Interactive Methods for Multiple Objective Linear Programming in Decision Support

A dissertation submitted in fulfillment of the
requirements of the Masters Degree in Operations Research
at the University of Cape Town

By
Makaya L. Makaya

Supervisor: Prof. Theodor J. Stewart

March 2005

Acknowledgements

I would like to give special thanks to the following people:

- My supervisor, Professor Theo Stewart for introducing MCDM to me and the guidance he provided throughout this research project. He did not only provide a solution to my problems, but always maintained a positive attitude
- My parents, my sister and the whole family for their continual love and support
- My brother Jimmy, for knowing you always think of me wherever you are
- Statistical Sciences M.Sc. students for testing my model
- Kudy for being my inspiration throughout all these years I spent in Cape Town
- All my friends at UCT

Synopsis

Making real–life decisions often involve multiple criteria/objectives. Multiple criteria in decision making are often conflicting in nature. Due to conflicting nature of criteria, the search for the most preferred solution must require input and feedback from the decision maker (DM) in the form of preference information. In practice, the DM's preference structure may be complex, requiring sophisticated methods and tools capable of both capturing these complex structures and aiding the solution procedure in identifying the most preferred solution.

Despite the fact that multicriteria decision making tools can be broadly used in real–life problems, the use of the methods for multicriteria decision making (MCDM) in practice is still scarce. The basic reason for that situation might be that practitioners are not familiar with MCDM methods, which often are difficult and lengthy to understand and to work at. Nowadays computers can assist decision makers reach the satisfactory solutions faster, but still a comprehensive understanding of MCDM methods to use is a problem.

In this dissertation we design an interactive multicriteria decision support system (ICMDSS) for assisting decision makers solve multiple conflicting objective problems. The user–friendly interactive DSS is for solving multiple objective linear programming (MOLP) problems. The DSS is for a single decision maker.

In chapter 1 we discuss the general overview of multicriteria decision making. We also introduce the multiple objective linear programming. We give the basic definitions of the terms often used in MCDM. The interactive methods for MCDM are also briefly discussed. We give general introduction to two MCDM techniques mainly discussed in the dissertation being

- The generalized goal programming methods
- The value function methods.

In chapter 2 we go into deeper details of the interactive methods for MCDM. We focus on the generalized goal programming and value function methods. We focus mainly on some of the traditional generalized goal programming and value function methods, and the methods that we have implemented in the DSS. Some

comments on the methods discussed are given. We also review the literature on interactive methods, mainly multiple objective linear programming applications.

In chapter 3 we discuss the issues related to the development of the decision support systems. We describe the different definitions, types and characteristics of decision support systems. The essential requirements in a decision support system are also outlined. We also discuss the system development life cycle (SDLC).

In chapter 4 we discuss the development of the DSS. The algorithmic steps for three interactive MCDM methods implemented in the DSS are outlined. The methods are

- The reference point method [A.P80, P82b]
- The interactive weighted Tchebycheff method [RE83]
- The value function method [VJ02]

Some modifications made to the methods are also discussed. Different user interfaces in the DSS are shown. We illustrate the decision maker's interaction part in the decision making process through the DSS.

In chapter 5 we discuss the validation and evaluation of the DSS. The results for laboratory tests using three implemented MCDM methods are given. Issues related to the field test of the DSS are discussed. Descriptive and summary statistics of the DSS and the methods implemented in the DSS are given. In chapter 6 we give the conclusions and suggestions for further research.

In appendix A the pictorial examples of problem formulation in the DSS are given. In appendix B we describe the multicriteria problems used for validating the DSS. Two different problems and their mathematical presentations are described in this appendix. The problems are for laboratory and field tests. In appendix C the pictorial examples of the solution process using three different methods implemented in the DSS are given. In appendix D the questionnaire used for evaluating the DSS is given. In appendix E the guidance for accessing the Visual Basic for Applications (VBA) code is given.

Finally the list of references is given.

Contents

List of Figures

List of Tables

List of Abbreviations

4GL – Fourth Generation Language

ABSALG – Aspiration–Based Search Algorithm

AIM – Aspiration–Level Interactive Model

ANOVA – Analysis of Variance

CP – Compromise Programming

DBMS – Data Base Management System

DM – Decision Maker

DMP – Decision Making Process

DNN – Decision Neural Network

DSS – Decision Support Systems

ES – Expert Systems

FFANN – Feed–Forward Artificial Neural Networks

GDF – Geoffrion, Dyer and Feinberg

GP – Goal Programming

GP_ϵ – ϵ–constraint Goal Programming

GPL – General Purpose Programming Language

GUI – Graphical User Interface

IDSS – Intelligent Decision Support Systems

IGP – Interactive Goal Programming

IIASA – Institute for Applied Systems Analysis

IMCDSS – Integrated Multicriteria Decision Support System

IMGP – Interactive Multiple Goal Programming

IWTP – Interactive Weighted Tchebycheff Procedure

JMS – Jacquet–Lagrèze, Menziani and Slowinski

LGP – Lexicographic Goal Programming

LP – Linear Programming

MAUF – Multiattribute Utility Function

MAUT – Multiattribute Utility Theory

MAVT – Multiattribute Value Theory

MCDA – Multicriteria Decision Aid

MCDM – Multicriteria Decision Making

MIS – Management Information Systems

MODM – Multiobjective Decision Making

MOLP – Multiple Objective Linear Programming

MP – Mathematical Programming

MS – Management Science

OR – Operational Research

PROMISE – PRogrammation Multiobjectif Interactive StochastiquE

RINGS – Utility Range based INteractive Group support System

ROMC – Representations, Operations, Memory aids and Controls

SDLC – Systems Development Life Cycle

SIMOLP – Simplified Multiobjective Linear Programming Procedure

SIS – Strategic Information Systems

UI – User Interface

V&V – Verification and Validation

VBA – Visual Basic For Applications

WGP – Weighted Goal Programming

ZW – Zionts and Wallenius

Chapter 1

Introduction to Multicriteria Decision Making

Processes of evaluation and decision making with multiple criteria/objectives are common in people's daily living and work experiences. Some of these decisions are directly related to the main worries of humankind, such as survival, security or perpetuation. Regardless of whether these processes involve everyday choices of meals or complex, consequential national energy policy decisions, they encompass multiple criteria/objectives and preferences of the decision makers. The environment, in which many of these decisions have to be made, is often unstructured and the consideration of multiple conflicting criteria/objectives is the rule rather than the exception. As human beings we do not always have the ability to take cognisance of all of these multiple influences when attempting to make a rational and meaningful decision.

