AN INTERACTIVE COMPUTER PROGRAM FOR ASSESSING AND ANALYZING PREFERENCES CONCERNING MULTIPLE OBJECTIVES

Ralph L. Keeney
Alan Sicherman

April 1975

Research Memoranda are informal publications relating to ongoing or projected areas of research at IIASA. The views expressed are those of the authors, and do not necessarily reflect those of IIASA.

An Interactive Computer Program for Assessing and Analyzing Preferences Concerning Multiple Objectives*

Ralph L. Keeney† and Alan Sicherman‡

Abstract

An interactive computer program designed to facilitate the quantification of a decision maker's preferences for multiple objectives in terms of a multiattribute utility function is described. It is meant to alleviate many of the operational difficulties with current procedures for assessing and using multiattribute utility functions. The package includes commands for structuring the utility function, assessing single-attribute component utility functions of the overall multiattribute utility function, identifying the preference trade-offs between attributes, evaluating alternatives, and performing sensitivity analysis. Suggestions for using the program are included.

Preface

The program described in this paper is currently available for the use of IIASA at IBM in Vienna. If interested, please contact Ralph L. Keeney.

---

*Most of the work reported here was conducted while both authors were at the MIT Operations Research Center, Cambridge, Massachusetts, USA.

†International Institute for Applied Systems Analysis, Laxenburg, Austria.

‡Operations Research Center, MIT, Cambridge, Massachusetts, USA.

1. Introduction

Many complex decision problems have the characteristic of being multiple objective in nature. Inevitably, these multiple objectives are conflicting objectives in the sense that, once dominated alternatives have been eliminated, further achievement in terms of one objective can occur at the expense of some achievement of another objective. Thus, in evaluating potential alternatives, the decision maker must consider his preference trade-offs between various degrees of achievement of one objective and degrees of achievement of others. The real problems are even more complicated because uncertainty is usually present. That is, one cannot predict with certainty what the consequences of each of the alternatives under consideration will be.

In evaluating alternatives, it is very difficult to logically and consistently consider the above complexities informally in the mind. Hence there is a need for formal analysis. Decision analysis is an approach which does explicitly address the multiple objective and uncertainty issues. The theoretical basis for this is well established. However, an important practical problem concerns quantifying the decision maker's preference structure for multiple objectives. Without this mathematical representation--called a utility function--of the decision maker's preferences one cannot formally evaluate the alternatives.

This paper describes an interactive computer package designed to facilitate the assessment and use of a decision maker's utility function for multiple objectives. At present, some of the subroutines in the package are rather crude. However, the package is currently operational and does overcome many of the major difficulties previously experienced in

assessing and using utility functions in complex problems.

1.1 Decision Analysis

By briefly outlining the decision analysis approach, we hope to motivate the work described here and place it properly in a broader context. Raiffa [10] discusses the philosophy and techniques of decision analysis in detail. For our purposes, let us categorize it with four steps:

1) structuring the problem,

2) quantifying the uncertainties involved,

3) quantifying the decision maker's preferences,

4) evaluating the alternatives.

Structuring includes problem specification and identification of the decision maker. The decision maker must articulate his objectives and attributes (i.e. measure of effectiveness) for each objective. An attribute is a measurement scale used to indicate the degree to which the corresponding objective is achieved. The alternatives must also be specified. Let us designate our set of attributes as $X_1, X_2, \ldots, X_n$ and use x_i to indicate a specific amount of attribute X_i. For instance, X_1 may designate profit in 1975 measured in thousands of dollars and x_1 may be 188. With this convention, the consequence of any alternative is $\underline{x} \equiv (x_1, x_2, \ldots, x_n)$.

Quantifying uncertainties involves describing the uncertainty about the possible consequences of each alternative. For each alternative A_j, a probability distribution $p_j(\underline{x})$ indicating which consequences might occur and their likelihood is required. The p_j may be specified using any combination of analytical models, simulation models, subjective assessments, and data that is available and appropriate.

Quantifying preferences means assessing the decision maker's utility function $u(\underline{x}) \equiv u(x_1, x_2, \ldots, x_n)$, which is called a multiattribute utility function since the argument of the utility function is a vector indicating levels of the several attributes. The multiattribute utility function, which will be referred to by the mnemonic MUF, has two properties which make it useful in addressing the issues of uncertainty and trade-offs between objectives. These properties are:

1) $u(\underline{x}') > u(\underline{x}'')$ if and only if $\underline{x}'$ is preferred to $\underline{x}''$, and
2) in situations with uncertainty, the expected value of u is the appropriate guide to make decisions; i.e., the alternative with the highest expected value is the most preferred.

This second property follows directly from the axioms of utility theory postulated first in von Neumann and Morgenstern [15].

Evaluating alternatives involves calculating the expected utility of each of the alternatives and conducting sensitivity analysis. Given p_j for each A_j and u from the previous steps, the expected utilities for the alternatives can be evaluated. To gain additional confidence and insight into which alternative should be chosen and why, various parameters in both the probability distributions and the utility function can be varied to see how these affect the expected utility of the alternatives.

