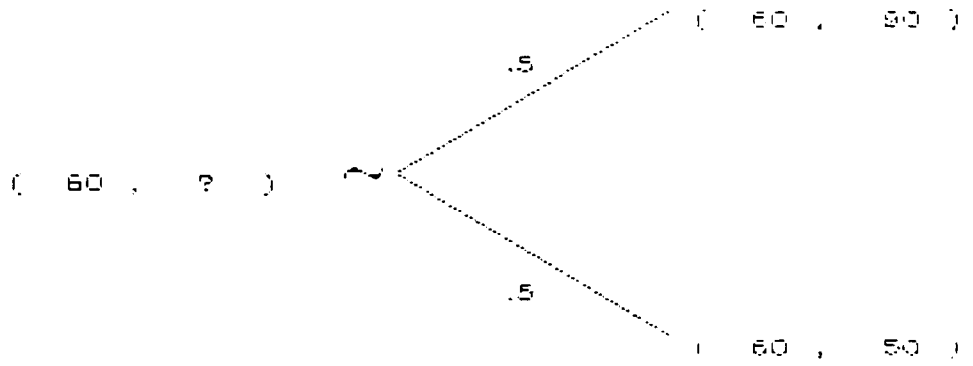


Figure 4.55 C.E. for Pa Marginal Utility Curve Determination (2nd try, MP 2)

```
In terms of the (PA        , PD        ) outcome,
please enter your value for the "?": 80
```

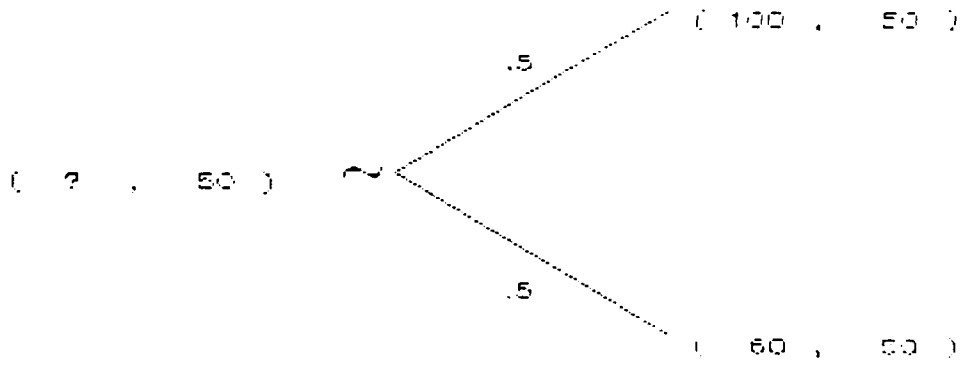


Figure 4.56 C.E. for Pd Marginal Utility Curve Determination (at Pa_{max}) (2nd try, MP 2)

In terms of the (PA , PD) outcome,
please enter your value for the "?": 55

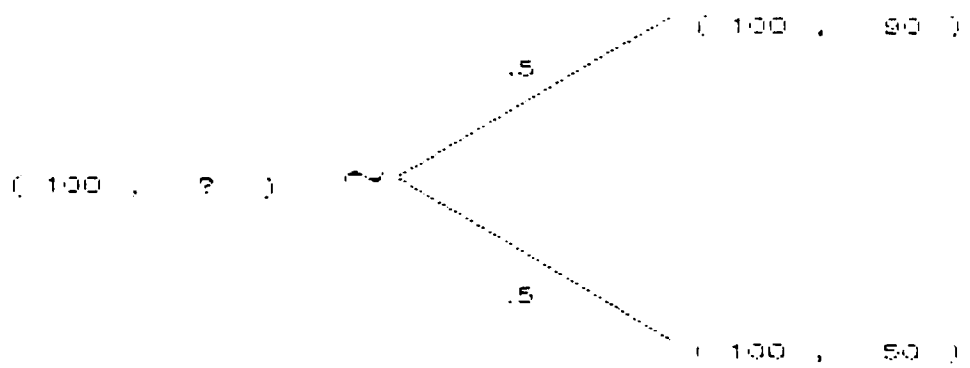


Figure 4.57 Viewpoint 1 (MP 2)

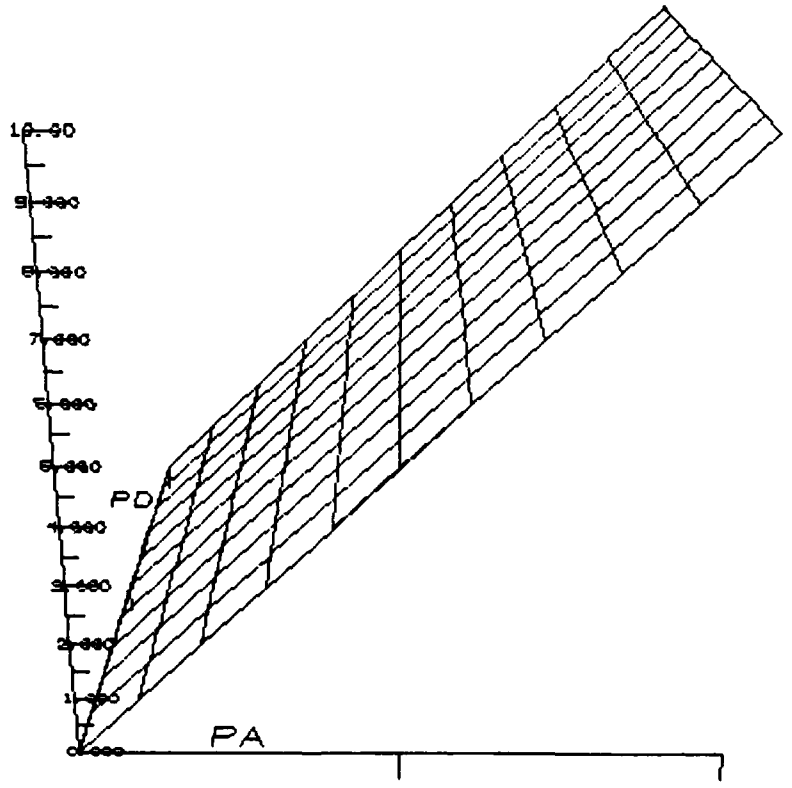


Figure 4.58 Viewpoint 2 (MP 2)

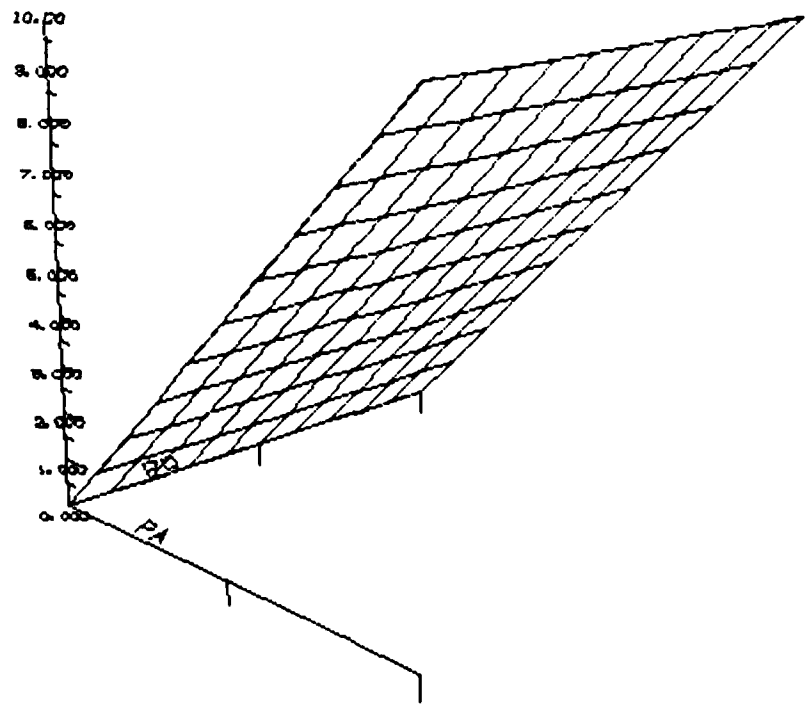


Figure 4.59 Viewpoint 3 (MP 2)

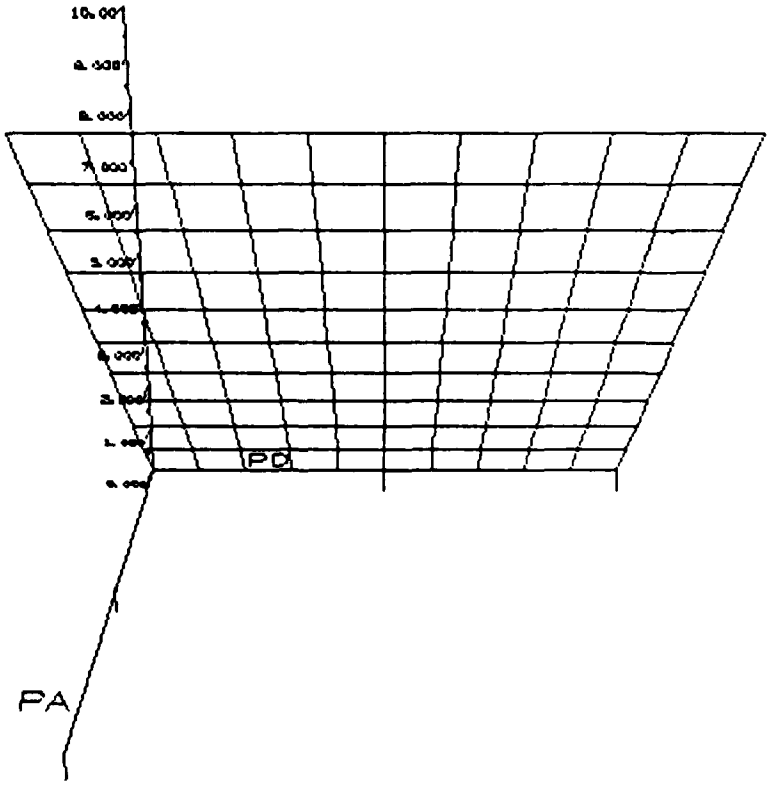


Figure 4.60 Viewpoint 4 (MP 2)

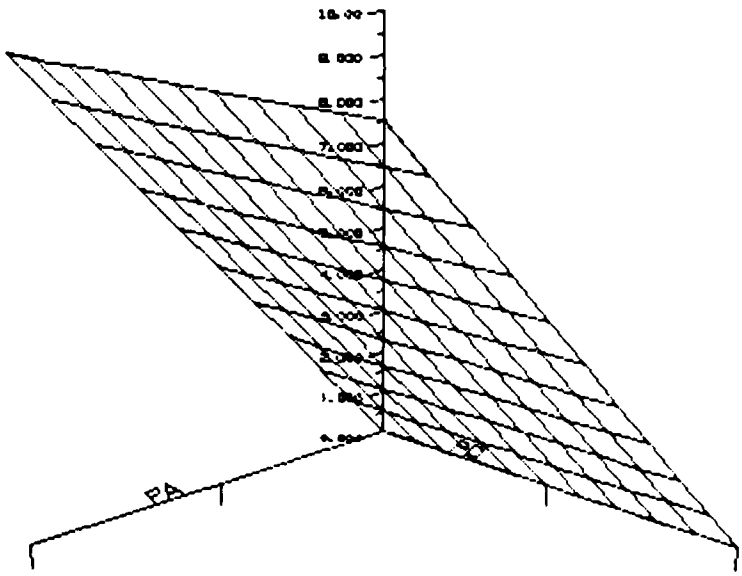


Figure 4.61 Viewpoint 5 (MP 2)

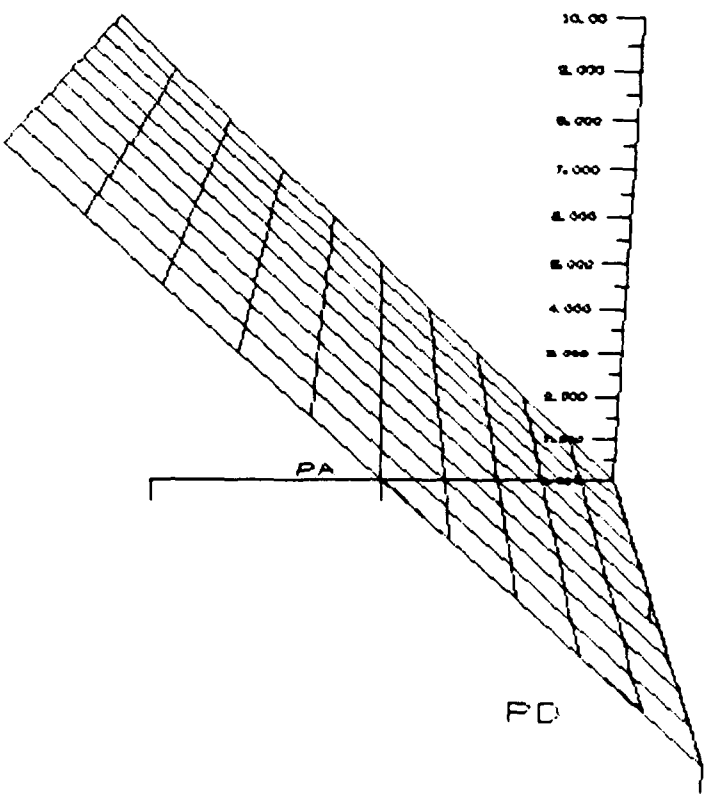


Figure 4.62 Viewpoint 6 (MP 2)