The mathematical modelling of these decision making problems started in the 20th century with applied mathematicians and economists such as Pareto, Von-Neumann, Morgenstern and may more. These researchers were the first to realize the 'dilemma of conflicting criteria/objectives'. It is, however, only during the last three decades that there has been an increased awareness of the need to identify and consider simultaneously several criteria/objectives in the analysis and solution procedure of decision problems. Multiple criteria decision making (MCDM), as it is known today, has evolved in response to these practical needs.

1.1 General Overview of Multiple Criteria Decision Making Ideas

Multicriteria decision–aid (MCDA) or 'multiple criteria decision making' (MCDM) refers to a structured (organized) approach to decision making, or solving of decision and planning problems in the presence of multiple, usually conflicting objectives [Sta79]. *Solving* means that the decision-maker (DM) will choose one *reasonable* alternative from among a set of available ones. Bogetoft and Pruzan [PP91] also define MCDM as both an approach and a body of techniques designed to help people make choices which are in accord with their values in cases characterized by multiple, noncommensurate and conflicting criteria. The concept of an optimum does not exist in a multicriteria framework and thus multicriteria analysis cannot be justified within the optimization paradigm frequently adopted in traditional Operational Research/Management Science [VJ02]. This is why the word 'aid' seems essential to MCDA. The following points are further outlined for MCDM:

- MCDM seeks to take explicit account of multiple, conflicting criteria in aiding decision making;
- The MCDM process helps to structure the problem;
- The principal aim is to help decision makers learn about the problem situation, about their own and others values and judgements, and through organization, synthesis and appropriate presentation of information to guide them in identifying, often through extensive discussion, a preferred course of action;
- The analysis serves to complement and to challenge intuition, acting as a sounding–board against which ideas can be tested – it does not seek to replace intuitive judgement or experience;
- The process leads to better considered, justifiable and explainable decisions – the analysis provides an audit trail for a decision;
- The most useful approaches are conceptually simple and transparent;
- The previous point notwithstanding, non–trivial skills are necessary in order to make effective use even of such simple tools in a potentially complex environment. (Belton and Stewart [VJ02], p5)

The MCDM approach involves describing a decision problem with the following basic elements:

Value: Something a person cares deeply about, for example, physical, biological, and/or socio-cultural environments.

Goal: Something that is either achieved or not. For example, the Decision maker (DM) may seek to reduce travelling costs by at least 15%. If the goal cannot be or is unlikely to be achieved, it can be changed to an objective.

Objective: Something to be pursued to its fullest. For example, a business may want to minimize its customers' complaints. An objective basically indicates the direction desired.

Attribute: It is a measure that evaluates the achievement of goals and objectives.

Decision Maker: A single person, a group of persons, or an organization tasked with making decisions. The decision maker is supposed to have a better insight into the problem and is supposed to express preference relations between different solutions. Everyone makes many decisions daily. Thus Makowski and Wierzbicki [MP03], limit the concept of a *decision maker* to those persons who are aware of the importance of decisions made by them or at least reflect how these decisions are made.

Decision Alternatives: Feasible options for a decision; feasible solutions to a decision problem.

Criteria: The basis for evaluating decision alternatives. Criteria may be defined in terms of goals and objectives. Criteria measure the effectiveness of performance.

Constraint: A limit on attributes and decision variables that may or may not be stated mathematically. For example, we may consider that a person has to work at most ten hours per day as a constraint.

Outcomes: achievement or performance of each decision alternative on criteria.

Our values, beliefs and perceptions are the force behind almost any decision making activity. They are responsible for the perceived discrepancy between the present and desirable state. For example, valuing highly the cultural opportunities of living in a city propels a suburban dweller to make a decision of moving to an apartment in downtown. Values are articulated in an objective, which is often the first step in formal (supported by decision making techniques) decision process. Following the above example, an objective may be formulated as "find a good apartment downtown". This objective may be put forth by an individual (decision maker) or a group of people (a family). The actual decision boils down to selecting "good apartment" from a number of available apartments. Each available apartment represents a decision alternative. In the MCDM context, the selection is facilitated by evaluating each apartment on the set of criteria. The criteria in this case may include: rent/purchase price, parking availability, quality of neighborhood, distance from the shops, quality of

apartment, level of traffic noise and many more. Identifying the criteria should have the following characteristics:

> *Value relevance*: The DM's should be able to link the concept of their goals, thereby enabling them to specify preferences which relate directly to the concept. For instance, in the above example, distance from the shops may cause a confusion as to whether an apartment ought to be far from the shops or nearer to them.
>
> *Understandability and Operational Meaningfulness*: DM's should have a shared understanding of concepts to be used in an analysis.
>
> *Measurability*: All multiple criteria decision analysis (MCDA) implies some degree of measurement of the performance of alternatives against specified criteria, thus it must be possible to specify this in a consistent manner.
>
> *Decomposition*: The criteria can be broken down into parts to simplify the process.
>
> *Non-redundancy*: There should be no more than one criterion measuring the same factor (i.e. double counting should be avoided).
>
> *Judgemental independence*: Criteria are judgementally independent if preferences with respect to a single criterion, or trade-offs between two criteria, do not depend on the level of another.
>
> *Completeness*: The criteria should cover all aspects of the problem.
>
> *Operationality*: The criteria should be meaningfully used in the analysis.
>
> *Simplicity versus complexity*: The modeler should strive for the simplest value tree or criteria which adequately captures the problem for the DM. (Belton and Stewart [VJ02], p55-58)

In addition to above MCDM elements we take a look at three generic types of MCDM problems being [P92]:

- **Selection**. Given a set $\mathbf{S}$ of alternatives (also called options), the *selection* task operation involves finding a subset $\mathbf{S}'$ of $\mathbf{S}$ composed of as small as possible number of alternatives, judged by DM's as the most satisfying.
- **Sorting**. The sorting operation (also known as classification) consists of assigning each alternative from $\mathbf{S}$ to one of the pre-

defined categories. The assignment should be based on the intrinsic measure of a criterion for an alternative and not on its comparison with other alternatives from **S**. However, in practice, assignment is often based on relative differences of alternatives along a criterion.