1.2 Statement of the Problem

The weakest link of the four above steps in rendering decision analysis operational for multiple objective problems is quantifying the decision maker's preferences. Defining the problem is common to all attempts to systematize the decision making process. Quantifying uncertainties has also been widely addressed in modeling efforts. The outputs of many simulation models include probability distributions over the

relevant attributes for each of the alternatives under consideration. However, the decision maker is usually required to review these outputs--informally combining them with his preferences--to select an alternative. Because multiattribute utility theory was only recently developed [1,2,5,6,8,11] and because the operational procedures to put it into practice are not well developed, the third and fourth steps are informally carried out simultaneously. The critical step is actually the quantification of preferences because, as indicated above, evaluation of alternatives is fairly straightforward once probabilities and preferences are quantified.

Much of multiattribute utility theory is developed as follows. Assumptions about the decision maker's preferences are postulated, and the restrictions these assumptions place on the functional form of the utility function are derived. Then, for any specific problem, the appropriateness of the assumptions for a particular MUF should be verified with the decision maker and parameters for the utility function assessed and checked for internal consistency. Ideally, the functional form of the MUF would have the following properties:

1) be general enough to allow application to many real problems,
2) require a minimal number of assessment questions to be asked of the decision maker,
3) require assessments which are reasonable for a decision maker to consider,
4) be easy to use in evaluating alternatives and conducting sensitivity analyses.

Even with a convenient functional form for the MUF, the nature and magnitude of a problem can make the bookkeeping and use of quantitative assessments a formidable task. The computer package described in this paper is designed to handle this task for a variety of problem contexts.

1.3 Organization of the Paper

Section 2 summarizes the theoretical development of the functional forms of the MUF's upon which the computer package is based. Section 3 discusses existing methods and their difficulties for assessing and using these MUF's. A description of the computer package and the manner in which it alleviates such difficulties is in Section 4. Section 5 describes an application of the package to an important "typical" multiple objective problem, followed by suggestions for using and improving the package. The Appendix briefly describes the program commands.

2. The Additive and Multiplicative Utility Functions

Conditions which imply that a MUF is either additive or multiplicative are very similar. None of the conditions require the decision maker to consider preference trade-offs among more than two attributes simultaneously or to consider lotteries (specifying various levels of $\underline{x}$ and the probabilities of receiving them) with the level of more than one attribute being varied. Furthermore, the assessments needed to specify an n-attribute utility function are n one-attribute utility functions and n scaling constants.

2.1 The Basic Assumptions

The two basic assumptions which we use for both additive and multiplicative utility functions are referred to as preferential independence and utility independence. These are defined as follows:

Preferential Independence: The pair of attributes $\{X_1, X_2\}$ is preferentially independent of the other attributes $\{X_3,\ldots,X_n\}$ if preferences among $\{X_1, X_2\}$ pairs given that $\{X_3,\ldots,X_n\}$ are held fixed, do not depend on the level where $\{X_3,\ldots,X_n\}$ are fixed.

Preferential independence implies that the trade-offs between attributes X_1 and X_2 do not depend on $X_3,\ldots,X_n$.

Utility Independence: The attribute X_1 is utility independent of the other attributes $\{X_2,\ldots,X_n\}$ if preferences among lotteries over X_1, (i.e. lotteries with uncertainty about the level of X_1 only) given $X_2,\ldots,X_n$ are fixed, do not depend on the level where those attributes are fixed.

The main result can now be stated.

Theorem 1. For $n \geq 3$, if for some X_i, $\{X_i, X_j\}$ is preferentially independent of the other attributes for all $j \neq i$ and X_i is utility independent of all the other attributes, then either

$$u(\underline{x}) = \sum_{i=1}^{n} k_i u_i(x_i) \quad , \tag{1}$$

$$1 + ku(\underline{x}) = \prod_{i=1}^{n} [1 + kk_i u_i(x_i)] \quad , \tag{2}$$

where

u and u_i are utility functions scaled from zero to one,

the k_i's are scaling constants with $0 < k_i < 1$, and

$k > -1$ is a non-zero scaling constant satisfying the equation

$$1 + k = \prod_{i=1}^{n} (1 + kk_i) \quad . \tag{3}$$

The proof of this result is found in Keeney [4]. Alternative sets of assumptions leading to either form (1) or (2) are found in Fishburn [1], Pollak [8], and Meyer [6]. The functional form (1) is referred to as the additive utility function and (2) is the multiplicative utility function. For the case of two attributes, the following is proved in Keeney [5]:

Theorem 2. For $n = 2$, if X_1 is utility independent of X_2 and X_2 is utility independent of X_1, then the utility function $u(x_1,x_2)$ is either additive or multiplicative.

Using either (1) or (2), if $\sum_{i=1}^{n} k_i = 1$, the utility function is additive, and if $\sum_{i=1}^{n} k_i \neq 1$, it is multiplicative. When $\sum_{i=1}^{n} k_i > 1$, then $-1 < k < 0$, and when $\sum_{i=1}^{n} k_i < 1$, then $0 < k < \infty$. To use either the additive or multiplicative form, we need to obtain exactly the same information. We have to assess the n single-attribute utility functions $u_i(x_i)$ and the n scaling constants k_i. How this information is obtained and used is the subject of Sections 3 and 4.