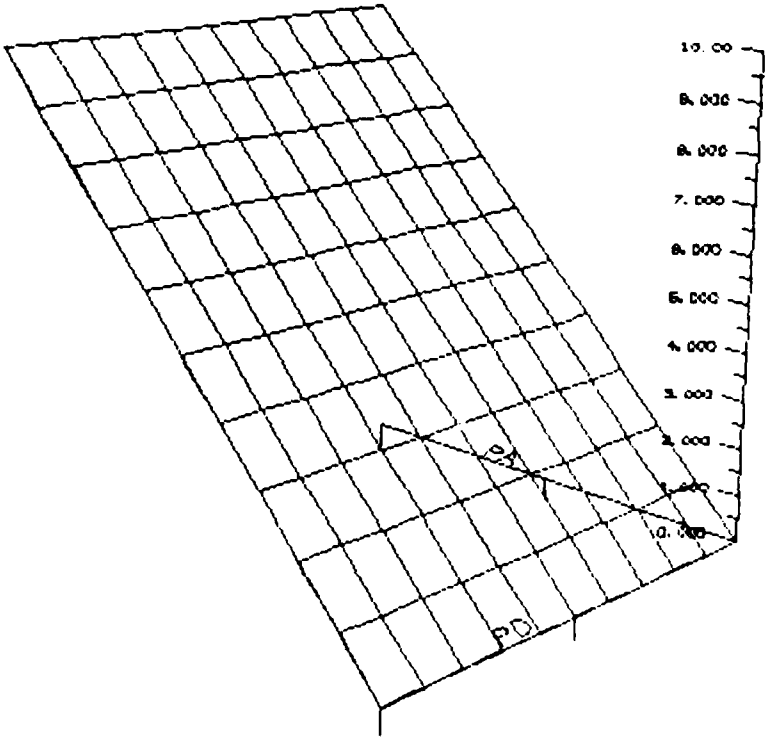


Figure 4.63 Viewpoint 7 (MP 2)

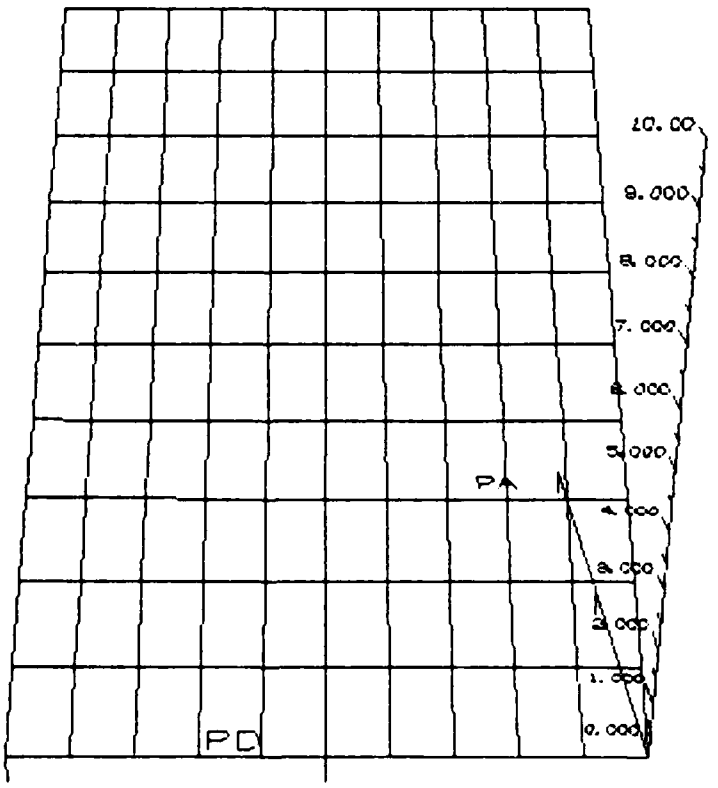
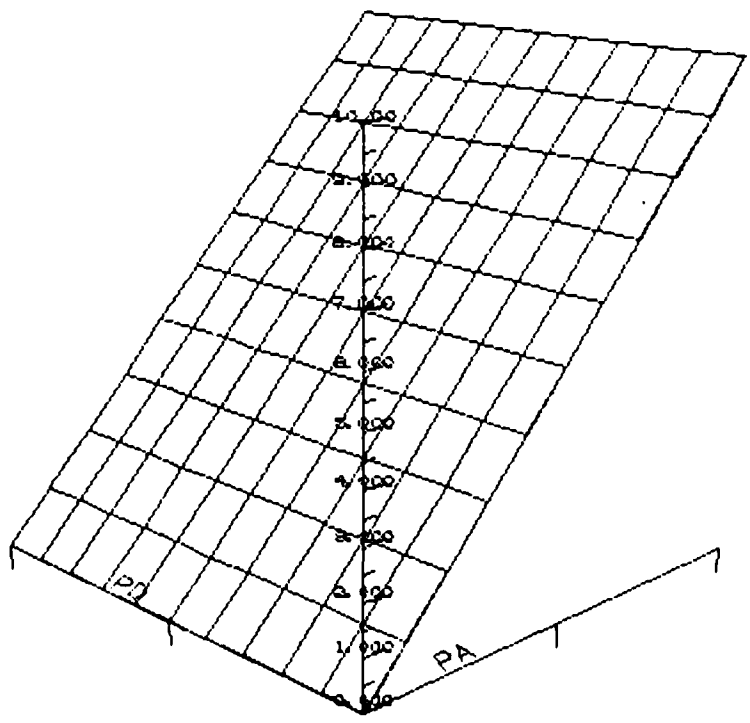


Figure 4.64 Viewpoint 8 (MP 2)



4.2.4 Comments and Critiques

Both officers were impressed with the potential of multi-attribute utility theory. Major Case, in particular, had previously worked at trying to elicit preferences from Tactical Air Force personnel, and was appreciative of the structure the program lent to the process.

They both were concerned, however, over the subjectiveness of the Pa attribute. An accurate model to estimate Pa would have to be developed before reliance could be placed on using it as an attribute.

Both officers commented on how they preferred a technique which was representative of an individual's preferences. It seems some expert systems are being developed which are not powerful enough to assess situations as well as humans, so nobody agrees with them. Multi-attribute utility theory is flexible enough to be used in these instances since rather than tell a decision maker what to do, it merely maps his preferences for him.

The biggest problem, however, was one of information portrayal. If, for example, the graphs had been presented on a touch screen, it would have been possible to map points on the graph back to a specific utility. As it is now, perspective effects make it difficult to find actual utility value points.

5. CONCLUSION

During the development of, and application of, the utility assessment graphics program, a number of areas were uncovered which need modification or expansion. Even so, we feel the program has merit as a tool for practical problems of choice involving only two attributes.

5.1 Extensions and Areas for Further Research

Based on our experience in developing and applying the two-attribute utility program, the following areas are those further work is needed.

The first extension is more of a modification. During the course of the project, we grew increasingly frustrated by the lack of flexibility of the DI-3000 graphics package (in particular with the contouring package). Because the specific application it was designed for was terrain contouring, it was difficult to accomplish many things a more generalized graphics package would provide. Also, as mentioned earlier, hardware which allows user interaction (such as a touch screen, mouse, or joystick) for finding utility values would prevent the perspective distortion effects.

As for the program itself, its major areas of additional work required are in the incorporation of more pre-specified utility functions and consistency checking. A consistency check for decision makers forced to use the

direct assessment option is desperately needed since the process is so difficult to perform. A consistency checking routine after the program is completed would be a nice feature also. Since we mentioned the difficulty of direct assessment, we shall reiterate the need for a procedure which helps the decision maker focus in on the two-attribute certainty equivalents that are required for scaling the attributes.

There are also problems with the scale of the utility axis on the utility graphs. They range from 0 to 10 instead of from 0 to 1. This was due to the automatic labeling function of the Contouring Package. Any future work should insure the axis is labeled according to the 0 to 1 scale.

As a final extension, we would recommend the incorporation of fuzzy attributes into the consequences. Particularly in the case of mission planning, the use of a fuzzy number for Pa is much more realistic. Good articles on the topic of fuzzy decision analysis are (Litchfield 1976) and (Tong 1979).

5.2 Concluding Remarks

By now the benefits of multi-attribute utility theory, even limited to the two-attribute case, are evident. The ability to break a problem down into smaller ones has long been advocated as a way of making it easier to solve (i.e., divide and conquer). In the same way, the use of this

methodology to develop a decision maker's utility is made simpler by the exploitation of the four possible utility independence properties. As is shown by the application of even the simple program developed here to a real world problem, multi-attribute utility theory forces a decision maker to view a problem in the structured, consistent fashion necessary to help him achieve a set of consequences and their respective utilities. Once he has these utilities, to make the best decision, he has only to maximize expected utility to determine which alternatives to choose.

In regard to what this project has meant to the author; it has been a progression towards a fuller understanding of the range and depth of utility theory This project has deepened her perception of the field of decision analysis in general, and has been a natural extension to her previous work in the Decision Aids Section of RADC. As an Air Force officer, being able to see the results of applying the utility assessment graphics program to a real world problem has been rewarding and the knowledge gained from this experience can be used in future assignments. Finally, from a computer systems standpoint, the ability to learn a new editor and operating system (VAX 11/780), as well as using a graphics package with which no previous experience was had, has given the author an advantage in her next assignment since their use will be required.

REFERENCES

Bolinger, J.J., Ghose, P., Sosinski, J.H., and Esser, W.F. 1978. Decision Analysis Utilizing Multi Attribute Utility Theory in Engineering Evaluations. IEEE Trans PAS. Vol 97 No 4:1245-1253.

Howard, R.A. 1968. The Foundations of Decision Analysis. IEEE Trans SSC. Vol 4 No 3:211-219.

Johnson, E.M. and Huber, G.P. 1977. The Technology of Utility Assessment. IEEE Trans SMC. Vol 7 No 5:311-325.

Keeney, R.L. and Nair, K. 1975. Decision Analysis for the Siting of Nuclear Power Plants - The Relevance of Multi Attribute Utility Theory. Proc IEEE. Vol 63 No 3:494-501.

Keeney, R.L. and Raiffa H. 1976. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. New York: John Wiley and Sons.

Litchfield, J.W., Hansen, J.V., and Beck, L.C. 1976. A Research and Development Model Incorporating Utility Theory and Measurement of Social Values. IEEE Trans SMC. p. 400-410.

Sheridan, T.B. and Sicherman, A. 1977. IEEE Trans SMC. Vol 7 No 5:392-394.

Spetzler, C.S. 1968. The Development of a Corporate Risk Policy for Capital Investment Decisions. IEEE Trans SSC. Vol 4 No 3:279-300.

Stimpson, W. 1981. MADAM: Multiple Attribute Decision Analysis Model, Vol I and II. Master's Thesis. Air Force Institute of Technology, WPAFB, OH. AD-A111 104 and 105.

Tong, R.M. and Bonissone, P.P. 1979. Linguistic Decision Analysis Using Fuzzy Sets. Memorandum No UCB/ERL M79/72. Electronics Research Laboratory, College of Engineering, University of California, Berkeley.

Worth, D.W. 1968. A Tutorial Introduction to Decision Theory. IEEE Trans SSC. Vol 4 No 3:200-210.