- **Ranking**. The ranking operation involves establishing a preference weak–ordering on the set of alternatives **S**.

1.1.1 Types of MCDM Problems

MCDM problems and the methodologies for solving them can mainly be categorized as being either *continuous* or *discrete*. Goicoechea et al.[ARL82] further adds that the nature of MCDM problems is reflected by whether the decision variables of the problem are continuous or discrete. Continuous MCDM problems (because of their continuous decision variables) usually result in a choice situation involving an infinite number of possible alternatives. *Goal programming* (GP) and *utility function assessment* are two methods applicable for solving both continuous and discrete MCDM problems. Yet there are also other methods strictly dealing with MCDM problems with continuous decision variables.

Unlike in continuous MCDM problems, there are many decision situations in which the DM must choose among a finite number of alternatives which are evaluated on a common set of noncommensurable multiple criteria. Problems of this sort occur in many practical situations, for example, which one of five candidates should be hired. These are the so-called *discrete* MCDM problems. The structure of the discrete problem is usually presented in a payoff table. Note that we shall go into details of methods used to solve mainly continuous types of MCDM problems.

1.2 Interactive Multiple Objective Linear Programming As A Subset of MCDM

Multiple objective linear programming (MOLP) is an important subset of MCDM. MOLP is also a subset of a larger class of models called **mathematical programming models**. Mathematical programming[1] (MP) under multiple objectives has emerged as a powerful tool to assist in the process of searching for decisions which best satisfy a multitude of conflicting objectives [TJT99]. MOLP falls under the continuous types of MCDM where the set of alternatives **S** is defined implicitly.

[1]Mathematical programming model is a mathematical abstraction and should not be confused with programming computers (although building and using the model inevitably requires programming of a computer)

The field of multiple objective linear programming(MOLP) has attracted a lot of attention since the early 1970's and many approaches were developed to address the MOLP problems. Most MOLP methods utilize one of the three approaches: (1) a vector maximization approach, (2) an aspiration approach based on a DM's aspiration levels or goals and (3) a utility maximization approach [VYS97]. Our main focus in this research will be on the last two approaches of MOLP. In the essence of the precise statement of the decision maker's preference for outcomes in objective space, MOLP problems do not generally have a unique solution like a single objective linear programming problem (Arbel and Korhonen [AP01]). Instead a family of reasonable solutions is identified and the intervention of the decision maker (DM) is required to find the "most preferred" solution. The resulting set of procedures developed for these types of problems is referred to as *interactive methods* for MOLP problems. A number of computer implementations were developed (and we shall develop in this thesis) over the years differing from each other in a way they assess preferences from the DM and the way they derive search directions to move from current iterate to the next in MOLP problems.

A traditional MOLP problem is usually described mathematically as follows:

$$\begin{aligned} &\text{Max} && \mathbf{z} = \mathbf{C}\mathbf{x} \\ &\text{Subject to :} && \mathbf{x} \in \mathbf{S} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{b} \in \mathbb{R}^m\} \end{aligned} \tag{1.1}$$

where $\mathbf{A}$ is an $m \times n$ constraint matrix of full rank m, b_l $(l = 1, 2, ..., m)$ are the values of the right side of the constraints and $\mathbf{C}$ a $k \times n$ objective matrix whose rows C_i $(i = 1, 2, \ldots, k)$ are formed by k individual objectives, and $z_i = C_i\mathbf{x}$, where $\mathbf{z} \in \mathbb{R}^k$ is the objective vector. The set $\mathbf{S}$ of feasible solution vectors is a subset in the so-called decision (variable) space $\mathbb{R}^n$. The set $\mathbf{Z}$= $\{\mathbf{z} \in \mathbb{R}^k : \mathbf{z} = \mathbf{C}\mathbf{x},\ \mathbf{x} \in \mathbf{S}\}$ of all objective vectors corresponding to feasible solution vectors $\mathbf{x} \in \mathbf{S}$, defines the set of attainable outcomes in the so–called *objective space* $(\mathbb{R}^k)$.

In other words, the model has n decision variables, m constraints and k explicit objectives. The primary assumption is that all the relations in the above mathematical model are linear (i.e the objective functions and constraints are linear). Without loss of generality, we shall assume that all the objectives/criteria are to be maximized.

As stated before, the DM wishes to maximize all single objective functions; however, due to the conflicting nature of objectives, there may not exist one alternative that maximizes all objectives. In fact, there may exist many alternatives from which, the most preferred alternative must be selected. Often, there is an infinite number of solutions and they are not comparable [BT01]. To facilitate the selection of the most preferred alternative, one may assume there exists a function that represents the preference of the DM with respect to all objectives (or so-called criteria).

1.3 Basic MCDM Definitions

This section gives brief definitions of MCDM which are usually used.

1.3.1 Dominance

Definition

Let $\mathbf{z}$, $\mathbf{z}' \in \mathbb{R}^k$ be two criterion vectors. Then $\mathbf{z}$ *dominates* $\mathbf{z}'$ if and only if

$$\mathbf{z} \geq \mathbf{z}' \text{ and } \mathbf{z} \neq \mathbf{z}'$$

i.e.

$$z_i \geq z_i' \text{ for all } i \text{ and } z_i > z_i' \text{ for at least one } i$$

In other words, if $\mathbf{z}$ dominates $\mathbf{z}'$, no component of $\mathbf{z}$ is less than the corresponding component of $\mathbf{z}'$, and at least one component of $\mathbf{z}$ is greater than its corresponding component of $\mathbf{z}'$.