2.2 Nesting Utility Functions

The results above are valid regardless of whether the X_i's are scalar attributes or vector attributes. This means that the x_i's can be either scalars or vectors. In the former case, the component utility functions u_i are single-attribute utility functions, whereas in the latter case, u_i is itself a multiattribute utility function. If X_i is a vector attribute, it is possible, subject to satisfying the requisite assumptions, to use Theorems 1 and 2. In such a case, we will say u_i is a nested MUF. That is, u_i is a MUF nested within the MUF u. Our interest in nesting utility functions is that it provides more general utility functions which are still tractable enough to assess and use.

2.3 Applicability of the Functional Forms

In terms of the required assessments and general robustness, the additive and multiplicative utility functions appear to be the practical ones for say $n \geq 4$. Even when the requisite assumptions do not precisely

hold over the domains of all the attributes, it may be a good approximation to assume they do, or it may be reasonable to integrate different additive and multiplicative utility functions over separate regions of these attributes. Furthermore, by nesting one MUF inside another, additional flexibility in the preference structure can be achieved.

The effect of nesting multiplicative forms is to create an extra degree of freedom in the problem by having an extra independent scaling constant. Without nesting, the number of independent scaling constants is equal to the number of single attributes. However, suppose u_n is a MUF nested within u and that u_n has three single attributes. Then one would need n scaling constants for the "outer MUF" and three for the "inner MUF" for a total of n + 3, even though there are only n + 2 single attributes, $X_1,\ldots,X_{n-1}$ and the three single attributes in u_n. The degree of freedom afforded by the extra parameter permits trade-offs between two attributes to be dependent on a third. This allows for some violation of the preferential independence conditions. By various nesting schemes, enough extra constants could be provided to model situations in which trade-offs between many pairs of attributes depend on the level of other attributes.

In the case of utility independence violations, the particular problem may be far more sensitive to the scaling constants or trade-offs among the attributes than to the conditional single-attribute utility function variations. Thus even in these cases, the additive or multiplicative form may provide an adequate model for the problem.

In summary, the additive and multiplicative utility functions are simple enough to be tractable and yet, especially with nesting, robust enough to adequately quantify preferences for many problems. In practice, however, assessing and using such MUF's is "easier said than done."

3. Difficulties with Existing Methods for Assessment and Use

Aspects of the state-of-the-art for assessing and using MUF's are discussed in this section. Some of the important shortcomings of existing procedure are identified. These include:

1) the necessity to ask "extreme value" questions to keep the computational requirements for specifying a utility function to a manageable level,
2) the tedium of calculating the component utility functions and scaling constants even in this case,
3) the lack of immediate feedback to the decision maker of the implications of his preferences,
4) the absence of an efficient procedure to "update" the decision maker's preferences and conduct sensitivity analysis.

In the discussion that follows, we will assume that the assumptions for the MUF to be either additive or multiplicative have been verified.

3.1 Specifying the Utility Functions over the Single Attributes

Techniques for assessing single-attribute utility functions have become fairly standard (Raiffa [10], Schlaifer [12]), and sophisticated computer programs have been developed for fitting single-attribute utility functions (Meyer and Pratt [7], Schlaifer [13]). Such programs provide quick feedback to allow the decision maker to check if his assessments and their implications appear reasonable. There is a difficulty in using these programs interactively in assessing multiattribute utility functions, since at present they do not exist in conjunction with a multiattribute utility assessment package. This minor shortcoming can be easily remedied.

3.2 Assessing the Trade-offs Among Attributes

The issue of trade-offs among the attributes is addressed by assessing the k_i's in the utility functions (1) and (2). In theory the manner of doing this is very simple. If there are n attributes, we want to assess the n unknown k_i's by creating n independent equations with the n unknowns and solving. An equation is created by i) having the decision maker indicate two options, where an option is either a consequence or a lottery, between which he is indifferent, and ii) equating the expected utilities of these options using either (1) or (2). For instance, if the decision maker finds $\underline{x}'$ and $\underline{x}''$ indifferent, then $u(\underline{x}') = u(\underline{x}'')$ provides one equation with at most n unknowns.

Manually solving n equations, which are not necessarily linear, with n unknowns is, to say the least, tedious. Current practice in assessing the k_i's usually requires sets of equations which are simple to evaluate. This basically limits the questions to two types. To indicate these, let us define $\underline{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ and $\underline{x}^o = (x_1^o, x_2^o, \ldots, x_n^o)$ as the most desirable and least desirable consequences. Then, because of the scaling conventions given in Theorems 1 and 2,

$$u(\underline{x}^*) = 1 \quad , \qquad u(\underline{x}^o) = 0 \quad , \tag{4}$$

and

$$u_i(x_i^*) = 1 \quad , \qquad u_i(x_i^o) = 0 \quad , \qquad i = 1,2,\ldots,n \quad . \tag{5}$$

One type of practical question can be illustrated as follows:

Question I. For what probability p are you indifferent between

i) the lottery giving a p chance at $\underline{x}^*$ and a 1 - p chance at $\underline{x}^o$, and

ii) the consequence $(x_1^o, \ldots, x_{i-1}^o, x_i^*, x_{i+1}^o, \ldots, x_n^o)$.