APPENDIX

Program Listing

```
Program draw(input, output);


label
   restart;

const
   attsize = 10;
   wrdsize = 40;
   stsize = 4;

type
   attribute = packed array[1..attsize] of char;
   wordstr = packed array[1..wrdsize] of char;
   parray = packed array[1..stsize] of char;
   sarray = varying[stsize] of char;
   range = integer;
   lotrecord = record
      attrib : attribute;
      pos : range;
      ans : range;
   end;
   tarray = array[1..10] of lotrecord;
   gtable = packed array[1..11,1..11] of real;
   worldarray = packed array[1..3] of real;
   jstarray = array[1..2] of integer;

var
   i : integer;
   attribute1, attribute2 : attribute;
   minimum1, minimum2, maximum1, maximum2 : range;
   lottable : tarray;
   xdellen, xratmin,
   ydellen, yratmin,
   xdellen, xratmin, yb, zb, yc, zc : real;
   wldmin, wldmax, eye, at, xaxsbeg,
   zaxsend, xdeltik, xdellab, xdelbas,
   xdelpln, xskpval, yaxsbeg,
   yaxsend, ydeltik, ydellab, ydelbas,
   ydelpln, yskpval, xaxsbeg,
   xaxsend, xdeltik, xdellab, xdelbas,
   xdelpln, xskpval : worldarray;
   xlabjst, ylabjst, xlabjst : jstarray;
   xnminor, xlablfg, ynminor, ylablfg,
   xnminor, xlablfg : integer;
   go_on : boolean;
   rerun_option : integer;
   attiscale, att2scale : attribute;


  { These procedures are external to this program.  They are written in }
  { Fortran, and resident in the DI3000 Software Graphics Package under }
  { VAX/VMS. }

procedure JHSTRG(%stdescr wstrg : wordstr); extern;
procedure JFONT(i : integer); extern;
procedure JJUST(i,j : integer); extern;
procedure JSIZE(x,y : real); extern;
procedure JOPEN; extern;
procedure JROPEN(i : integer); extern;
```

```
procedure JRCLOS(i : integer); extern;
procedure JFRAME; extern;
procedure JCOLOR(i : integer); extern;
procedure JLSTYL(i : integer); extern;
procedure JLWIDE(i : integer); extern;
procedure JCLOSE; extern;
procedure JBEGIN; extern;
procedure JDINIT(i : integer); extern;
procedure JDEVON(i : integer); extern;
procedure JVSPAC(w,x,y,z : real); extern;
procedure JVPORT(w,x,y,z : real); extern;
procedure JWINDO(w,x,y,z : real); extern;
procedure JMOVE(x,y : real); extern;
procedure J3MOVE(x,y,z : real); extern;
procedure JPLANE(x,y,z : real); extern;
procedure JBASE(x,y,z : real); extern;
procedure JDRAW(x,y : real); extern;
procedure JPAUSE(i : integer); extern;
procedure JSSINI; extern;
procedure JSSPRO(m, n, e, a : worldarray); extern;
procedure JSSAXS(m, n : worldarray; z : real; a : integer; o : worldarray;
                 y : real; b : integer; h : jstarray;
                 p, q, r, s : worldarray); extern;
procedure JSSM1(g : gtable; i, j, k : integer; m, n, o, p : real;
                q : integer; r : real); extern;
procedure JDEVOF(i : integer); extern;
procedure JDEND(i : integer); extern;
procedure JEND; extern;



{ This is a macro to clear text from the dialog area }

procedure HOME_N_CLEAR; extern;



{*************************************************************************}
{*                                                                       *}
{*  INITGRAPHICS initializes the graphics devices used.                  *}
{*                                                                       *}
{*************************************************************************}
procedure INITGRAPHICS;
begin
   JBEGIN;
   JDINIT(1);
   JDEVON(1);
   JVSPAC(-1.0, 1.0, -0.7, 0.7);
   JVPORT(-1.0, 1.0, -0.7, 0.7);
   JWINDO(0.0, 100.0, 0.0, 70.0);
end; {INITGRAPHICS}



{*************************************************************************}
{*                                                                       *}
{* ENDGRAPHICS deactivates the graphics devices.                         *}
{*                                                                       *}
```

```
{*************************************************************************}
procedure ENDGRAPHICS;
begin
   JDEVOF(1);
   JDEND(1);
   JEND;
end;


{*************************************************************************}
{*                                                                       *}
{* PUTVAL writes val1 and val2 in the position determined by x and y.    *}
{* Values are surrounded by parentheses and separated by a comma.        *}
{*                                                                       *}
{*************************************************************************}
procedure PUTVAL(x, y : real; val1, val2 : parray);
begin
   JMOVE(x, y);
   JHSTRG(%stdescr '(');
   JMOVE(x + 1.0, y);
   JHSTRG(%stdescr val1);
   JMOVE(x + 11.0, y);
   JHSTRG(%stdescr ',');
   JMOVE(x + 13.0, y);
   JHSTRG(%stdescr val2);
   JMOVE(x + 23.0, y);
   JHSTRG(%stdescr ')');
end; {PUTVAL}


{*************************************************************************}
{*                                                                       *}
{* PUT1VAL puts val1 in place surrounded by parentheses, and val2 in a   *}
{* different place, also surrounded by parentheses.                      *}
{*                                                                       *}
{*************************************************************************}
procedure PUT1VAL(x, y : real; val1, val2 : parray);
begin
   JMOVE(x, y);
   JHSTRG(%stdescr '(');
   JMOVE(x + 1.0, y);
   JHSTRG(%stdescr val1);
   JMOVE(x + 11.0, y);
   JHSTRG(%stdescr ')');
   JMOVE(x, y - 30.0);
   JHSTRG(%stdescr '(');
   JMOVE(x + 1.0, y - 30.0);
   JHSTRG(%stdescr val2);
   JMOVE(x + 11.0, y - 30.0);
   JHSTRG(%stdescr ')');
end; {PUT1VAL}


{*************************************************************************}
{*                                                                       *}
{* CONVERT performs the conversion necessary to change an integer into a *}
{* string equivalent which can be printed by the JHSTRG procedure.       *}
{*                                                                       *}
```

```
{*************************************************************************}
procedure CONVERT(intger : integer; var wrd : parray);
var
   temp : sarray;
begin
   temp := DEC(intger, stsize, 1);  {DEC is a predeclared pascal function}
   wrd := temp;
end; {CONVERT}



{*************************************************************************}
{*                                                                       *}
{* WRITEADDVAL prints the lottery values needed by CHECKADDIND to deter- *}
{* mine whether additive independence holds.                             *}
{*                                                                       *}
{*************************************************************************}
procedure WRITEADDVAL(min1, max1, min2, max2 : integer; xpos, ypos : real);
var
   smax1, smax2,
   smin1, smin2 : parray;
begin
   CONVERT(max1, smax1);
   CONVERT(min1, smin1);
   CONVERT(max2, smax2);
   CONVERT(min2, smin2);
   PUTVAL(xpos, ypos, smax1, smax2);
   PUTVAL(xpos, ypos - 30.0, smin1, smin2);
   PUTVAL(xpos + 40.0, ypos, smax1, smin2);
   PUTVAL(xpos + 40.0, ypos - 30.0, smin1, smax2);
end; {WRITEADDVAL}



{*************************************************************************}
{*                                                                       *}
{* GETSTRING gets the particular values needed for lotteries in a varying *}
{* array format.                                                         *}
{*                                                                       *}
{*************************************************************************}
procedure GETSTRING(att1 : attribute; min1, max1, min2, max2 : range;
                    var i : integer; var s1, s2, s3, s4, s5, s6 : sarray;
                    t : tarray);
begin
   if t[i].attrib = att1 then
   begin
      s1 := DEC(max1, stsize, 1);
      s2 := DEC(t[i].pos, stsize, 1);
      s3 := DEC(min1, stsize, 1);
      s4 := DEC(t[i].pos, stsize, 1);
      s5 := ' ? ';
      s6 := DEC(t[i].pos, stsize, 1);
   end
   else
   begin
      s1 := DEC(t[i].pos, stsize, 1);
      s2 := DEC(max2, stsize, 1);
      s3 := DEC(t[i].pos, stsize, 1);
      s4 := DEC(min2, stsize, 1);
      s5 := DEC(t[i].pos, stsize, 1);
```

```
      s6 := '  ? ';
   end;
end; {GETSTRING}


{*************************************************************************}
{*                                                                       *}
{* SETUP draws a lottery frame with .5's on it, but prints no values nor  *)
{* reads the c.e.                                                        *}
{*                                                                       *}
{*************************************************************************}
procedure SETUP;
var
   strg : wordstr;
begin
   JCOLOR(6);
   JSIZE(2.0, 2.0);
   JJUST(2, 2);
   JMOVE(65.0, 45.0);
   JDRAW(40.0, 30.0);
   JDRAW(65.0, 15.0);
   JJUST(1, 2);
   JMOVE(50.0, 40.0);
   JHSTRG(%stdescr '.5');
   JJUST(1, 3);
   JMOVE(50.0, 20.0);
   JHSTRG(%stdescr '.5');
   JMOVE(35.0, 32.0);
   JSIZE(4.0, 4.0);
   JHSTRG(%stdescr '~');
   JSIZE(2.0, 2.0);
end; {SETUP}


{*************************************************************************}
{*                                                                       *}
{* DRAWLOT adds appropriate values to the lottery drawn by SETUP.  The    *}
{* c.e.'s are used to  help determine independence properties.           *}
{*                                                                       *}
{*************************************************************************}
procedure DRAWLOT(att1, att2 : attribute; min1, max1, min2, max2 : range;
                  var index : integer; tab : tarray);
var
   strg1, strg2,
   strg3, strg4, strg5, strg6 : sarray;
   s1, s2 : parray;
   dummychar : char;
   x, y : real;

begin
   SETUP;
   GETSTRING(att1, min1, max1, min2, max2, index, strg1, strg2,
             strg3, strg4, strg5, strg6, tab);
   JJUST(1, 2);
   s1 := strg3;
   s2 := strg4;
   PUTVAL(67.0, 15.0, s1, s2);

   s1 := strg5;
   s2 := strg6;
```

```
   PUTVAL(5.0, 30.0, s1, s2);

   s1 := strg1;
   s2 := strg2;
   PUTVAL(67.0, 45.0, s1, s2);

end; {DRAWLOT}


{*************************************************************************}
{*                                                                       *}
{* LOTTERIES has DRAWLOT present the lotteries, and it reads in the cer- *}
{* tainty equivalents for them into an array.                            *}
{*                                                                       *}
{*************************************************************************}
procedure LOTTERIES(att1, att2 : attribute; min1, max1, min2, max2 : range;
                    var lottab : tarray);
var
   a, i : integer;
   dummychar : char;
begin
   HOME_N_CLEAR;
   for i := 1 to 10 do
      writeln;
   writeln('   You will now be shown a series of lotteries for which you');
   writeln('must enter your certainty equivalents:');
   for i := 1 to 5 do
      writeln;
   write('Hit any key followed by return to continue: ');
   readln(dummychar);
   HOME_N_CLEAR;
   INITGRAPHICS;
   a := 0;
   for i := 1 to 10 do
   begin
      a := a + 1;
      JOPEN;
      DRAWLOT(att1, att2, min1, max1, min2, max2, a, lottab);
      JCLOSE;
      writeln('In terms of the (',att1scale,', ',att2scale,') outcome,');
      write('please enter your value for the question mark: ');
      readln(lottab[i].ans);
      JFRAME;
      HOME_N_CLEAR;
   end;
end; {LOTTERIES}




{*************************************************************************}
{*                                                                       *}
{* WELCOME is a short introductory paragraph to the program.             *}
{*                                                                       *}
{*************************************************************************}
procedure WELCOME;
var
   i : integer;
   dummychar : char;
```

```
begin
   HOME_N_CLEAR;
   for i := 1 to 8 do
      writeln;
   writeln('     This program is a generic multi-attribute utility theory');
   writeln('based methodology for assisting a decision maker.  From inter-');
   writeln('acting with the decision maker, it will determine the decision');
   writeln('maker''s utility function over two user specified attributes.');
   writeln('From there, it will present the utility function graphically,');
   writeln('at the angle of rotation desired by the decision maker.');
   for i := 1 to 10 do
      writeln;
   write('Please enter any character followed by a return to continue: ');
   readln(dummychar);
   HOME_N_CLEAR;
end; {WELCOME}



{*************************************************************************}
{*                                                                       *}
{* ATTRIBUTES asks the decision maker for his attributes and their       *}
{* ranges.                                                               *}
{*                                                                       *}
{*************************************************************************}
procedure ATTRIBUTES(var att1,att2 : attribute; var min1, max1, min2, max2 :
                     range);
var
   i : integer;
   dummychar : char;
begin
   for i := 1 to 3 do
      writeln;
   writeln('   Please enter your first attribute.  If the attribute is over');
   writeln('10 characters long, please abbreviate it to be 10 characters.');
   writeln;
   write('attribute name: ');
   readln(att1);
   for i := 1 to 2 do
      writeln;
   write('What units is this attribute measured in?: ');
   readln(att1scale);
   for i := 1 to 2 do
     writeln;
   writeln('  Please enter the least and most preferred values this attribute');
   writeln('can be -- (to the nearest whole number).');
   writeln;
   write('least preferred: ');
   read(min1);
   write('most preferred: ');
   readln(max1);
   for i := 1 to 2 do
      writeln;
   writeln('   Please enter your second attribute.  Again, please insure it');
   writeln('does not exceed 10 characters in length.');
   writeln;
   write('attribute name: ');
   readln(att2);
   for i := 1 to 2 do
   writeln;
```

```
    write('What units is this attribute measured in?: ');
    readln(att2scale);
    for i := 1 to 2 do
      writeln;
    writeln('  Please enter the least and most preferred values this attribute');
    writeln('can have -- (to the nearest whole number). ');
    writeln;
    write('least preferred: ');
    read(min2);
    write('most preferred: ');
    readln(max2);
    write('Hit any key followed by return to continue: ');
    readln(dummychar);
end; {ATTRIBUTES}


{*************************************************************************}
{*                                                                       *}
{* DEVTABLE builds a table of values for each attribute at (min, .25, .5, *}
{* .75, and max) values.                                                 *}
{*                                                                       *}
{*************************************************************************}
procedure DEVTABLE(att1, att2 : attribute; min1, max1, min2, max2 : range;
                   var table : tarray);
begin
    table[1].attrib := att1;   {For example, table[1] will be used to add}
    table[1].pos := min2;      {values to a lottery assessing the c.e. of}
                               {attribute 1 at attribute 2's minimum value}
    table[2].attrib := att1;
    table[2].pos := ROUND((max2 - min2)/2 + min2);
    table[3].attrib := att2;
    table[3].pos := max1;
    table[4].attrib := att1;
    table[4].pos := ROUND((max2 - min2)/4 + min2);
    table[5].attrib := att2;
    table[5].pos := ROUND((max1 - min1)/2 + min1);
    table[6].attrib := att2;
    table[6].pos := ROUND((max1 - min1)/4 + min1);
    table[7].attrib := att1;
    table[7].pos := max2;
    table[8].attrib := att2;