Definition

Let $\mathbf{z} \in \mathbf{Z} \subset \mathbb{R}^k$. Then $\mathbf{z}$ is *nondominated* if and only if there does not exist $\mathbf{z}' \in \mathbf{Z} \subset \mathbb{R}^k$ such that

$$\mathbf{z}' \geq \mathbf{z} \text{ and } \mathbf{z} \neq \mathbf{z}'.$$

Otherwise $\mathbf{z}'$ is a *dominated* criterion vector.

In other words, a criterion vector is *nondominated* if it is not dominated by any other feasible criterion vector. The set of all nondominated criterion vectors is called a *nondominated set.*

1.3.2 Efficiency

Definition

Let z_i be the function that defines criterion i. A point $\mathbf{x} \in \mathbf{S}$ is *efficient* if and only if there does not exist another point $\mathbf{x}' \in \mathbf{S}$ such that

$$z_i(\mathbf{x}') \geq z_i(\mathbf{x}) \text{ and } z_i(\mathbf{x}') \neq z_i(\mathbf{x}), \text{ for } i = 1, 2, ..., k$$

An alternative/action is said to be *efficient* if and only if no alternative of **S** (a set of all alternatives) dominates it. The set of efficient alternatives (which can be **S** when the dominance relation is empty) is generally considered as the set of the only interesting actions, even if there are sometimes good reasons for not definitely rejecting non–efficient actions [P92]. The definition of an efficient alternative gave rise to some variants, leading to concepts of weak efficiency, strong efficiency, proper efficiency and so on.

The set of all efficient points is called the *efficient set.*

Definition

The point $\mathbf{x} \in \mathbf{S}$ is said to be *weakly efficient* if and only if there does not exist another $\mathbf{x}' \in \mathbf{S}$ such that $z_i(\mathbf{x}') \geq z_i(\mathbf{x})\ \forall i$.

1.3.3 Ideal Point (Anchor Value)

Let $\mathbf{x}^j$ $(j = 1, 2, ..., k)$ be the solution to the problem

$$\begin{aligned} &\text{Max} && C_j\mathbf{x} \\ &\text{Subject to} && \mathbf{x} \in \mathbf{S} \end{aligned} \tag{1.2}$$

Let $z_{ij} = z_i(\mathbf{x}^j)$ $(i = 1, 2, ..., k;\ j = 1, 2, ..., k)$ define a payoff table. That is, z_{ij} is the value achieved for the i^{th} objective when maximizing $\mathbf{x}^j$.

Definition

The *ideal point* in $\mathbb{R}^k$ is the point whose coordinates are $(z_1^*, z_2^*, \ldots, z_k^*)$, where z_i^* is the maximum of $z_i(\mathbf{x})$ over the feasible set. By definition

$$z_i^* = z_{ii}$$

1.3.4 The Nadir

Definition

The *nadir* is defined as the minimum of $z_i(\mathbf{x})$ over the efficient set.

The nadir can be very difficult to calculate precisely. A commonly used approximation of the nadir is given by

$$\underline{z}_i = \min_j^k z_{ij},\ \ i = 1, 2, ..., k$$

However, Weistroffer [R85] shows (with counterexamples) that these approximate nadir values (from the payoff table) are not lower bounds for the objective function values on the efficient solution set. He furthermore advises that any solution technique should allow the user to investigate solutions with objective

function values lower than the nadir values from the payoff table. Steuer [E88] also reports computational experience which demonstrates that the discrepancies between the payoff table minimums and the minimums over the efficient set can often be large. Korhonen et al.[PSE97] further comment that

> unless special measures are taken when there are alternative optima, there is no guarantee that all row criterion vectors from the payoff table will be nondominated. If the minimum column value occurs in a row whose criterion vector is only weakly nondominated, the minimum column value may actually *underestimate* the nadir. Otherwise, the minimum column value will *correctly* specify the nadir or *overestimate* it... The problem of overestimating the nadir is much more difficult

Korhonen et al.[PSE97] pursue a heuristic for the purpose of obtaining improved estimates of the nadir values but without adding great complexity to the task. Being a heuristic, the finding of nadir values is not guaranteed, but computational results show that much better estimates can be obtained utilizing their approach than from using payoff tables alone. Ehrgott and Tenfelde–Podehl [MD03] also investigate the problem of finding the nadir point. They propose a general method to compute nadir values based on theoretical results on Pareto optimal solutions of subproblems with fewer criteria.

It is evident that there is a room for further research in finding a better method than payoff tables for estimating the minimum criterion (nadir) values over the efficient set. Since only the heuristics are present in literature, we shall in this dissertation continue to use the payoff tables for the estimation of the nadir in the generation of MCDM models for our decision support system (DSS). The user of the DSS should however bear in mind that there might be the minimum criterion (nadir) values which are below the nadir values estimated by the payoff tables. Any nadir point of this sort still provides for a feasible solution over the efficient set.

1.4 Methods of Interaction in General

In this section we give a brief introduction for the interactive methods for multicriteria decision making (MCDM) also known as the progressive articulation of preference in MCDM. According to Stewart [J99b], these are the methods in which the full preference structure of the DM is not structured and elicited *a priori*[2], but is evaluated progressively and locally in response to simple choices made by the DM. He adds that all MCDM and technical mathematical programming aspects of (for example) identifying efficient solutions, need to be proceeded by the interaction with the DM to identify and structure the criteria.

[2]If preferences are *a priori*, the DM has to define his/her preferences in advance (before actually performing the search).

A number of MCDM approaches have come to be termed *interactive methods* in that they involve the following characteristic steps [RE94]:

- A feasible (and usually efficient) solution, or a small number of solutions is generated according to some specified procedure and presented to the DM.
- If the DM is satisfied with the solution generated, then the process stops. Otherwise he/she is requested to provide some local preference information in the vicinity of the solution(s) presented, such as direct comparisons between (actual and hypothetical) solutions, tradeoffs or desired directions of improvement.
- In the light of the local information provided, preference models are updated and/or parts of the decision space are eliminated and the process returns to the first step.