If we define the decision maker's answer as p_i, then using (4), the expected utility of the lottery is p_i, and using either (1) or (2), the utility of the consequence is k_i. Equating the expected utilities, we find

$$k_i = p_i \quad . \tag{6}$$

One could then clearly generate the values of each of the k_i's in this fashion.

The second type of question is illustrated by:

Question II. Select a level of X_i, call it x_i', and a level of X_j, call it x_j', such that, for any fixed levels of all the other attributes, you are indifferent between

i) a consequence yielding x_i' and x_j^o together, and

ii) a consequence yielding x_j' and x_i^o together.

Using (5) and either the multiplicative or additive utility function, the utilities of these two indifferent consequences can be equated to yield

$$k_i u_i(x_i') = k_j u_j(x_j') \quad . \tag{7}$$

Once the single attribute utility functions u_i and u_j are assessed, both $u_i(x_i')$ and $u_j(x_j')$ are easily found, so (7) is a simple linear equation. Suppose in addition, for example, that $x_i' = x_i^*$. Then by (5), the relationship between k_i and k_j given by (7) is even simpler.

A major shortcoming of questions of both types I and II is the use of the extreme levels of the attributes, that is the x_i^* and x_i^o. Since the range from x_i^o to x_i^* must cover the range for x_i, the implications of, and hence preferences for, the extreme levels are usually very difficult for a decision maker to assess. A further difficulty with Question I is the fact that the effect due to varying all n attributes

simultaneously must be considered. Hence for computational ease we must force the decision maker to respond to questions much more difficult to evaluate than would be theoretically necessary.

A common practice in assessing the k_i's would be to use a question I to evaluate the largest k_i, and then use type II questions to evaluate the magnitude of the other k_j's relative to the largest k_i. Once we have the k_i's, the additive form must hold if they sum to one. Otherwise, the k_i's are substituted into (3) to evaluate k for the multiplicative form. This task in itself can be time consuming using only a calculator.

3.3 Evaluating Alternatives and Sensitivity Analysis

Manual calculations are clearly impractical for evaluating alternatives. With uncertainty, we need to evaluate the expected value of u using the probability distribution describing the possible consequences. Even with probabilistic independence among the X_i's, the computational task is large. It is also clear that sophisticated sensitivity analyses are out of the question without major computational help.

On the other hand, it is a large requirement to develop a special computer program to accommodate a particular problem. Such programming is often inflexible because of the special nature of the situation for which it was done. For instance, it would usually be very difficult to add additonal attributes, to try different "nesting" schemes, or to explore the preference structure for "hints" of creative new alternatives to generate.

4. The Computer Package

This section describes the major features of a computer package designed to alleviate some of the shortcomings with existing methods for the assessment and use of multiattribute utility functions. The package

is referred to by the mnemonic MUFCAP standing for "multiattribute utility function calculation and assessment package." Steps customarily followed in obtaining and using a MUF are presented with a description of the MUFCAP commands appropriate in performing the particular step. For illustration, the multiplicative form will be used for both the overall utility function u and any nested MUF's. However, MUFCAP employs the additive utility function, rather than the multiplicative form, in problems where it is appropriate. A complete summary of the package and listing of the program are found in Sicherman [14]. A list of the package commands is given in the Appendix.

4.1 Commands to Structure the Utility Function

Structuring a utility function consists of specifying a functional form, its attributes, and the ranges for each of the attributes. MUFCAP has several commands for structuring a preference function. The INPUT command requests a name for the utility function and asks for the number of attributes which are arguments of this function. The package then requests a name and a range for scalar attributes. This consists of two numbers which bound the amounts to be considered for each attribute. To specify a vector attribute, one inputs a range with one bound equal to the other bound such as 0,0. MUFCAP recognizes this as a signal for a vector attribute and notes that the u_i associated with that attribute is a nested MUF. The package then requests the number of attributes which are arguments of this nested MUF. For each of these, a name and range will be solicited. Further levels of nesting could be specified if desired and the information requested would be analogous to the material above. After a nested MUF is completely specified, the program returns to ask for the names and ranges for whatever attributes have not yet been covered in the outer MUF. When all the attributes have been input, the structure is complete and MUFCAP requests a new command from the user.

The INPUT command provides for all the bookkeeping which will be necessary for information to follow. Each k_i and u_i, including those in a nested MUF, can be accessed using the name of the attribute with which it is associated. The INPUT command is quite flexible in having no limit to the degree of nesting allowed.

In addition to INPUT, the package has commands for adding or deleting attributes to or from the utility function. It also has a command for switching the order of the attributes in a utility function. In this way, attributes may be conveniently "regrouped" to alter the model for the problem in terms of different nesting schemes.

4.2 Commands to Specify the Single Attribute Utility Functions

The next step in assessing a MUF involves specifying the u_i's for the single attributes. As noted in Section 3, sophisticated computer programs do exist for assessing single (scalar) attribute utility functions. One could incorporate these into MUFCAP. Initially, however, simpler routines for assessing unidimensional utility functions, referred to as UNIF's, were developed.