    table[8].pos := min1;
    table[9].attrib := att1;
    table[9].pos := ROUND(((max2 - min2)/4) * 3 + min2);
    table[10].attrib := att2;
    table[10].pos := ROUND(((max1 - min1)/4) * 3 + min1);
end; {DEVTABLE}


{*************************************************************************}
{*                                                                       *}
{* LOTFRAME is a generic procedure for drawing lottery angles.           *}
{*                                                                       *}
{*************************************************************************}
procedure LOTFRAME(xcord, ycord, xdisp, ydisp : real);
begin
    JMOVE(xcord, ycord);
    JDRAW(xcord - xdisp, ycord - ydisp);
    JDRAW(xcord, ycord - (ydisp * 2));
end; {LOTFRAME}
```

```
{*************************************************************************}
{*                                                                       *}
{* CHECKADDIND is used to present additional lotteries to determine if   *}
{* the additive independence property holds once mutual utility indepen- *}
{* dence has been determined.                                            *}
{*                                                                       *}
{*************************************************************************}
procedure CHECKADDIND(tab : tarray; var addindep : boolean);
var
   x, y : real;
   choice : integer;
   answer1, answer2 : varying[3] of char;
begin
   JFRAME;
   HOME_N_CLEAR;
   JOPEN;
   JSIZE(2.0, 2.0);
   JCOLOR(6);
   LOTFRAME(20.0, 45.0, 15.0, 15.0);
   LOTFRAME(60.0, 45.0, 15.0, 15.0);
   JJUST(1, 1);
   x := 6.0;
   y := 37.0;
   repeat
      repeat
         JMOVE(x, y);
         JHSTRG(%stdescr '.5');
         x := x + 40.0;
      until x > 60.0;
      x := 6.0;
      y := y - 14.0;
      JJUST(1, 3);
   until y < 23.0;
   x := 21.0;
   y := 45.0;
   WRITEADDVAL(minimum1, maximum1, minimum2, maximum2, x, y);
   JCLOSE;
   writeln('With respect to the lotteries below, in terms of the');
   writeln('(',att1scale,', ',att2scale,') outcome, please indicate ');
   writeln('which statement corresponds to your feelings about this lottery: ');

   writeln('1) I am indifferent between the lotteries.');
   writeln('2) I prefer the lottery on the left.');
   writeln('3) I prefer the lottery on the right.');
   write('Choice: ');
   readln(choice);
   if choice = 1 then
      addindep := true
   else
      addindep := false;
   HOME_N_CLEAR;
   JFRAME;
end; {CHECKADDIND}
```

```
{*************************************************************************}
{*                                                                       *}
{* UTIL finds the values of c and b which fit a utility function to the  *}
{* decision maker's utility.                                             *}
{*                                                                       *}
{*************************************************************************}
procedure UTIL(maxt, cmt : real; var b, c : real; sign : integer);
var
   dummychar   char;
   i, j, loop   integer;
   rmax, rmin,
   valnw   real,
   continue   boolean;
begin
   j   = 0,
   c   = 0.5,
   i   = 0;
   continue   = true;
   rmax   = 1,
   rmin   = 0;
   while continue do
   begin
      valnw   = 2 * (1.0 - EXP(sign * cmt * c)) - (1.0 - EXP(sign * maxt * c));
      if ABS(valnw) < 0.0001 then
      begin
         continue   = false;
         b   = 1.0/(1.0 - EXP(sign * c * maxt));
      end
      else
      begin
         i   = i + 1;
         if i > 35 then
         begin
            continue   = false;
            j   = 1;
         end;
         if valnw > 0 then
            rmax   = c
         else
            rmin   = c;
         c   = (rmax + rmin)/2.0;
      end;
   end;
   c   = c * sign;
   if j = 1 then
   begin
      HOME_N_CLEAR;
      for loop   = 1 to 5 do
         writeln;
      writeln('   The prespecified curve used to approximate your utility');
      writeln('curve doesn''t match your values well enough to be used as');
      writeln('an accurate representation.   Unfortunately, this program');
      writeln('can''t be used to assess your utility.  SORRY!');
      for loop   = 1 to 5 do
         writeln;
      writeln('Hit any key followed by return to exit back to system level.');
      readln(dummychar);
      HALT;
   end;
end; {UTIL}
```

```
{*************************************************************************}
{*                                                                       *}
{* SCALE assesses the user's preferences between outcomes to determine   *}
{* the values for ky, kz, and kyz in the mutually independent and addi-  *}
{* tive cases.  In the additive case, kyz is automatically zero.         *}
{*                                                                       *}
{*************************************************************************}
procedure SCALE(mode : char; ymax, ymin, zmax, zmin : integer; att1,
                att2 : attribute;
                var ky, kz, kyz : real; scaley, scalez, zb, yb,
                zc, yc : real);
var
   i, choice, yval,
   zval, yval2, zval2 : integer;
   symax, symin, szmax, szmin : parray;
   k, eq1, eq2a, eq2b, eq3, eq4, tempk : real;
begin
   HOME_N_CLEAR;
   for i := 1 to 5 do
      writeln;
   writeln('In terms of the (',attiscale,', ',att2scale,') outcome,');
   writeln('Please enter your preference.  ');
   writeln;
   writeln('     (',ymax,', ',zmin,')   or   (',ymin,', ',zmax,')');
   writeln;
   writeln('1) I prefer the outcome on the left.');
   writeln('2) I prefer the outcome on the right.');
   writeln('3) I am indifferent between outcomes.');
   write('Choice:  ');
   readln(choice);
   for i := 1 to 5 do
      writeln;
   if ((choice = 1) or (choice = 3)) then
   begin
      writeln('What amount of ', attiscale,' would make you indifferent to ');
      writeln('the outcomes below?');
      writeln;
      writeln('(  ? , ',zmin,') ~ (',ymin,' ,',zmax,')');
      write('amount of ',attiscale,':  ');
      readln(yval);
   end
   else
   begin
      writeln('What amount of ',att2scale,' would make you indifferent to ');
      writeln('the outcomes below?');
      writeln;
      writeln('(',ymin,',  ? ) and (',ymax,', ',zmin,')');
      write('amount of ',att2scale,':  ');
      readln(zval);
   end;
   HOME_N_CLEAR;
   JOPEN;
   JSIZE(2.0, 2.0);
   JCOLOR(6);
   LOTFRAME(65.0, 45.0, 25.0, 15.0);
   JJUST(1, 2);
   JMOVE(50.0, 40.0);
```

```
JHSTRG(%stdescr '.5');
JJUST(1, 3);
JMOVE(50.0, 20.0);
JHSTRG(%stdescr '.5');
JJUST(1, 2);
CONVERT(zmin, szmin);
CONVERT(zmax, szmax);
CONVERT(ymin, symin);
CONVERT(ymax, symax);
PUTVAL(67.0, 45.0, symax, szmax);
PUTVAL(67.0, 15.0, symin, szmin);
JSIZE(4.0, 4.0);
PUTVAL(5.0, 30.0, ' ? ', ' # ');
JMOVE(35.0, 30.0);
JHSTRG(%stdescr '~');
JSIZE(2.0, 2.0);
JCLOSE;
writeln('In terms of the (',attiscale,', ',att2scale,') outcome,');
writeln('what values of ',attiscale,' (?) and ',att2scale,' (#)');
writeln('will make you indifferent between the c.e. and the lottery?');
write('Value of ',attiscale,': ');
readln(yval2);
write('Value of ',att2scale,': ');
readln(zval2);
JFRAME;
HOME_N_CLEAR;

if mode = 'm' then
begin
   case choice of
   1: begin
      tempk := yb * (1.0 - EXP(yc * ((yval - ymin) * scaley)));
      eq1 := yb * (1.0 - EXP(yc * ((yval2 - ymin) * scaley)));
      eq2a := zb * (1.0 - EXP(zc * ((zval2 - zmin) * scalez)));
      eq2b := zb * (1.0 - EXP(zc * ((zval2 - zmin) * scalez))) * tempk;
   end;
   2 : begin
      tempk := zb * (1.0 - EXP(zc * ((zval - zmin) * scalez)));
      eq1 := zb * (1.0 - EXP(zc * ((zval2 - zmin) * scalez)));
      eq2a := yb * (1.0 - EXP(yc * ((yval2 - ymin) * scaley)));
      eq2b := yb * (1.0 - EXP(yc * ((yval2 - ymin) * scaley))) * tempk;
   end;
   3 : begin
       tempk := 1.0;
       eq1 := yb * (1.0 - EXP(yc * ((yval2 - ymin) * scaley)));
       eq2a := zb * (1.0 - EXP(zc * ((zval2 - zmin) * scalez)));
       eq2b := eq2a;
    end;
   end; { of case }
   eq3 := eq1 * eq2a;
   eq4 := (-1.0 * (1.0 + tempk)) * eq3;
   k := (0.5 - eq3) / (eq1 + eq2b + eq4);
   case choice of
   1 : begin
       ky := k;
       kz := tempk * ky;
    end;
   2 : begin
       kz := k;
       ky := tempk * kz;
```

```
        end;
      3 : begin
            ky := k;
            kz := k;
        end;
      end; {of case}
      kyz := 1.0 - (ky + kz);
   end
   else
   begin
      eq1 := yb * (1.0 - EXP(yc * ((yval2 - ymin) * scaley)));
      eq2a := zb * (1.0 - EXP(zc * ((zval2 - zmin) * scalez)));
   if eq1 = eq2a then
      ky := 0.5
   else
      ky := (0.5 - eq2a)/(eq1 - eq2a);
      kz := 1.0 - ky;
      kyz := 0.0;
   end;
end; {SCALE}
```

```
{*************************************************************************}
{*                                                                       *}
{* UTILITY assumes a particular utility function, and determines the uti- *}
{* lity of the input value (of an attribute).                            *}
{*                                                                       *}
{*************************************************************************}
function UTILITY(val, b, c, scale : real; maximum, minimum : integer):real;
begin
   if val = 0 then
      UTILITY := 0
   else
      if val = maximum then
         UTILITY := 1.0
      else
         UTILITY := b * (1.0 - EXP(c * ((val - minimum) * scale)));
end; {UTILITY}
```

```
{*************************************************************************}
{*                                                                       *}
{* FILLARRAYS initializes the world coordinates' maximums and minimums.  *}
{* It also locates the center of the mesh surface to be drawn (at array). *}
{*                                                                       *}
{*************************************************************************}
procedure FILLARRAYS;
var
   i : integer;
begin
   for i := 1 to 3 do
   begin
```

```
        wldmin[i] := 0.0;
        wldmax[i] := 10.0;
        at[i] := 5.0;
    end;
end; {FILLARRAYS}



{*************************************************************************}
{*                                                                       *}
{* GETVIEW offers the user the choice of eight different viewpoints for  *}
{* examining the mesh surface.                                           *}
{*                                                                       *}
{*************************************************************************}
procedure GETVIEW(var choose : integer);
begin
    HOME_N_CLEAR;
    writeln('Please indicate your choice of viewpoint for the utility graph.');
    writeln('1)front        3)right        5)back       7)left');
    writeln('2)front-right  4)back-right   6)back-left  8)front-left  9)quit');
    write('choice: ');
    readln(choose);
end; {GETVIEW}



{*************************************************************************}
{*                                                                       *}
{* ZAXSINIT initializes values needed to draw the z axis for the mesh    *}
{* face. (for greater detail, see a DI-3000 Contouring User's Manual.)   *}
{*                                                                       *}
{*************************************************************************}
procedure ZAXSINIT(var beg, endpt, tik : worldarray; var jst : jstarray;
                   var lab, skp : worldarray);
begin
    beg[1] := 0.0;              {axis beginning}
    beg[2] := 0.0;
    beg[3] := 0.0;
    endpt[1] := 0.0;            {axis ending}
    endpt[2] := 0.0;
    endpt[3] := 10.0;
    tik[1] := 0.5;              {length and direction of tic marks}
    tik[2] := 0.0;
    tik[3] := 0.0;
    jst[1] := 3;                {justification of tic marks and labels}
    jst[2] := 2;
    lab[1] := 0.7;
    lab[2] := 0.0;
    lab[3] := 0.0;
    skp[1] := 20.0;             {value to skip when adding tic marks - in}
    skp[2] := 20.0;             {this case, none.                       }
    skp[3] := 20.0;
end; {ZAXSINIT}



{*************************************************************************}
{*                                                                       *}
{* YAXSINIT performs the same function for the y axis as ZAXSINIT does   *}
```

```
{* for the z axis.                                                          *}
{*                                                                         *}
{*************************************************************************}
procedure YAXSINIT(var beg, endpt, tik : worldarray; var jst : jstarray;
                   var lab, bas, pln, skp : worldarray; number : integer);
begin
   beg[1] := 0.0;
   beg[2] := 0.0;
   beg[3] := 0.0;
   endpt[1] := 0.0;
   if number = 0 then
      endpt[2] := 10.0
   else
      endpt[2] := 5.0;
   endpt[3] := 0.0;
   tik[1] := 0.0;
   tik[2] := 0.0;
   tik[3] := -0.5;
   jst[1] := 2;
   jst[2] := 3;
   lab[1] := 0.0;
   lab[2] := 0.0;
   lab[3] := -0.7;
   bas[1] := 1.0;
   bas[2] := 0.0;
   bas[3] := 0.0;
   pln[1] := 0.0;
   pln[2] := 1.0;
   pln[3] := 0.0;
   skp[1] := 0.0;
   skp[2] := 0.0;
   skp[3] := 0.0;
end; {YAXSINIT}


{*************************************************************************}
{*                                                                       *}
{* XAXSINIT performs the same function for the x axis as ZAXSINIT does   *}
{* for the z axis.                                                       *}
{*                                                                       *}
{*************************************************************************}
procedure XAXSINIT(var beg, endpt, tik : worldarray; var jst : jstarray;
                   var lab, bas, pln, skp : worldarray; number : integer);
begin
   beg[1] := 0.0;
   beg[2] := 0.0;
   beg[3] := 0.0;
   if number = 0 then
      endpt[1] := 10.0
   else
      endpt[1] := 5.0;
   endpt[2] := 0.0;
   endpt[3] := 0.0;
   tik[1] := 0.0;
   tik[2] := 0.0;
   tik[3] := -0.5;
   jst[1] := 2;
   jst[2] := 3;
   lab[1] := 0.0;
```

```
   lab[2] := 0.0;
   lab[3] := -0.7;
   bas[1] := 1.0;
   bas[2] := 0.0;
   bas[3] := 0.0;
   pln[1] := 0.0;
   pln[2] := 1.0;
   pln[3] := 0.0;
   skp[1] := 0.0;
   skp[2] := 0.0;
   skp[3] := 0.0;
end; {XAXSINIT}



{*************************************************************************}
{*                                                                       *}
{* PLOTGRAPH draws the 3-D mesh surface based on the user's choice of    *}
{* view.  It also draws axes for the mesh surface.                       *}
{*                                                                       *}
{*************************************************************************}
procedure PLOTGRAPH(choice : integer; grid : gtable; num : integer);
var
   i, j : integer;
   g : gtable;
begin
      for i := 1 to 11 do
         for j := 1 to 11 do
            g[i, j] := grid[i, j] * 10;   {scales utility}
                                           {from 1 to 10}
                                       {for greater visibility}


{sets the viewpoint the user has specified}
   case choice of
   1 : begin
          eye[1] := 5.0;
          eye[2] := -20.0;
          eye[3] := 15.0;
          zdelbas[1] := 1;
          zdelbas[2] := 0;
          zdelbas[3] := 0;
          zdelpln[1] := 0;
          zdelpln[2] := 0;
          zdelpln[3] := 1;
       end;
   2 : begin
          eye[1] := 30.0;
          eye[2] := -20.0;
          eye[3] := 15.0;
          zdelbas[1] := 1;
          zdelbas[2] := 1;
          zdelbas[3] := 0;
          zdelpln[1] := 0;
          zdelpln[2] := 0;
          zdelpln[3] := 1;
       end;
   3 : begin
```

```
            eye[1] := 5.0;
            eye[2] := 30.0;
            eye[3] := 15.0;
            zdelbas[1] := -1;
            zdelbas[2] := 0;