According to Goicoechea et al.[ARL82], procedures of interactive multiple objective programming more or less follow the same general algorithmic outline as depicted in the next figure:

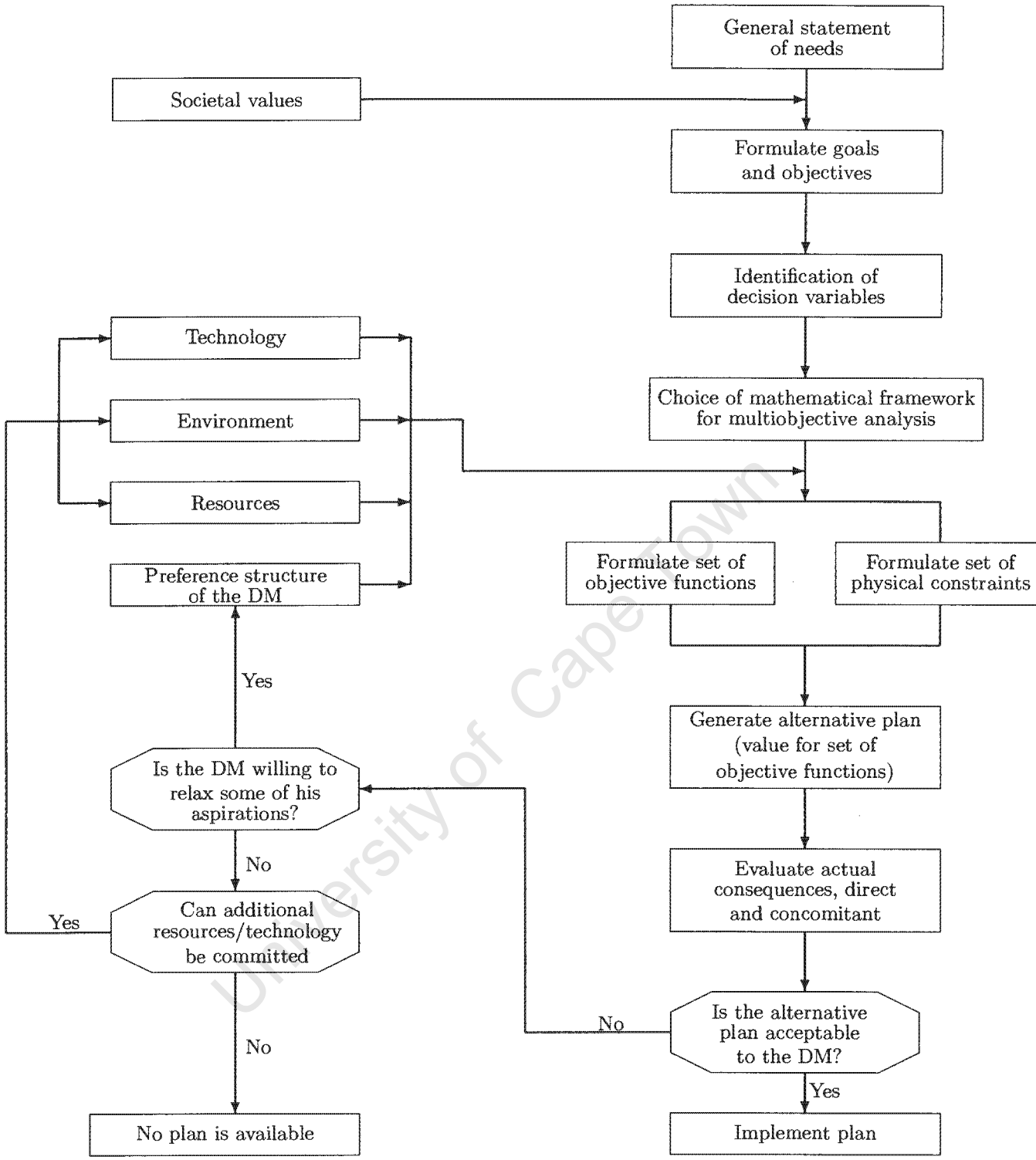


Figure 1.1: An Operational Framework for Interactive Methods (From Goicoechea et al[ARL82], p9)

These methods typically require greater DM involvement in the solution process. This has an advantage of allowing the DM to gain greater understanding and feel for the structure of the problem. Stewart [J99b] further clarifies that the use of interactive methods is well suited to situations in which a single individual, or a small homogeneous group, needs quite quickly to find one or a few satisfactory alternatives to the decision problem, primarily when the criteria are well-represented in terms of *quantitative attributes*. The methods are totally not well suited in group decision making contexts in which there are substantial conflicts, in which many criteria are qualitative, or in which a clearly defensible justification for the solution obtained need to be established. In interactive methods, DM's preferences are also either *quantitative* or *qualitative*. Quantitative information is given in a numerical form e.g. a question requiring a quantitative DM's preference may be "By what amount do you wish to raise the target value", whereas the qualitative question may be "Do you wish to improve the target value". Qualitative questions are generally easy to answer, especially when the DM is unsure of his/her preference structure [MF97].

Although *a priori* and *a posteriori*[3] decision making are common in Operational Research (OR) literature, interactive approaches (the progressive articulation of preferences) are often favoured by researchers for several reasons:

1. Perception is influenced by the total set of elements in a situation and the environment in which the situation is embedded.
2. Individual preference functions or value structures cannot be expressed analytically, although it is assumed that the DM subscribes to a set of beliefs.
3. Value structures change over time, and preferences of the DM can change over time as well.
4. Aspirations or desires change as a result of learning and experience.
5. The DM normally looks at trade–offs that satisfy a certain set of criteria, rather than at optimizing all the objectives at a time.

1.5 Goal Programming and its Related Techniques

1.5.1 The Evolution of Goal Programming

Romero [C91] claims goal programming (GP) to have been, and still the most widely used Multicriteria Decision Making (MCDM) technique. The primary reason for the popularity of GP appears to be its underlying philosophy of satisficing. GP is often cited as being the "workhorse" of multiple objective

[3] If the preferences are expressed *a posteriori*, we search first and decide later.

optimization [C04].