MUFCAP has available commands to specify conveniently three UNIF types: linear, exponential, and piecewise linear. Pratt [9] considers the implications of these forms. The linear utility function implies risk neutrality. This form requires no more information than the range of the attribute. The exponential form implies constant risk aversion or constant risk proneness. It requires the specification of a certainty equivalent for a single lottery. Given this, the exponential form is fitted and scaled automatically by the program. The piecewise linear utility function is specified by providing the abscissa and ordinate values for n points ($3 \leq n \leq 15$) of the utility function. This form

can be used for non-monotonic or S-shaped utility functions. These three types provide the user with the means of specifying a UNIF appropriate for many situations. More forms can easily be added to the package in the future.

MUFCAP also has commands which enable a user to quickly display the assessed UNIF for purposes of checking its appropriateness. The command UNICAL calculates the utility for one or a series of attribute levels. INVERSE calculates the attribute level corresponding to a given utility. LOTTERY evaluates the certainty equivalent for any lottery with n consequences and their associated probabilities over that attribute, where $2 \leq n \leq 15$. When there are two consequences, LOTTERY can also calculate the probability which will make the lottery indifferent to a given certainty equivalent.

To summarize, MUFCAP has convenient commands to assess u_i's which are UNIF's and to examine their implication as a check on their reasonableness.

4.3 Commands to Specify the Scaling Constants

Using the attribute names as identifiers, MUFCAP allows the user to set the scaling constants in the MUF corresponding to each attribute. If X_i is a vector attribute, the u_i associated with it is a MUF with its own internal scaling constants. By referring to the name of this vector attribute, the user can specify the internal scaling constants for the associated nested MUF. When all the k_i's for a particular MUF have been set, the program automatically calculates the corresponding k using (3).

Once u_i's have been evaluated, the package has several commands useful for assessing the k_i's in any particular MUF. The command INDIF2

takes as input two indifference pairs, each consisting of two indifference consequences. These consequences can vary only in terms of the two attributes, say X_j and X_m, whose k_i's are the object of assessment. Using (2), the program equates utilities of the indifferent consequences and computes the relative value of k_j and k_m implied by the indifference pairs. With INDIF2, the user is not limited to choosing consequences which have one attribute at a least desirable level in order to determine the relative k_i's.

Given the information from INDIF2, indifference curves over X_j and X_m can be calculated with the command IMAP. IMAP permits a user to get immediate feedback on the implications of the relative k_i's which he has specified. He can quickly see if the points "claimed" to be indifferent really appear so to him. If not, the relative k_i's can be changed until they represent the user's preferences for trade-offs between those attributes.

Once we know the relative k_i's, the command INDIF1 takes as input a single pair of indifference consequences and computes the k and the absolute magnitude of the k_i's implied by that pair and the relative k_i's. For consistency checks, a new indifference pair of consequences can be input into INDIF1, which then computes the factor by which the current k_i's need to be multiplied to be consistent with the indifference point just given. MUFCAP provides a routine which allows the user to multiply the currently assigned k_i's for any MUF by any factor. In this way, INDIF1 enables the calculation of the magnitude of the k_i's using an indifference relation instead of a lottery over all the attributes at once.

4.4 Commands for Evaluating Alternatives and Sensitivity Analysis

Once the u_i's and k_i's have been set, the utility function is completely specified and can be used. To help explore the implications

of the utility function and to perform "rough" analysis, MUFCAP has commands for specifying two kinds of alternatives: certain and uncertain. For certain alternatives, which are simply consequences, uniattribute amounts are solicited until the alternative is completely described. For uncertain alternatives, at present, MUFCAP assumes probabilistic independence and requests a probability distribution function for each single attribute. The probability distribution function currently used is a piecewise linear approximation to the cumulative probability distribution for X_i. The user supplies n abscissa-ordinate pairs, where $2 \leq n \leq 9$ to specify the cumulative distribution. Then MUFCAP calculates the expected utilities for probabilistic alternatives. The cumulative distribution was chosen rather than the probability density function because the fractile method of assessing probabilities (see Schlaifer [12]) yields points of the cumulative distribution. Other forms of probability distributions such as the Gaussian as well as probabilistic dependencies could be added to the package in the future.

The specified alternatives are given names by the user. With these names, the user may add, change or delete alternatives. He may also choose the ones which are to be evaluated by listing their names with the appropriate commands about to be described.

The command EVAL is used to evaluate (i.e. compute the expected utility for) any alternative or group of alternatives. By specifying a group of alternatives differing slightly in some feature, one can conduct a sensitivity analysis of the probabilistic inputs. Also, EVAL will compute the expected utilities for any multiattribute utility function specified in the command. Thus, using EVAL, one can conduct a sensitivity analysis of the preference structure by varying parameters,

such as the scaling constants, in the multiattribute utility function. In this same way, different utility functions of members of a decision making group can be used to evaluate and rank the alternatives. This might help clarify differences of opinion and suggest certain creative compromises if needed.

The command GRAD evaluates the gradient of a utility function at any number of specified consequences. The gradient is defined as the vector $\left(\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots, \frac{\partial u}{\partial x_n}\right)$ and indicates the direction of steepest increase in the utility function at a specified point. The gradient components tells us which attribute level changes would yield large increases in utility. This could be useful in generating worthwhile alternatives. Of course, one must keep in mind the scales of the attributes in interpreting the gradient.