            zdelbas[3] := 0;
            zdelpln[1] := 0;
            zdelpln[2] := 0;
            zdelpln[3] := 1;
        end;
    8 : begin
            eye[1] := -20.0;
            eye[2] := -20.0;
            eye[3] := 15.0;
            zdelbas[1] := 1;
            zdelbas[2] := -1;
            zdelbas[3] := 0;
            zdelpln[1] := 0;
            zdelpln[2] := 0;
            zdelpln[3] := 1;
        end;
    3 : begin
            eye[1] := 30.0;
            eye[2] := 5.0;
            eye[3] := 15.0;
            zdelbas[1] := 0;
            zdelbas[2] := 1;
            zdelbas[3] := 0;
            zdelpln[1] := 0;
            zdelpln[2] := 0;
            zdelpln[3] := 1;
        end;
    4 : begin
            eye[1] := 30.0;
            eye[2] := 30.0;
            eye[3] := 15.0;
            zdelbas[1] := -1;
            zdelbas[2] := 1;
            zdelbas[3] := 0;
            zdelpln[1] := 0;
            zdelpln[2] := 0;
            zdelpln[3] := 1;
        end;
    7 : begin
            eye[1] := -20.0;
            eye[2] := 5.0;
            eye[3] := 15.0;
            zdelbas[1] := 0;
            zdelbas[2] := -1;
            zdelbas[3] := 0;
            zdelpln[1] := 0;
            zdelpln[2] := 0;
            zdelpln[3] := 1;
        end;
    6 : begin
            eye[1] := -20.0;
            eye[2] := 30.0;
            eye[3] := 15.0;
            zdelbas[1] := -1;
            zdelbas[2] := -1;
```

```
        zdelbas[3] := 0;
        zdelpln[1] := 0;
        zdelpln[2] := 0;
        zdelpln[3] := 1;
    end;
end; {of case}
ENDGRAPHICS;
if num = 1 then
   for i := 1 to 2 do
      eye[i] := eye[i]/2;
JBEGIN;
JDINIT(1);
JDEVON(1);
JSSINI;
JSSPRO(wldmin, wldmax, eye, at);
ZAXSINIT(zaxsbeg, zaxsend, zdeltik, zlabjst, zdellab, zskpval);
YAXSINIT(yaxsbeg, yaxsend, ydeltik, ylabjst, ydellab, ydelbas, ydelpln,
         yskpval, num);
XAXSINIT(xaxsbeg, xaxsend, xdeltik, xlabjst, xdellab, xdelbas, xdelpln,
         xskpval, num);
JOPEN;
JSSAXS(zaxsbeg, zaxsend, 1.0, 2, zdeltik, 0.5, 3, zlabjst,
       zdellab, zdelbas, zdelpln, zskpval);
JSSAXS(yaxsbeg, yaxsend, 5.0, 1, ydeltik, 0.5, 0, ylabjst,
       ydellab, ydelbas, ydelpln, yskpval);
JSSAXS(xaxsbeg, xaxsend, 5.0, 1, xdeltik, 0.5, 0, xlabjst,
       xdellab, xdelbas, xdelpln, xskpval);
JCLOSE;

if num = 0 then
   JSSM1(g, 11, 11, 11, 0.0, 0.0, 0.0, 0.0, -3, 1.0)
else
   JSSM1(g, 11, 6, 6, 0.0, 0.0, 0.0, 0.0, -3, 1.0);
JOPEN;
JSIZE(0.5, 0.5);
if num = 0 then
case choice of
  1:  begin
      JPLANE(0.0, 1.0, 0.5);
      J3MOVE(2.0, 0.0, 0.0);
      JHSTRG(%stdescr att1scale);
      J3MOVE(-1.0, 8.0, 0.0);
      JHSTRG(%stdescr att2scale);
      end;

  2:  begin
      JPLANE(-1.0, 1.0, 0.5);
      J3MOVE(2.0, 0.0, 0.0);
      JHSTRG(%stdescr att1scale);
      JBASE(0.0, 1.0, 0.0);
      J3MOVE(0.0, 2.0, 0.0);
      JHSTRG(%stdescr att2scale);
      JBASE(1.0, 0.0, 0.0);
      end;

  3:  begin
      JPLANE(-1.0, 0.0, 0.5);
      JBASE(0.0, 1.0, 0.0);
      J3MOVE(9.0, -1.0, 0.0);
      JHSTRG(%stdescr att1scale);
```

```
        J3MOVE(0.0, 2.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JBASE(1.0, 0.0, 0.0);
        end;

4:      begin
        JPLANE(-1.0, -1.0; 0.5);
        JBASE(-1.0, 0.0, 0.0);
        J3MOVE(2.0, 0.0, 0.0);
        JJUST(3, 1);
        JHSTRG(%stdescr att1scale);
        JJUST(1, 1);
        JBASE(0.0, 1.0, 0.0);
        J3MOVE(0.0, 2.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JBASE(1.0, 0.0, 0.0);
        end;

5:      begin
        JPLANE(0.0, -1.0, 0.5);
        J3MOVE(2.0, 0.0, 0.0);

        JBASE(-1.0, 0.0, 0.0);
        JJUST(3, 1);
        JHSTRG(%stdescr att1scale);
        JJUST(1, 1);
        J3MOVE(3.0, 9.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JBASE(1.0, 0.0, 0.0);
        end;

6:      begin
        JPLANE(1.0, -1.0, 0.5);
        JBASE(-1.0, 0.0, 0.0);
        JJUST(3, 1);
        J3MOVE(2.0, 0.0, 0.0);
        JHSTRG(%stdescr att1scale);
        JBASE(0.0, -1.0, 0.0);
        J3MOVE(0.0, 2.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JJUST(1, 1);
        JBASE(1.0, 0.0, 0.0);
        end;

7:      begin
        JPLANE(1.0, 0.0, 0.5);
        JBASE(0.0, -1.0, 0.0);
        JJUST(3, 1);
        J3MOVE(9.0, -3.0, 0.0);
        JHSTRG(%stdescr att1scale);
        J3MOVE(0.0, 2.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JJUST(1, 1);
        JBASE(1.0, 0 0, 0.0);
        end;

8:      begin
        JPLANE(1.0, 1.0, 0.5);
        J3MOVE(2.0, 0.0, 0.0);
        JHSTRG(%stdescr att1scale);
```

```
        JBASE(0.0, -1.0, 0.0);
        JJUST(3, 1);
        J3MOVE(0.0, 2.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JJUST(1, 1);
        JBASE(1.0, 0.0, 0.0);
        end;

end { case }
else
case choice of
  1:  begin
        JPLANE(0.0, 1.0, 0.5);
        J3MOVE(1.0, 0.0, 0.0);
        JHSTRG(%stdescr att1scale);
        J3MOVE(-0.5, 4.0, 0.0);
        JHSTRG(%stdescr att2scale);
        end;

  2:  begin
        JPLANE(-1.0, 1.0, 0.5);
        J3MOVE(1.0, 0.0, 0.0);
        JHSTRG(%stdescr att1scale);
        JBASE(0.0, 1.0, 0.0);
        J3MOVE(0.0, 1.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JBASE(1.0, 0.0, 0.0);
        end;

  3:  begin
        JPLANE(-1.0, 0.0, 0.5);
        JBASE(0.0, 1.0, 0.0);
        J3MOVE(4.5, -0.5, 0.0);
        JHSTRG(%stdescr att1scale);
        J3MOVE(0.0, 1.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JBASE(1.0, 0.0, 0.0);
        end;

  4:  begin
        JPLANE(-1.0, -1.0, 0.5);
        JBASE(-1.0, 0.0, 0.0);
        J3MOVE(1.0, 0.0, 0.0);
        JJUST(3, 1);
        JHSTRG(%stdescr att1scale);
        JJUST(1, 1);
        JBASE(0.0, 1.0, 0.0);
        J3MOVE(0.0, 1.0, 0.0);
        JHSTRG(%stdescr att2scale);
        JBASE(1.0, 0.0, 0.0);
        end;

  5:  begin
        JPLANE(0.0, -1.0, 0.5);
        J3MOVE(1.0, 0.0, 0.0);

        JBASE(-1.0, 0.0, 0.0);
        JJUST(3, 1);
        JHSTRG(%stdescr att1scale);
        JJUST(1, 1);
```

```
            J3MOVE(1.5, 4.5, 0.0);
            JHSTRG(%stdescr att2scale);
            JBASE(1.0, 0.0, 0.0);
            end;

        6:  begin
            JPLANE(1.0, -1.0, 0.5);
            JBASE(-1.0, 0.0, 0.0);
            JJUST(3, 1);
            J3MOVE(1.0, 0.0, 0.0);
            JHSTRG(%stdescr att1scale);
            JBASE(0.0, -1.0, 0.0);
            J3MOVE(0.0, 1.0, 0.0);
            JHSTRG(%stdescr att2scale);
            JJUST(1, 1);
            JBASE(1.0, 0.0, 0.0);
            end;

        7:  begin
            JPLANE(1.0, 0.0, 0.5);
            JBASE(0.0, -1.0, 0.0);
            JJUST(3, 1);
            J3MOVE(4.5, -1.5, 0.0);
            JHSTRG(%stdescr att1scale);
            J3MOVE(0.0, 1.0, 0.0);
            JHSTRG(%stdescr att2scale);
            JJUST(1, 1);
            JBASE(1.0, 0.0, 0.0);
            end;

        8:  begin
            JPLANE(1.0, 1.0, 0.5);
            J3MOVE(1.0, 0.0, 0.0);
            JHSTRG(%stdescr att1scale);
            JBASE(0.0, -1.0, 0.0);
            JJUST(3, 1);
            J3MOVE(0.0, 1.0, 0.0);
            JHSTRG(%stdescr att2scale);
            JJUST(1, 1);
            JBASE(1.0, 0.0, 0.0);
            end;

    end; { case }


    JCLOSE;
end; {PLOTGRAPH}


{*************************************************************************}
{*                                                                       *}
{* BOXPIC is a representation of the different viewpoints from which the *}
{* user can see his utility function.                                    *}
{*                                                                       *}
{*************************************************************************}
procedure BOXPIC;
var
  dummychar : char;
begin
  JOPEN;
```

```
JCOLOR(1);
JLWIDE(32767);
JLSTYL(0);
JSIZE(2.0, 2.0);
JMOVE(50.0, 56.0);
JHSTRG(%stdescr 'UTILITY');
JMOVE(53.0, 55.0);
JDRAW(53.0, 38.0);
JLSTYL(3);
JDRAW(53.0, 18.0);
JDRAW(70.0, 18.0);
JMOVE(53.0, 18.0);
JDRAW(58.0, 26.0);
JLSTYL(0);
JMOVE(70.0, 18.0);
JDRAW(80.0, 18.0);
JMOVE(82.0, 18.0);
JHSTRG(%stdescr att1scale);
JMOVE(58.0, 26.0);
JDRAW(64.0, 33.0);
JMOVE(66.0, 33.0);
JHSTRG(%stdescr att2scale);
JLWIDE(16383);
JCOLOR(6);
JLSTYL(3);
JMOVE(30.0, 10.0);
JDRAW(30.0, 30.0);
JDRAW(45.0, 45.0);
JDRAW(75.0, 45.0);
JDRAW(60.0, 30.0);
JDRAW(60.0, 10.0);
JDRAW(75.0, 25.0);
JDRAW(45.0, 25.0);
JDRAW(30.0, 10.0);
JDRAW(60.0, 10.0);
JMOVE(75.0, 25.0);
JDRAW(75.0, 45.0);
JMOVE(45.0, 45.0);
JDRAW(45.0, 25.0);
JMOVE(30.0, 30.0);
JDRAW(60.0, 30.0);
JLSTYL(0);
JMOVE(45.0, 30.0);
JSIZE(3.0, 3.0);
JCOLOR(5);
JHSTRG(%stdescr '1');
JMOVE(60.0, 30.0);
JHSTRG(%stdescr '2');
JMOVE(68.0, 38.0);
JHSTRG(%stdescr '3');
JMOVE(75.0, 45.0);
JHSTRG(%stdescr '4');
JMOVE(60.0, 45.0);
JHSTRG(%stdescr '5');
JMOVE(45.0, 45.0);
JHSTRG(%stdescr '6');
JMOVE(38.0, 38.0);
JHSTRG(%stdescr '7');
JMOVE(30.0, 30.0);
JHSTRG(%stdescr '8');
```

```
  JCLOSE;
  writeln('This picture represents the viewpoints from which you');
  writeln('can look at your utility function.');
  writeln('1)front        3)right       5)back       7)left');
  writeln('2)front-right  4)back-right  6)back-left  8)front-left');
  writeln('Hit the S Copy key to get a hard copy of the picture.');
  write('Hit any key followed by return to continue: ');
  readln(dummychar);
end; {BOXPIC}



{*************************************************************************}
{*                                                                       *}
{* FILLGRID fills an array with appropriate utility values based on the  *}
{* given attribute values (y and z).                                     *}
{*                                                                       *}
{*************************************************************************}
procedure FILLGRID(var grid : gtable; ky, kz, kyz, ycon, zcon, zb, yb, zc,
                       yc : real);
var
   y, z, valy, valz : integer;
begin
   for y := 0 to 10 do
      for z := 0 to 10 do
      begin
         valy := ROUND((y/ycon) + minimum1);
         valz := ROUND((z/zcon) + minimum2);
         grid[y + 1, z + 1] := ky * UTILITY(valy, yb, yc, ycon, maximum1,
                                            minimum1)
                               + kz * UTILITY(valz, zb, zc, zcon, maximum2,
                                            minimum2)
                               + kyz * UTILITY(valy, yb, yc, ycon, maximum1,
                                             minimum1)
                               * UTILITY(valz, zb, zc, zcon, maximum2, minimum2);

         if grid[y + 1,z + 1] > 1.0 then      {* this check clips utility *}
            grid[y + 1,z + 1] := 1.0          {* values between 0 and '   *}
         else
            if grid[y + 1,z + 1] < 0.0 then
               grid[y + 1,z + 1] := 0.0;

      end;
end; {FILLGRID}



{*************************************************************************}
{*                                                                       *}
{* MUTORADD is used to gather additional data for developing the utility *}
{* curve with UTIL once it's been determined that mutual utility indepen- *}
{* dence or additive independence holds.                                 *}
{*                                                                       *}
{*************************************************************************}
procedure MUTORADD(mode : char; tab : tarray; var by, cy, cz, bz, cony,
                       conz : real);
var
   continue : boolean;
   dummychar : char;
```

```
    ymid, zmid,
    ymaxt, zmaxt, cmyt, cmzt : real;
    i, j, cmy, cmz, pick,
    yflag, zflag, ly, lz, gnum : integer;
    smax1, smin1, smin2, smax2 : parray;
    gridtable = gtable;
    k1, k2, k3 : real;
begin
    HOME_N_CLEAR;
    for i := 1 to 15 do
       writeln;
    writeln('Since your responses indicate that the');
    if mode = 'm' then
       writeln('attributes are mutually utility independent,')
    else
       writeln('attributes are additive independent,');

       writeln('your utility function can easily be developed by assessing');
       writeln('the marginal utility curve for each attribute.  To do this,');
       writeln('you will be shown additional lotteries for which you must');
       writeln('enter your certainty equivalents.');
    for i := 1 to 5 do
       writeln;
    write('Hit any key followed by return to continue: ');
    readln(dummychar);
    HOME_N_CLEAR;
    JOPEN;
    CONVERT(maximum1, smax1);
    CONVERT(maximum2, smax2);
    CONVERT(minimum1, smin1);
    CONVERT(minimum2, smin2);
    SETUP;
    PUT1VAL(67.0, 45.0, smax1, smin1);
    JMOVE(5.0, 30.0);
    JSIZE(4.0, 4.0);
    JHSTRG(%stdescr '(  ? )');
    JSIZE(2.0, 2.0);
    JCLOSE;
    write('Please enter your certainty equivalent for the "?": ');
    readln(cmy);
    HOME_N_CLEAR;
    JFRAME;
    JOPEN;
    SETUP;
    PUT1VAL(67.0, 45.0, smax2, smin2);
    JMOVE(5.0, 30.0);
    JSIZE(4.0, 4.0);
    JHSTRG(%stdescr '(  ? )');
    JSIZE(2.0, 2.0);
    JCLOSE;
    write('Please enter your certainty equivalent for the "?": ');
    readln(cmz);
    JFRAME;
    cony := 10.0/(maximum1 - minimum1);
    conz := 10.0/(maximum2 - minimum2);
    ymaxt := (maximum1 - minimum1) * cony;
    zmaxt := (maximum2 - minimum2) * conz;
    cmyt := (cmy - minimum1) * cony;
    cmzt := (cmz - minimum2) * conz;
    ymid := (maximum1 - minimum1)/2;
```

```
    zmid := (maximum2 - minimum2)/2;
    yflag := 1;
    zflag := 1;
    if cmy < ymid then
       yflag := -1;
    if cmz < zmid then
       zflag := -1;
    if yflag > 0 then
       UTIL(ymaxt, cmyt, by, cy, 1)
    else
       UTIL(ymaxt, cmyt, by, cy, -1);
    if zflag > 0 then
       UTIL(zmaxt, cmzt, bz, cz, 1)
    else
       UTIL(zmaxt, cmzt, bz, cz, -1);
    SCALE(mode, maximum1, minimum1, maximum2, minimum2, attribute1, attribute2,
          k1, k2, k3, cony, conz, bz, by, zc, yc);
    FILLGRID(gridtable, k1, k2, k3, cony, conz, bz, by, zc, yc);
    gnum := 0;
    continue := true;
    FILLARRAYS;
    HOME_N_CLEAR;
    BOXPIC;
    HOME_N_CLEAR;
    JFRAME;
    while continue do
    begin
       GETVIEW(pick);
       if pick <> 9 then
       begin
          PLOTGRAPH(pick, gridtable, gnum);
          HOME_N_CLEAR;
       end
       else
          continue := false;
    end;
end; {MUTORADD}



{*************************************************************************}
{*                                                                       *}
{* ONEWAY fills the mesh surface array grid in the case where the attri- *}
{* butes are utility independent in one direction only.                  *}
{*                                                                       *}
{*************************************************************************}
procedure ONEWAY(b_1, b_2, b_3, c_1, c_2, c_3, con_1, con_2 : real;
                 var grid : gtable; direction : char);
var
   y, z : integer;
   valy, valz : integer;
begin
   for y := 0 to 10 do
   begin
      valy := ROUND((y/con_2) + minimum1);
      for z := 0 to 10 do
      begin
         valz := ROUND((z/con_1) + minimum2);
         if direction = 'z' then
            grid[y + 1, z + 1] := UTILITY(valy, b_2, c_2, con_2, maximum1,
```

```
                                    minimum1) * (1 - UTILITY(valz, b_1, c_1,
                                    con_1, maximum2, minimum2)) + UTILITY(valy,
                                    b_3, c_3, con_2, maximum1, minimum1) *
                                    UTILITY(valz, b_1, c_1, con_1, maximum2,
                                    minimum2)
            else
               grid[y + 1, z + 1] := UTILITY(valz, b_1, c_1, con_1, maximum2,
                                    minimum2) * (1 - UTILITY(valy, b_2, c_2,
                                    con_2, maximum1, minimum1)) + UTILITY(valz,
                                    b_3, c_3, con_1, maximum2, minimum2) *
                                    UTILITY(valy, b_2, c_2, con_2, maximum1,