Goal Programming is a multiobjective linear programming (MOLP) technique. The ethos of GP lies in Simon's [A55] concept of 'satisficing' of objectives which involves the DM in the process that attempts to achieve a 'satisfactory' level of multiple objectives rather than an optimal solution for a single objective. The roots of GP lie in the paper of Charnes et al.[AWO55] in which they deal with the executive compensation methods. A more explicit definition is given by Charnes and Cooper [AW61] in which the term Goal Programming was introduced for the first time. Until the middle of the 1970's, GP applications reported were rather scarce. Interestingly, GP was not presented as a unique or revolutionary methodology but an extension of Linear Programming (LP). Charnes and Cooper [AW61] only suggested goal programming for use in solving unsolvable LP problems (for example, infeasible LP problems). GP principles can now carry through to non-linear and non-convex (including discrete choice) problems. From that impressive work of Charnes and Cooper and chiefly due to seminal works by Lee and Ignizio [M72, P6a, P6b], an impressive boom of GP applications and non–technical improvements have arisen. In his book, Schniederjans [J95] cite references of GP dating to as early as the 1950's.

More recently other variants of GP have emerged, including a variety of interactive methods (which we shall review in this research project) and reference point methods as formulated by Wierzbicki [A.P80, P99], which Korhonen and Laakso [PJ85] term 'generalized goal programming'. Their generalization tries to preserve the main advantages of GP and to overcome its basic disadvantages.

1.5.2 GP Terminology

For each *objective* in a goal programming problem, the DM has to specify a numerical *aspiration level* that serves to relate or transform the objective into a numerical *goal*. The DM, for example may seek to reduce travelling costs by at least 15% (which can be seen as a goal). Since, in most situations the aspiration levels cannot be fully satisfied, *deviations* from goals can be expected. The reader should bear in mind that in goal programming a specific numerical (quantitative) goal is first established for each objective.

1.5.3 Traditional GP model

The vast popularity of GP is due to the fact that it is easy to understand and the fact that it is easy to apply, since it constitutes an extension of (multiple objective) linear programming , for which very effective solving algorithms are available (Aouni and Kettani [BO01]). Since the origin of GP can be traced to LP, a starting point for GP model can be found by restating the LP model, its assumptions and modelling notation (Schniederjans [J95]).

The LP model can be expressed as follows:

$$\begin{aligned} \text{Min} \qquad & Z = \sum_{j=1}^{n} C_j x_j \\ \text{Subject to}: \quad & \sum_{j=1}^{n} a_{jl} x_j = b_l, \\ & x_j \geq 0 \end{aligned} \tag{1.3}$$

for $l = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, where x_j are non-negative decision variables, and the C_j are contribution coefficients that represent the marginal contribution to Z for each unit of their respective decision variable. This LP model seeks a single objective or goal of minimizing the objective function (Z). The above model has m constraints with a_{jl} ($l = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$) as technological coefficients that represent the per unit usage by x_j of the right hand side coefficient b_l. All the decision variables should be non–negative.

The method of GP was initially approached by Charnes and Cooper [AW61] in their first text book presentation of GP. The aim was to seek a solution that minimizes the (weighted) sum of deviations of the objectives goals (or targets). The process involves identifying objectives and setting a goal for each objective. When objectives are identified, the function F_i ($i = 1, 2, \ldots, k$) measuring attainment for each objective i is set equal to the goal q_i. But since the overachievements and underachievements of goals are possible, deviational variables d_i^+ (for overachievement) and d_i^- (for underachievement) are introduced.

The model can be set mathematically as follows:

Let $x_1, x_2, \ldots, x_n$ be the decision variables of the problem.
Let C_{ij} be the coefficient of x_j ($j = 1, 2, \ldots, n$) in the objective function for each objective i ($i = 1, 2, \ldots, k$), where k is the number of objectives.
Let q_i be the goal (target) for objective i.

The solution of the GP should be as close as possible to attaining the goals.That is, we seek as far as possible to solve:

$$\sum_{j=1}^{n} C_{ij} x_j = q_i \quad , \text{ for } i = 1, 2, \ldots, k$$

where

$$\sum_{j=1}^{n} C_{ij} x_j, \quad (i = 1, 2, \ldots, k)$$

is the function measuring the attainment of objective i, introduced as F_i before. But it is generally not possible to satisfy these goals simultaneously. One possibility is to express the objective function for the GP model as:

$$\text{Min} \quad Z = \sum_{i=1}^{k} |(\sum_{j=1}^{n} C_{ij}x_j - q_i)|, \tag{1.4}$$

That is, minimize the positive sum of deviations from goals. Hence the objective function of the GP model is expressed as a *preference function* or *achievement function*. Then the objective function given in (1.4) has to be transformed into a linear programming format where the new variables are introduced (to the objective function) as follows:

$$d_i = \sum_{j=1}^{n} C_{ij}x_j - q_i, \text{ for } i = 1, 2, ..., k \tag{1.5}$$

so that the objective function is reduced as follows:

$$\text{Min} \quad Z = \sum_{i=1}^{k} |d_i| \tag{1.6}$$

But since d_i can be either positive or negative, it (d_i) can be replaced by the difference of d_i^+ and d_i^- mentioned earlier. That is,

$$d_i = d_i^+ - d_i^-$$

where

$$d_i^+ \geq 0, \; d_i^- \geq 0$$

such that

$$|d_i| = |d_i^+ - d_i^-| = d_i^+ + d_i^-, \text{ for } i = 1, 2, ..., k$$

Hence the complete GP model can be written as:

$$\text{Min} \qquad Z = \sum_{i=1}^{k} (d_i^+ + d_i^-)$$

$$\text{Subject to :} \qquad \sum_{j=1}^{n} C_{ij}x_j - (d_i^+ + d_i^-) = q_i$$

$$\Rightarrow \sum_{j=1}^{n} C_{ij}x_j - d_i^+ + d_i^- = q_i, \text{ for i } = 1, 2, ..., \text{k}$$

$$\text{and any other original LP constraints involving } x_j \tag{1.7}$$

Note that since we cannot have both underachievement and overachievement of a single objective simultaneously, either one of the deviational variables should be set equal to zero. That is, $d_i^+ \times d_i^- = 0 \Rightarrow$ either $d_i^+ = 0$ or $d_i^- = 0$ for $i = 1, 2, ..., k$.