In addition to the gradient, GRAD also computes the vector $\left(\frac{\partial u}{\partial u_1}, \frac{\partial u}{\partial u_2}, \ldots, \frac{\partial u}{\partial u_n}\right)$. Each component represents the rate of change of u with respect to a change in the utility u_i. These components reveal the attributes for which an increase in its utility will yield the largest increase in u. The advantage of calculating these quantities in addition to the gradient components are a) components can be calculated for MUF's as well as UNIF's, and b) the unit of measurement for a uniattribute does not distort the magnitude of the component. Thus in some cases, $\frac{\partial u}{\partial u_i}$ might better indicate possible improved alternatives than $\frac{\partial u}{\partial x_i}$. MUFCAP makes both available.

Summarizing, EVAL permits the evaluation of alternatives, and along with routines which alter parameters, provides for sensitivity analysis. GRAD makes use of the analytical formulation of the problem to calculate quantities useful in suggesting alternatives which might be better than the ones currently specified.

4.5 General Command Format and Commands for Facilitating Use of the Package

MUFCAP commands are designed to be concise and are for the most part no longer than three words. These words may initiate a dialogue when more information is necessary. The input format is free, i.e. words need not begin in a particular position on the page. For many commands, the user will be prompted if he has left out a necessary word.

Mistyping causing invalid numbers on input is handled automatically by the program and a correct number is requested. Provision is made for the user to terminate a lengthy dialogue by specifying the word QUIT for the next number to be input. A new command can then be entered. In the future, a help command could be easily implemented which would explain the syntax of any other command, give definitions of terms used in the program and make suggestions concerning what kinds of steps to perform in assessing and using the MUF.

In addition to these features, MUFCAP has the facility for saving the current state of the multiattribute utility structure and the current alternatives in a file of the user's choosing to be read in at a later time. This gives MUFCAP the capability for filing away several different MUF models as well as a large number of alternatives for the same problem. It also allows the user to build up his model over many different sessions at the terminal and restore any status he has saved away with which he wishes to calculate at any particular time.

Another feature of MUFCAP is the supplying of default settings when the INPUT command is used to structure the MUF for the problem. After INPUT, the default for all MUF's is the additive form, with all the k_i's equal to each other, and for all UNIF's, it is the linear utility function. With these defaults, the user is set to calculate immediately after input. Thus feedback can begin right away without requiring the user to completely specify everything first. Scaling constants and utility functions can then be alterred after observing some feedback to refine the model for the problem.

Finally, MUFCAP provides commands to print out the current status of the assessments. There are routines to display the k_i's and k for any MUF, the range and type for any single attribute utility function, the probability distribution of any attribute for any alternative, multiattribute utility function structure (i.e. nesting) and the currently defined alternatives. Commands are also provided for easily changing parameters such as individual k_i's or the components of any alternative.

5. A Simulated Application of MUFCAP: The Mexico City Airport

This section briefly illustrates how MUFCAP could be used in practice. An application chosen is that of the Mexico City Airport described in Keeney [3]. This problem was approached using the existing methods for MUF assessment and calculation and utilized special computer programming to aid in the calculations. This section presents what might have been done if MUFCAP had been available then.

5.1 Attributes for the Problem

The Mexico City Airport problem was defined in terms of the following attributes:

$X_1 \equiv$ total cost in millions of pesos,

$X_2 \equiv$ the capacity in terms of the number of aircraft operations per hour,

$X_3 \equiv$ access time to and from the airport in minutes,

$X_4 \equiv$ number of people seriously injured or killed per aircraft accident,

$X_5 \equiv$ number of people displaced by airport development,

$X_6 \equiv$ number of people subject to a high noise level (i.e. 90 CNR or more).

To incorporate time effects of building the airport, the appropriate attributes were defined using present values or averages where appropriate. The capacity attribute X_2 had to be made a function of capacity for 1975, capacity for 1985, and capacity for 1995, and thus it was a vector attribute.

5.2 Summary of the Method Used in the Problem

After verifying assumptions concerning preferential and utility independence and ascertaining the appropriateness of the multiplicative model, assessments were begun. First, the fractile method was used to obtain probability distributions for all of the alternatives under consideration. Probabilistic independence was assumed to simplify calculations. Then uniattribute utility functions were assessed for all eight scalar attributes. The k_i's were assessed using the lottery over all the attributes illustrated by Question I in Section 3.1 for both the overall MUF and nested capacity MUF. Consistency checks on the relative k_i's involving trade-offs of two attributes at a time (see Question II, Section 3.1) were also employed. Special computer programs and graphic

displays were developed for evaluating alternatives and sensitivity analysis. For sensitivity analysis, the program allowed changes in i) the endpoints for the fractile cumulative probability distributions and ii) the scaling factors k_i. The shapes of the utility functions or the cumulative probability distributions could not be changed without programming adjustments.

5.3 A MUFCAP Approach to the Mexico City Problem

The MUFCAP approach would follow the existing methods scheme in making and verifying the preferential independence and utility independence assumptions. The INPUT command would structure the multiplicative function giving names such as "cost" and "access" to the various attributes along with ranges for the attribute amounts. Capacity would be put in as a nested MUF.

Alternatives would be specified by inputing the nine-point assessed fractile distribution for each uniattribute of an alternative. Utility functions for single attributes would be specified using any of the three forms available in MUFCAP.