                                    minimum1);

         end;
      end;
   end; {ONEWAY}


{*************************************************************************}
{*                                                                       *}
{* UTIND1WAY guides the process for drawing the mesh surface when only   *}
{* one of the attributes is utility independent of the other.            *}
{*                                                                       *}
{*************************************************************************}
procedure UTIND1WAY(dir : char);
var
   continue : boolean;
   smax1, smax2, smin1, smin2 : parray;
   dummychar : char;
   ymid, zmid,
   b1, b2, b3, c1, c2, c3,
   con1, con2,
   cm1t, cm2t, cm3t,
   max1t, max2t, max3t : real;
   pick,
   l1, l2, l3,
   cm1, cm2, cm3,
   flag1, flag2, flag3 : integer;
   gridtable : gtable;
   dumscale_A, dumscale_B : attribute;

begin
   if dir = 'y' then
   begin
      dumscale_A := att1scale;
      dumscale_B := att2scale;
   end
   else
   begin
      dumscale_A := att2scale;
      dumscale_B := att1scale;
   end;
   HOME_N_CLEAR;
   for i := 1 to 15 do
      writeln;
      writeln('Since your responses indicate that ',dumscale_A,' are ');
      writeln('utility independent of ',dumscale_B,', ');
      writeln('your utility function can easily be assessed by developing');
      writeln('three marginal utility curves for the attributes.  To do this,');
```

```
   writeln('you will be shown additional lotteries for which you must');
   writeln('enter your certainty equivalents.');
for i := 1 to 5 do
   writeln;
write('Hit any key followed by return to continue: ');
readln(dummychar);
HOME_N_CLEAR;
JOPEN;
CONVERT(maximum1, smax1);
CONVERT(maximum2, smax2);
CONVERT(minimum1, smin1);
CONVERT(minimum2, smin2);
SETUP;
PUTVAL(67.0, 45.0, smin1, smax2);
PUTVAL(5.0, 30.0, smin1, '  ? ');
PUTVAL(67.0, 15.0, smin1, smin2);
JCLOSE;
writeln('In terms of the (',att1scale,', ',att2scale,') outcome.');
write('please enter your value for the "?": ');
if dir = 'z' then
   readln(cm1)
else
   readln(cm2);
HOME_N_CLEAR;
JFRAME;

JOPEN;
SETUP;
PUTVAL(67.0, 45.0, smax1, smin2);
PUTVAL(5.0, 30.0, '  ? ', smin2);
PUTVAL(67.0, 15.0, smin1, smin2);
JCLOSE;
writeln('In terms of the (',att1scale,', ',att2scale,') outcome.');
write('please enter your value for the "?": ');
if dir = 'z' then
   readln(cm2)
else
   readln(cm1);
HOME_N_CLEAR;
JFRAME;


JOPEN;
SETUP;
if dir = 'z' then
begin
   PUTVAL(67.0, 45.0, smax1, smax2);
   PUTVAL(5.0, 30.0, '  ? ', smax2);
   PUTVAL(67.0, 15.0, smin1, smax2);
end
else
begin
   PUTVAL(67.0, 45.0, smax1, smax2);
   PUTVAL(5.0, 30.0, smax1, '  ? ');
   PUTVAL(67.0, 15.0, smax1, smin2);
end;
JCLOSE;
writeln('In terms of the (',att1scale,', ',att2scale,') outcome.');
write('please enter your value for the "?": ');
readln(cm3);
```

```
HOME_N_CLEAR;
JFRAME;

con1 := 10.0/(maximum2 - minimum2);
con2 := 10.0/(maximum1 - minimum1);

max1t := (maximum2 - minimum2) * con1;
max2t := (maximum1 - minimum1) * con2;

if dir = 'z' then
begin
   max3t := (maximum1 - minimum1) * con2;
   cm1t := (cm1 - minimum2) * con1;
   cm2t := (cm2 - minimum1) * con2;
   cm3t := (cm3 - minimum1) * con2;
end
else
begin
   max3t := (maximum2 - minimum2) * con1;
   cm1t := (cm1 - minimum1) * con2;
   cm2t := (cm2 - minimum2) * con1;
   cm3t := (cm3 - minimum2) * con1;
end;


ymid := (maximum1 - minimum1)/2;
zmid := (maximum2 - minimum2)/2;

flag1 := 1;
flag2 := 1;
flag3 := 1;

if cm1 < zmid then
   flag1 := -1;
if cm2 < ymid then
   flag2 := -1;
if cm3 < ymid then
   flag3 := -1;

if flag1 > 0 then
   UTIL(max1t, cm1t, b1, c1, 1)
else
   UTIL(max1t, cm1t, b1, c1, -1);

if flag2 > 0 then
   UTIL(max2t, cm2t, b2, c2, 1)
else
   UTIL(max2t, cm2t, b2, c2, -1);

if flag3 > 0 then
   UTIL(max3t, cm3t, b3; c3, 1)
else
   UTIL(max3t, cm3t, b3; c3, -1);


ONEWAY(b1, b2, b3, c1, c2, c3, con1, con2, gridtable, dir);
FILLARRAYS;
HOME_N_CLEAR;
BOXPIC;
HOME_N_CLEAR;
```

```
    JFRAME;
    continue := true;
    while continue do
    begin
       GETVIEW(pick);
       if pick <> 9 then
       begin
          PLOTGRAPH(pick, gridtable, 0);
          HOME_N_CLEAR;
       end
       else
          continue := false;
    end;
end; {UTIND1WAY}


{*************************************************************************}
{*                                                                       *}
{* BOTHDEP guides the process for drawing the mesh surface when neither  *}
{* attribute in utility independent of the other.                        *}
{*                                                                       *}
{*************************************************************************}
procedure BOTHDEP(var option : integer);
var
   i : integer;
begin
   HOME_N_CLEAR;
   for i := 1 to 5 do
      writeln;
   writeln('Since your responses indicate that the attributes are not at all');
   writeln('utility independent, there are three possible approaches to ');
   writeln('assessing utility values for the attributes. The first approach');
   writeln('requires you to rerun the utility program using transformed');
   writeln('attributes which may exhibit utility independence. The second');
   writeln('approach requires decomposing the attribute ranges and');
   writeln('then assessing the utility function over these decomposed');
   writeln('ranges, which may be utility independent. Further guidance on');
   writeln('this option is provided when you choose it. Finally, the ');
   writeln('last approach requires you to indicate your utilities directly ');
   writeln('for different combinations of attributes. Since this is the ');
   writeln('most difficult option to perform, it is recommended you try ');
   writeln('one of the other two if possible.');
   for i := 1 to 5 do
      writeln;
   writeln('Please indicate which option you want.');
   writeln('1) Use transformed attributes.');
   writeln('2) Decompose the attribute ranges.');
   writeln('3) Direct assessment.');
   write('Choice: ');
   readln(option);
   HOME_N_CLEAR;
end; {BOTHDEP}


{*************************************************************************}
{*                                                                       *}
{* WRITEVALUES finds the correct values and prints them in the appropri- *}
{* ate position for the 5 lotteries shown to the decision maker in DRAW- *}
```

```
{* CHECK.                                                                    *}
{*                                                                          *}
{**************************************************************************}
procedure WRITEVALUES(c : char; x, y : real; pos, ans, max, min :
                      parray);
var
   val1, val2 : parray;

begin
   if c = 'y' then
   begin
      val1 := max;
      val2 := pos;
   end
   else
   begin
      val1 := pos;
      val2 := max;
   end;
   PUTVAL(x, y, val1, val2);

   if c = 'y' then
   begin
      val1 := ans;
      val2 := pos;
   end
   else
   begin
      val1 := pos;
      val2 := ans;
   end;
   PUTVAL(x - 7.0, y - 10.0, val1, val2);

   if c = 'y' then
   begin
      val1 := min;
      val2 := pos;
   end
   else
   begin
      val1 := pos;
      val2 := min;
   end;
   PUTVAL(x, y - 20.0, val1, val2);

end; {WRITEVALUES}



{**************************************************************************}
{*                                                                        *}
{* DRAWCHECK shows the decision maker his previous answers for c.e.'s to  *}
{* help remedy possible inconsistencies.                                  *}
{*                                                                        *}
{**************************************************************************}
procedure DRAWCHECK(ch : char; table : tarray);
var
   i, val1, val2, val3, val4,
   val5, pos1, pos2, pos3, pos4,
```

```
    pos5, ans, pos, max, min : integer;
    x, y : real;
    spos, sans, smax, smin : parray;
begin
    if ch = 'y' then
    begin
        val1 := table[1].ans;
        pos1 := table[1].pos;
        val2 := table[2].ans;
        pos2 := table[2].pos;
        val3 := table[4].ans;
        pos3 := table[4].pos;
        val4 := table[7].ans;
        pos4 := table[7].pos;
        val5 := table[9].ans;
        pos5 := table[9].pos;
        max := table[3].pos;
        min := table[8].pos;

        write('(',val1:4,',  ',pos1:4,') ~ [0.5(',max:4,',  ',pos1:4,') ; ');
        writeln('0.5(',min:4,',  ',pos1:4,')]');
        writeln;
        write('(',val2:4,',  ',pos2:4,') ~ [0.5(',max:4,',  ',pos2:4,') ; ');
        writeln('0.5(',min:4,',  ',pos2:4,')]');
        writeln;
        write('(',val3:4,',  ',pos3:4,') ~ [0.5(',max:4,',  ',pos3:4,') ; ');
        writeln('0.5(',min:4,',  ',pos3:4,')]');
        writeln;
        write('(',val4:4,',  ',pos4:4,') ~ [0.5(',max:4,',  ',pos4:4,') ; ');
        writeln('0.5(',min:4,',  ',pos4:4,')]');
        writeln;
        write('(',val5:4,',  ',pos5:4,') ~ [0.5(',max:4,',  ',pos5:4,') ; ');
        writeln('0.5(',min:4,',  ',pos5:4,')]');
        writeln;
    end
    else
    begin
        val1 := table[3].ans;
        pos1 := table[3].pos;
        val2 := table[5].ans;
        pos2 := table[5].pos;
        val3 := table[6].ans;
        pos3 := table[6].pos;
        val4 := table[8].ans;
        pos4 := table[8].pos;
        val5 := table[10].ans;
        pos5 := table[10].pos;
        max := table[7].pos;
        min := table[1].pos;
        write('(',pos1:4,',  ',val1:4,') ~ [0.5(',pos1:4,',  ',max:4,') ; ');
        writeln('0.5(',pos1:4,',  ',min:4,')]');
        writeln;
        write('(',pos2:4,',  ',val2:4,') ~ [0.5(',pos2:4,',  ',max:4,') ; ');
        writeln('0.5(',pos2:4,',  ',min:4,')]');
        writeln;
        write('(',pos3:4,',  ',val3:4,') ~ [0.5(',pos3:4,',  ',max:4,') ; ');
        writeln('0.5(',pos3:4,',  ',min:4,')]');
        writeln;
        write('(',pos4:4,',  ',val4:4,') ~ [0.5(',pos4:4,',  ',max:4,') ; ');
        writeln('0.5(',pos4:4,',  ',min:4,')]');
```

```
      writeln;
      write('(',pos5:4,',  ',val5:4,') ~ [0.5(',pos5:4,',  ',max:4,') ;  ');
      writeln('0.5(',pos5:4,',  ',min:4,')]');
      writeln;

   end;

end; {DRAWCHECK}



{*************************************************************************}
{*                                                                       *}
{* EXPLAIN explains how to decompose a user's ranges so the properties of *}
{* utility independence may be used.  It also offers the user a chance to *}
{* review his previous answers to lotteries.                             *}
{*                                                                       *}
{*************************************************************************}
procedure EXPLAIN(var continue : boolean);
var
   i : integer;
   c : integer;
   dummychar : char;
begin
   for i := 1 to 5 do
      writeln;
   writeln('   Decomposing the attribute ranges should be done if there');
   writeln('is indication of utility independence over a subset of the range.');
   writeln('This is indicated by having a constant certainty equivalent for ');
   writeln('an attribute even when the other attribute''s value varies.  For ');
   writeln('example, if your certainty equivalent for attribute one (ranging');
   writeln('from, say, 0 to 100) is always constant (say, 60) over a subrange');
   writeln('(say, less than 350) of attribute two (ranging from, say, 200 to');
   writeln('400), then a good place to subset the range would be from 200 to');
   writeln('350 and 350 to 400.  This way, over the 200 to 350 range, some');
   writeln('utility independence properties will hold.');
   writeln;
   writeln('Would you like to see your original answers again? ');
   writeln('1) yes');
   writeln('2) no');
   write('Choice: ');
   readln(c);
   if c = 1 then
   begin
      HOME_N_CLEAR;
      writeln('These are the choices made for ',attribute1);
      writeln('by varying ',attribute2,' in terms of the');
      writeln('(',att1scale,',  ',att2scale,') outcome:  ');
      writeln;
      DRAWCHECK('y', lottable);
      Write('Hit any key followed by return to continue:  ');
      readln(dummychar);
      HOME_N_CLEAR;
      writeln('These are the choices made for ',attribute2);
      writeln('by varying ',attribute1,' in terms of the');
      writeln('(',att1scale,',  ',att2scale,') outcome:  ');
      writeln;
      DRAWCHECK('x', lottable);
      Write('Hit any key followed by return to continue:  ');
      readln(dummychar);
```

```
      HOME_N_CLEAR;
   end;
   writeln;
   writeln;
   writeln('Please indicate which operation you want to perform. ');
   writeln('1) Return to options menu.');
   writeln('2) Perform the analysis over a subset of ranges.');
   write('Choice: ');
   readln(i);
   if i = 1 then
      continue := false
   else
      continue := true;
end; {EXPLAIN}



{*************************************************************************}
{*                                                                       *}
{* DRAWSCALE draws the scale for the user to look at while assigning uti- *}
{* lity values to the outcomes.                                          *}
{*                                                                       *}
{*************************************************************************}
procedure DRAWSCALE;
var
   i : integer;
   a, x, y : real;
   c : char;

begin
   JMOVE(10.0, 20.0);
   JDRAW(90.0, 20.0);
   JMOVE(10.0, 25.0);
   JHSTRG(%stdescr '0');
   JMOVE(48.0, 25.0);
   JHSTRG(%stdescr '0.5');
   JMOVE(85.0, 25.0);
   JHSTRG(%stdescr '1.0');

   x := 10.0;
   y := 23.0;
   repeat
      JMOVE(x, y);
      JDRAW(x, y - 6.0);
      x := x + 8.0;
   until x = 98.0;
end; {DRAWSCALE}

{*************************************************************************}
{*                                                                       *}
{* PRESENT presents a user with 36 different outcomes to assign utility  *}
{* values to.                                                            *}
{*                                                                       *}
{*************************************************************************}

procedure PRESENT;
var
   gnum, pick,
   i, x, y, xstep, ystep : integer;
   xvalues, yvalues : array[0..5] of integer;
```

```
    utility : gtable;
    continue : boolean;

begin
    xstep := TRUNC((maximum1 - minimum1)/5);
    ystep := TRUNC((maximum2 - minimum2)/5);
    xvalues[0] := minimum1;
    yvalues[0] := minimum2;
    xvalues[1] := 0;
    yvalues[1] := 0;
    for i := 1 to 5 do
    begin
       xvalues[i] := xvalues[i - 1] + xstep;
       yvalues[i] := yvalues[i - 1] + ystep;
    end;
    INITGRAPHICS;
    JROPEN(1);
    JCOLOR(4);
    JFONT(3);
    JSIZE(2.0, 2.0);
    JMOVE(50.0, 35.0);
    DRAWSCALE;
    JRCLOS(1);
    HOME_N_CLEAR;
    for i := 1 to 5 do
       writeln;
    writeln('Based on the scale below, please enter the number you feel');
    writeln('indicates the worth of the following outcomes. The number ');
    writeln('entered may be.real or integer between and including 0 ');
    writeln('through 1.0.');
    for i := 1 to 3 do
       writeln;
    for y := 0 to 5 do
    begin
       writeln('(',attribute1,', ',attribute2,')');
       for x := 0 to 5 do
       begin
          write('(',xvalues[x],', ',yvalues[y],') : ');
          readln(utility[x + 1, y + 1]);  {save for mesh surface}
       end;
       HOME_N_CLEAR;
    end;
    ENDGRAPHICS;
    gnum := 1;
    continue := true;
    for i := 1 to 3 do
    begin
       wldmin[i] := 0.0;
       wldmax[i] := 5.0;
       at[i] := 2.5;
    end;
    at[3] := 5.0;
    wldmax[3] := 10.0;
    INITGRAPHICS;
    HOME_N_CLEAR;
    BOXPIC;
    HOME_N_CLEAR;
    JFRAME;
    while continue do
    begin
```