Markland [E89] points out that in practical situations deviations from certain goals may be much more important than deviations from other goals. This situation is addressed by assigning weights λ_{ip} and λ_{iq} to respective deviations d_i^+ and d_i^- for $i = 1, 2, ..., k$. As the result the weighted goal programming model can be written as:

$$\begin{aligned} \text{Min} \quad & Z = \sum_{i=1}^{k} (\lambda_{ip} d_i^+ + \lambda_{iq} d_i^-) \\ \text{Subject to} : \quad & \sum_{j=1}^{n} C_{ij} x_j - d_i^+ + d_i^- = q_i, \text{ for } i = 1, 2, ..., k \\ & \text{and any other original LP constraints involving } x_j \end{aligned} \tag{1.8}$$

For all objectives where less is better, d_i^+ is minimized, while d_i^- is allowed to take any positive value. Moreover, for objectives where more is better, d_i^- is minimized while d_i^+ is allowed to take on any value.

1.5.4 Extensions to Traditional Goal Programming

The standard GP formulation can produce inefficient solutions if the target values (goals) are set too pessimistically [MDC98]. This fact led some authors to argue against the use of GP [M81] and led to various GP variants and extensions.

As Tamiz et al.[MDC98] point out, goal programming can be classified into two major subsets. In the first type, the unwanted deviations (overachievements or underachievements) are assigned weights according to their relative importance to the DM and minimized. This is known as weighted goal programming (WGP). In another subset of GP, deviational variables are assigned into a number of priority levels and minimized in a lexicographic sense. Tamiz et al.[MDC98] define lexicographic minimization as a sequential minimization of each priority whilst maintaining the minimal values reached by all higher priority level minimization. This is known as lexicographic goal programming (LGP) as introduced and chiefly developed by Ijiri [Y65], Lee [M72] and Ignizio [P6a, P6b]. Lee and Oslon [ML99] call this type of GP, *preemptive goal programming.*

LGP and WGP are said to be the most widely used GP variants and Tamiz et al. [FE93] show that 64% of GP applications reported in literature use LGP, and 21% use WGP. Nevertheless, other GP variants are used in the remaining 15%. Other GP variants include MINMAX GP (Flavell 1976), where the maximum of the deviations is minimized. Lee and Oslon [ML99] point out that other non-linear goal programming variants are also being used, and their formulations are not more distinctly different from single objectives forms than the addition of deviational variables.

Lexicographic (Preemptive) Goal Programming

A brief introduction to LGP is given in this section since LGP is claimed to be the most commonly used variant of GP.

Preemptive GP involves a modelling process being [ML99]:

1. Identify the objectives
2. Set targets for objectives (multiple targets per objective are allowed)
3. Prioritize the objective-goal pairs. (If goals are incommensurable since they cannot be measured in the same units, tradeoff weights need to be identified for use within a priority level)
4. Solve a sequence of linear programming (LP) models by priority level.

The ordinal priority rankings are called preemptive priority factors [E89]. These priority factors denoted by P_i for $i = 1, 2, \ldots, s$ in s ordinal priority rankings have the relationship:
$P_1 >>> P_2 >>> \ldots >>> P_{s-1} >>> P_s$
where $>>>$ means "very much greater than".

This therefore indicates that the priority ranking is absolute; that is, the P_1 goal is so much important than P_2 goal that P_2 goal will never be achieved until the P_1 goal is achieved. Mathematically, the preemptive priority relation implies that multiplication by a real number $x > 0$, however large x may be, cannot make a lower-priority goal as important as a higher priority goal (i.e. $P_{s-1} > xP_s$). These preemptive priority factors are usually incorporated into the objective function as weights for deviational variables, since weights are usually not used in preemptive GP. Martel and Aoumi [JB90] further stress that weight aggregation can be difficult for human decision makers.

Mathematically, preemptive GP model is as follows:

$$\text{Min} \quad P_1 \sum_{i=1}^{k} \left(w_{ip} d_i^+ + w_{iq} d_i^-\right); \ \ldots; \ P_s \sum_{i=1}^{k} \left(w_{ip} d_i^+ + w_{iq} d_i^-\right)$$

$$\begin{aligned} \text{Subject to:} \quad & \sum_{j=1}^{n} C_{ij} x_j - d_i^- + d_i^+ = q_i \text{ for } i = 1, 2, \ldots, k \\ & x_j, \ d_i^-, \ d_i^+ \geq 0, \ \forall \ i, \ j \\ & P_1 >>> P_2 >>> \ldots >>> P_{s-1} >>> P_s \end{aligned} \tag{1.9}$$

where s is the number of preemptive levels.

1.6 Value (Utility) Methods

In value methods, also known as Multiattribute Value Theory (MAVT) or Multiattribute Utility Theory (MAUT) (even though MAVT[4] and MAUT[5] are slightly different), a global value (utility) function is established to represent the overall strengths of preference of the DM between the outcomes. The value (utility) functions are usually considered in addressing problems of choosing the most preferred alternative among a set of alternatives where each alternative is defined by the same criteria. In such problems the DM is supposed to express his/her preferences between presented alternatives.