Assessment of the k_i's could be accomplished without supplying the indifference probability for a lottery over all the attributes as was done. Pairs of indifference points for two attributes would be fed into MUFCAP to immediately produce indifference curves for examination and verification by the decision maker. In this way, the relative k_i's would be established with the aid of feedback. The magnitude of the k_i's would be established using INDIF1 (see Section 4.3), so a lottery over all the attributes could be avoided for this purpose. A good consistency check would be provided by comparing the magnitude of the k_i's implied by each method. Using MUFCAP, all of the initial assessments could be made and stored for later use. The assessments would have been made with

the aid of immediate feedback and with no need for very difficult lottery questions in which all the attributes were varied.

After the initial assessments, alternative evaluations and sensitivity analysis could be performed immediately with no need for special programming. Fractile distributions and utility function shapes could also be altered without programming adjustments. The different assessments of various individuals and groups could have been filed away for later reference using MUFCAP's filing capability.

In addition, other possibilities could have been explored with a minimum of extra effort. New attributes such as air pollution and political effects could have been added into the analysis with no special programming. The gradient calculation capability may have been used to suggest other alternatives for exploration and development. If the preferential independence of some attributes were questioned, different nesting schemes could have been tried to see if the ranking of the alternatives would be affected. Thus MUFCAP could have provided the assessment that was performed with no special programming and could have been used to explore variations of more parameters, other multiattribute nesting schemes, and additions of new attributes.

6. Summary and Suggestions

The current version of MUFCAP provides the basic features necessary to assess and use multiattribute utility functions in complex decision problems. In particular, it permits one to use realistic and simple questions in assessing the decision maker's preferences, rather than the "difficult to think about" types of questions previously used for computational reasons. MUFCAP provides for i) a variety of immediate feedback of implications of the decision maker's responses, ii) evaluation of alternatives and sensitivity analysis, and iii) analyzing differences

of preferences and judgments among various individuals in a decision making group.

The present MUFCAP should be considered a first edition, a basis on which to improve. In this regard, many possible improvements of existing routines have been suggested in the text such as a more sophisticated single-attribute utility function assessment technique and potential for evaluating alternatives where probabilistic independence need not be assumed. The program could then be easily coupled with simulation models producing probability distributions. Other important improvements would include the addition of new routines i) to help in verifying preferential and utility independence assumptions, ii) to facilitate sensitivity analysis and feedback, perhaps with the aid of graphical displays, and iii) to conduct conflict analyses in problems involving more than one decision maker.

APPENDIX

List of MUFCAP Commands with Brief Descriptions

Notation:

CE - Certainty equivalent

MUF - Multiattribute Utility Function

UNIF - Uniattribute (scalar attribute) utility function

$[y_1, y_2, \ldots, y_R]$ Brackets indicate the options which may be chosen. No option needs to be selected.

$(y_1, y_2, \ldots, y_R)$ Parentheses indicate that a choice must be made among the options given.

INPUT name - Inputs the structure of the multiattribute utility function to be referred to by "name." The dialogue requests names for the attributes and their ranges. Ranges for attributes over which preferences are monotonic should be input with the least desirable end of the range first. A vector attribute (and hence a nested MUF) is signalled by specifying a range whose lower and upper limits are the same. After INPUT, the default for all MUF's is the additive form with $k_i = k_j$ for all i, j. The default for all UNIF's is the linear utility function. The user is set to calculate immediately after INPUT.

SAVE filename - Saves the current preference and alternative specifications in file named "filename."

READ filename - Restores the information which was saved in "filename."

DEBUG - Lists all the attributes in the utility function structure including their names, scaling factors, ranges, and UNIF types (0, 1, and 2 indicate respectively linear, constant risk aversion, and piecewise linear). A vector attribute has its name and scaling factor listed and is followed by its component attributes.

ADDALT altname [factor] - Initiates dialogue to specify an alternative to be referred to by "altname." Either a probabilistic or certainty alternative may be specified. If the former is the case, a piecewise linear cumulative probability distribution is requested for each scalar attribute. (abscissa values for the cumulative are input in ascending order.) The option "factor" is a number which sets all of the scalar attributes at the factor level of their ranges, e.g. if factor is set equal to .1, all the scalar attributes are set at one-tenth of the way from the first range value to the second range value.

DROPALT altname - Removes the alternative "altname" from the status.

EVAL uname [A,B,...] - Evaluates the alternatives A,B,..., using the utility function associated with "uname." If no alternatives are specified, all alternatives in the status are evaluated and the results listed.

UNISET uname (LIN,CR,PL) - Sets the scalar attribute utility function associated with "uname" to linear, constant risk averse, or piecewise linear form. For the piecewise linear form, the abscissa values are input in ascending order.

KSET mname [factor,ADD,OVERIDE] - Sets the scaling factors for the MUF associated with "mname." The number "factor" causes the current scaling factors to be multiplied by that number. The program automatically calculates the k associated with the new scaling factors. If ADD is specified, the current factors are normalized to add to 1. The user may input k directly in response to the final prompt by the computer if OVERIDE has been specified.

GRAD uname [A,B,...] - Calculates the gradient components of the utility function associated with "uname" for all or some of the alternatives A,B,....