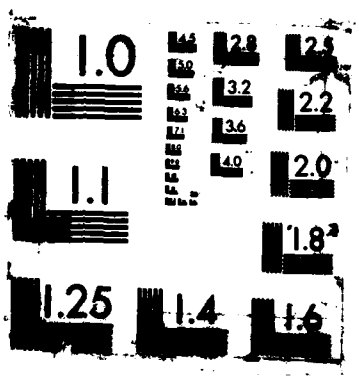
```
      GETVIEW(pick);
      if pick <> 9 then
      begin
         PLOTGRAPH(pick, utility, gnum);
         HOME_N_CLEAR;
      end
      else
         continue := false;
   end;
end; {PRESENT}


{*************************************************************************}
{*                                                                       *}
{* DIRECTASSESS introduces the direct assessment procedure.              *}
{*                                                                       *}
{*************************************************************************}
procedure DIRECTASSESS;
var
   i : integer;
   dummychar : char;
begin
   HOME_N_CLEAR;
   for i := 1 to 5 do
      writeln;
   writeln('     This procedure is difficult to perform because it is te-');
   writeln('dious, and you must have a good feel for your preferences. You');
   writeln('will be asked to assign a number from 0 to 1.0 to a variety of');
   writeln('outcomes.  Please think carefully about your choices. ');
   for i := 1 to 5 do
      writeln;
   write('Hit any key followed by return to continue: ');
   read_n(dummychar);
   PRESENT;
end; {DIRECTASSESS}




{*************************************************************************}
{*                                                                       *}
{* TREND is used to uncover inconsistencies in a decision maker's re-    *}
{* sponses. Possible inconsistencies are perceived to exist when all re- *}
{* sponses are close to, but not the same as, each other.                *}
{*                                                                       *}
{*************************************************************************}
function TREND(v, w, x, y, z : integer):boolean;
const
   point5 = 0.05;
var
   total, avg, fiveper : integer;
begin
   TREND := true;
   total := v + w + x + y + z;
   avg := ROUND(total/5);
```

```
   fiveper := ROUND((avg * point5));
   if ((ABS(v - avg) < fiveper) and (ABS(w - avg) < fiveper) and
      (ABS(x - avg) < fiveper) and (ABS(y - avg) < fiveper) and
      (ABS(z - avg) < fiveper)) then
         TREND := false;
end; {TREND}


{*************************************************************************}
{*                                                                       *}
{* CONSISCHECK reconfirms a decision maker's c.e.'s if inconsistencies   *}
{* are indicated.  The decision maker may change the c.e.'s to a common  *}
{* value if desired, or, in the case of a trend, they may be left as is. *}
{*                                                                       *}
{*************************************************************************}
procedure CONSISCHECK(c : char;var choyse : integer;var tab : tarray);
var
   dummychar : char;
   val, i : integer;
   dumatt, dumscale : attribute;
begin
   if c = 'y' then
   begin
      dumatt := attribute1;
      dumscale := attiscale;
   end
   else
   begin
      dumatt := attribute2;
      dumscale := att2scale;
   end;
   HOME_N_CLEAR;
   for i := 1 to 5 do
      writeln;
   writeln('   In terms of the (',attiscale,', ',att2scale,') outcome, since ');
   write('your responses for ',dumatt,' all fall within ');
   writeln('5 percent of the average,');
   writeln('please carefully reconsider your previous answers. You will');
   writeln('need to determine if the attribute values should be changed to a');
   writeln('common value, and if so, which one. (NEXT SCREEN)');
   for i := 1 to 10 do
      writeln;
   write('Hit any key followed by return to continue: ');
   readln(dummychar);
   HOME_N_CLEAR;
   DRAWCHECK(c, tab);
   writeln('Please enter the number corresponding to your choice. ');
   writeln('  1) Change previous answers to a common one.');
   writeln('  2) Leave answers as is.');
   write('Choice: ');
   readln(choyse);
   if choyse = 1 then
   begin
      write('What value would you like to change it to?: ');
      readln(val);
      if c = 'y' then
      begin
         tab[1].ans  := val;
         tab[2].ans  := val;
         tab[4].ans  := val;
```

```
            tab[7].ans := val;
            tab[9].ans := val;
        end
        else
        begin
            tab[3].ans := val;
            tab[5].ans := val;
            tab[6].ans := val;
            tab[8].ans := val;
            tab[10].ans := val;
        end;
    end;
end; {CONSISCHECK}


{*************************************************************************}
{*                                                                       *}
{* ASSESSIND assesses which independence properties hold.                *}
{*                                                                       *}
{*************************************************************************}
procedure ASSESSIND(var t : tarray);
var
    choice : integer;
    way : char;
    y_utind_z, z_utind_y, addind, mutind : boolean;
    scaley, scalez : real;
begin
    y_utind_z := false;
    z_utind_y := false;
    mutind := false;
    addind := false;
    if ((t[1].ans = t[2].ans) and (t[4].ans = t[7].ans) and
        (t[9].ans = t[1].ans) and (t[4].ans = t[1].ans)) then
            y_utind_z := true;


    if ((t[3].ans = t[5].ans) and (t[6].ans = t[8].ans) and
        (t[10].ans = t[3].ans) and (t[6].ans = t[3].ans)) then
            z_utind_y := true;


    if not y_utind_z then
    begin
        if not TREND(t[1].ans, t[2].ans, t[4].ans, t[7].ans, t[9].ans)
        then
            CONSISCHECK('y', choice, t);
        if choice = 1 then
            y_utind_z := true;
    end;
    if not z_utind_y then
    begin
        if not TREND(t[3].ans, t[5].ans, t[6].ans, t[8].ans, t[10].ans)
        then
            CONSISCHECK('z', choice, t);
        if choice = 1 then
            z_utind_y := true;
    end;
    if (y_utind_z and z_utind_y) then
    begin
```