INDIF1 uname1 uname2 - In the uname1-uname2 attribute plane, given relative k_i's, (i.e., scaling factors with the appropriate ratio relationship to each other but not necessarily the appropriate absolute value) the k is specified by a single pair of indifference consequences. INDIF1 requests a pair of indifference consequences and uses the current k_i's as the given relative k_i's. On output, the k is given along with the factor by which the current k_i's must be multiplied to yield the k (see KSET command with "factor" option).

INDIF2 uname1 uname2 - In the uname1-uname2 attribute plane, with scaling factors denoted by k_1 and k_2, inputting two pairs of two indifference consequences each specifies the ratio k_1/k_2 and $k = \text{constant}/k_1$. After INDIF2, the KSET command may be used to fix k_1, and then k_2 and k in terms of k_1. The command IMAP can then be used to generate indifference curves in the uname1 uname2 plane. (For these indifference curves, the values of k_i, $i \neq 1,2$, are irrelevant.)

UNICAL uname [n] - Prints a list of utilities using the UNIF associated with "uname." Once the number n is specified, the user supplies n attribute amounts and the program returns the n associated utilities.

INVERSE uname [n] - Prints a list of attribute amounts associated with utilities using the UNIF "uname." Once the number n is specified, the user supplies n utility amounts of "uname" and the program returns the n associated attribute levels. If n is not specified, the program has a default printout.

CHANGEALT uname altname - Routine to change the "uname" attribute component of the alternative "altname" without changing the other components.

CHANGE uname (NAME,K,RANGE) param - Routine to change the name or scaling factor or range of the attribute "uname" to param. When the range is changed, param is not required. The program requests respecification of the UNIF type when the range is changed. When the name is changed, param must not be left blank.

ALTLIST - Lists the current alternatives. The probabilistic alternatives are listed with their CE equivalent components.

DISPLAY uname - Displays the characteristics of the utility function associated with "uname." The scaling factors for the attribute arguments and their sum is listed for a MUF while the range and type is listed for a UNIF.

FRACTILE uname altname - Displays the cumulative distribution for "uname" in the alternative "altname."

LOTTERY uname n - Calculates the CE for a lottery involving the scalar attribute "uname." The number n specifies the number of possible lottery consequences. These are solicited with their corresponding probabilities and the CE is calculated.

IMAP uname1 uname2 - Initiates a dialogue to generate an indifference "curve" in the uname1-uname2 plane. A point through which the curve will pass is solicited. Then values of uname1 are input and the uname2 values required to maintain indifference are output.

STOP - Thanks the user for using MUFCAP and exits from the program.

ADDU uname1 uname2 - Initiates a dialogue which adds an attribute "uname1" to the argument list of the MUF associated with "uname2."

DELU uname - Deletes the attribute uname from the structure.

SWITCH uname uname2 - Adds current attribute "uname" to the argument list of the MUF associated with "uname2" and deletes uname as an argument of the MUF to which it originally belonged.

References

[1] Fishburn, P.C. "Independence in Utility Theory with Whole Product Sets." Operations Research, 13 (1965), 28-45.

[2] Fishburn, P.C. Utility Theory for Decision Making. New York, Wiley, 1970.

[3] Keeney, R.L. "A Decision Analysis with Multiple Objectives: The Mexico City Airport." Bell Journal of Economics and Management Science, 4 (1973), 101-117.

[4] Keeney, R.L. "Multiplicative Utility Functions." Operations Research, 22 (1974), 22-34.

[5] Keeney, R.L. "Utility Functions for Multiattributed Consequences." Management Science, 18 (1972), 276-87.

[6] Meyer, R.F. "On the Relationship Among the Utility of Assets, the Utility of Consumption, and Investment Strategy in an Uncertain, but Time Invariant World." Proceedings of the Fourth IFORS Conference, Venice, Italy, 1969.

[7] Meyer, R.F. and Pratt, J.W. "The Consistent Assessment and Fairing of Preference Functions." IEEE Transactions on Systems Science and Cybernetics, SSC-4 (1968), 270-278.

[8] Pollak, R.A. "Additive von Neumann-Morgenstern Utility Functions." Econometrica, 35 (1967), 485-595.

[9] Pratt, J.W. "Risk Aversion in the Small and in the Large," Econometrica, 32 (1964), 122-136.

[10] Raiffa, H. Decision Analysis. Reading, Massachusetts, Addison-Wesley, 1968.

[11] Raiffa, H. "Preferences for Multi-Attributed Alternatives." RM-5868-DOT/RC, RAND Corporation, April 1969.

[12] Schlaifer, R.O. Analysis of Decisions Under Uncertainty. New York, McGraw-Hill, 1969.

[13] Schlaifer, R.O. Computer Programs for Elementary Decision Analysis. Division of Research, Boston, Massachusetts, Harvard Business School, 1971.

[14] Sicherman, A. "An Interactive Computer Program for Quantifying and Analyzing Preferences concerning Multiple Objectives." M.S. Thesis, MIT, 1975.

[15] von Neumann, J. and Morgenstern, O. Theory of Games and Economic Behavior. Second edition. Princeton, New Jersey, Princeton University Press, 1947.