AD-A187 276 DEVELOPMENT OF A GRAPHICS BASED TWO-ATTRIBUTE UTILITY ASSESSMENT PROGRAM W..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH E H WANNER DEC 86 3/3 DATA

UNCLASSIFIED AFIT/CI/NR-87-73T F/G 12/4 NL


1.0
2.8
3.2
3.6
4.0
2.2
2.0
1.8
1.1
1.25
1.4
1.6

```
      mutind := true;
      CHECKADDIND(t, addind);
   end;
   if addind then
      MUTORADD('a', t, yb, yc, zc, zb, scaley, scalez)
   else
      if mutind then
         MUTORADD('m', t, yb, yc, zc, zb, scaley, scalez)
      else
         if y_utind_z or z_utind_y then
         begin
           if y_utind_z then
              way := 'y'
           else
              way := 'z';
           UTINDiWAY(way);
         end
         else
           BOTHDEP(rerun_option);
   ENDGRAPHICS;
end; {ASSESSIND}


                    {MAIN PROGRAM}

begin
   WELCOME;
   rerun_option := 5;  {rerun option of 5 means first time through}
restart:  {label for place to return when restarting}
   if rerun_option = 2 then  {it will if decomposition of ranges is needed}
      EXPLAIN(go_on);
   if (go_on) or (rerun_option = 1) or (rerun_option = 5) then

   {either decomposition has been explained and the user wishes to continue }
   {(go_on is true), or the user wants to use different attributes       }
   {(rerun_option is 1), or this is the first time through (rerun_option }
   { is 5).                                                              }

   begin
     ATTRIBUTES(attribute1, attribute2, minimum1, maximum1, minimum2, maximum2);
     DEVTABLE(attribute1, attribute2, minimum1, maximum1, minimum2, maximum2,
              lottable);
     LOTTERIES(attribute1, attribute2, minimum1, maximum1, minimum2, maximum2,
              lottable);
     ASSESSIND(lottable);

     {ASSESSIND will determine the independence properties.  If none are  }
     {present, the user will be asked if he wants to redo the program.  If }
     {so, rerun_option will be either 1 or 2.                              }

     if (rerun_option = 1) or (rerun_option = 2) then
        goto restart;
   end
   else
   if not go_on then
   begin
      BOTHDEP(rerun_option);
      if (rerun_option = 1) or (rerun_option = 2) then
         goto restart;
```

```
    end;
    if rerun_option = 3 then
       DIRECTASSESS;
end.
```

```
.SBTTL HOME_N_CLEAR
ESC = 27

HOME_N_CLR :    .BYTE ESC
                .BYTE 37
                .BYTE 33
                .BYTE 49

                .BYTE ESC
                .ASCII '[1;1f'
                .BYTE ESC
                .ASCII '[0J'

                .BYTE ESC
                .BYTE 37
                .BYTE 33
                .BYTE 48

.PSECT NONSHARED_DATA LONG, NOEXE, PIC
IO_CHAN : .LONG 0
     TT : .ASCID 'TT'
   IOSB : .BLKQ 1
          .PSECT CODE LONG, PIC
          .ENTRY HOME_N_CLEAR, ^M<R2, R3, R4, R5, R6, R7, R8, R9, R10>

$ASSIGN_S CHAN = IO_CHAN, DEVNAM = TT
$QIOW_S CHAN = IO_CHAN,-
        FUNC = #IO$_WRITEVBLK,-
        IOSB = IOSB,-
          P1 = HOME_N_CLR,-
          P2 = #18, -
          P4 = #0
$DASSGN_S CHAN = IO_CHAN
RET
.END
```

END

DATE

FILMED

FEB.

1988