It appears that many multiobjective decision making (MODM) techniques are based on the preference order of the DM, hence the value measurement approaches are used to construct a means of associating a real number with each alternative, in order to produce a preference order on the alternatives consistent with DM value measurement. Given any two alternatives $a, b \in \mathbf{S}$, a number (value) V(a) is associated with each alternative a, in such a way that a is judged to be preferred to b ($a \succ b$), taking all criteria into account if and only if V(a)>V(b), which also implies indifference between a and b ($a \sim b$) if and only if V(a)=V(b). Preference order implied by any such value function must of necessity constitute a *complete weak order (or preorder).*

That is [VJ02]:

> **Preferences have to be complete:** For any pair of alternatives, either one is strictly preferred to the other or there is an indifference between them (i.e. $a \succ b$, or $b \succ a$, or $a \sim b$).
>
> **Preferences and indifferences are transitive:** For any three alternatives a, b, and c, if $a \succ b$ and $b \succ c$ then $a \succ c$, similarly for indifferences, if $a \sim b$ and $b \sim c$ then $a \sim c$.

Belton and Stewart [VJ02] further state that:

> Within the value measurement approach, the first component of preference modelling (modelling the relative importance of achieving different performance levels for each identified criterion) is achieved by constructing "marginal" (or "partial") value functions say $v_i(a)$ for each fundamental criterion.... A fundamental property of the partial value function must be that the alternative a is preferred to b

[4]Value methods should be used in the context of deterministic decision situations where criterion values (outcomes of decision alternatives) are known or are highly predictable.

[5]Utility methods are aimed at probabilistic decision situations where there is an uncertainty about the outcomes of the decision outcomes.

in terms of criterion i if and only if $v_i(a) > v_i(b)$, similarly, indifference between a and b in terms of this criterion exists if and only if $v_i(a) = v_i(b)$.

In other words, the partial value function also satisfies the definition of preference function mentioned earlier. Partial value function can be described as the function of an attribute, say $v_i(z_i)$ without reference to any specific alternative [VJ02].

Hence the value function V can be represented as

$$V(\mathbf{z}) = \sum_{i=1}^{k} \lambda_i v_i(z_i),$$

where all criteria are associated with measurable attributes.

The weights (λ_i), which are the numeric values assigned to an evaluation criterion that indicate its importance relative to other criteria in the decision situation, can be determined directly by various weighting techniques such as the ratio method in Edwards [W77], the swing weights method in von Winterfeldt and Edwards [vWDE86], the tradeoff and pricing method in Keeney and Raiffa [LH76] etc. $\mathbf{z}$ represents the vector of attribute values, without reference to any specific alternative.

The above form of value functions is known as an *additive value function*. Stewart [J96] notes that the components of the additive value functions need to satisfy the following properties:

(1) The marginal values $v_i(z_i)$ must lie on an interval scale of preferences, i.e. equal increments on this scale have the same incremental value (measured perhaps by willingness to trade–off against some fixed currency) to the DM, irrespective of the baseline.

(2) The marginal values must also satisfy 'additive independence', i.e. the value gained by a fixed increment in $v_i(z_i)$ for some criterion, when performance levels of all other criteria are unchanged should not depend on the fixed levels of the other criteria.

(3) Irrespective of how the weights are actually assessed, they must be interpretable in tradeoff terms; that is if an increment $\triangle_i$ in $v_i(z_i)$ is just sufficient to compensate for a decrease $\triangle_p$ in $v_p(z_p)$, then

$$\frac{\lambda_i}{\lambda_p} = \frac{\triangle_p}{\triangle_i}$$

Weights are often assessed in practice by holistic subjective judgement rather than by the explicit specification of tradeoffs. This does require that the scale of measurement of the

value function be clearly and ambiguously identified, e.g. by careful definition of the end points of the scale.

However, the generation of the value (utility) function is still considered a very difficult task in multicriteria decision problems.

1.7 Concluding Remarks

This chapter is a brief introduction to MCDM and MCDM techniques which we shall deal with in this dissertation. Having the preliminaries of MCDM, we are in the position to take a deeper exploration of MCDM techniques in particular context and to review the related literature as a basis for further research.

Chapter 2

Interactive Methods for MCDM

In the previous chapter we gave general introduction to MCDM, especially value function and standard goal programming (GP) approaches. Concepts of interactive approaches were also introduced, but now expanded in this chapter, with particular emphasis on value function methods and generalization of GP. We also review the applications of the above methods reported in literature.

2.1 Generalized Goal Programming

According to Steuer and Gardiner [E90], the most prominent interactive procedures to have been developed are categorized as *value function procedures* and *generalized GP/reference point procedures.* Traditional generalized GP procedures can be further sub-divided as follows :

1. STEM [RJJO71]
2. Wierzbicki's Reference Point approach [A.P80, P82b]
3. Interactive Goal Programming [S72, SJ76]
4. Tchebycheff Method [RE83]
5. Visual Interactive Approach [JJ86]

Yet there are other recent reference point approaches such as Aspiration–Level Interactive Model (AIM) [VTS92], which solves decision problems involving the choice among discrete alternatives, and Aspiration–Based Search Algorithm (ABSALG) [VYS97], which can be applied in problems involving the choice among continuous alternatives (i.e. MOLP (mathematical programming) problems).

Although Wierzbicki [A.P80, P82b] names his method the *reference point approach*, there are several other methods which share the same features with his

method, hence they are generally called the *reference point methods*. Reference point methods are a *generalization of goal programming* to such cases when improvement to certain outcomes is wanted beyond their reference point [P99]. The generalization tries to preserve the main advantages of GP and to overcome some disadvantages. These approaches are a general framework rather than a specific method [P92]. The decision maker (DM) is iteratively asked to specify aspiration levels (reference point), which may or may not be feasible. Best approximation of these points is calculated by using an "achievement scalarizing function", which generates a non–dominated point closest to the desired levels if the aspiration levels are not attainable.

Definition: The most commonly used *achievement scalarizing function* (Wierzbicki [A.P80, P82b]) is the function of the form:

$$s(\mathbf{q}, \mathbf{z}, \lambda) = \max_i [\lambda_i(q_i - z_i)] + \varepsilon \sum_{i=1}^{k} \lambda_i(q_i - z_i) \tag{2.1}$$

where ε is an arbitrary small number, $\mathbf{q} \in \mathbb{R}^k$ is a "reference point" in the criterion space (which may or may not be feasible), and λ is a given weight vector from Λ. The idea of the model is to incorporate a small *regularization term*[1]. For MOLP problems with all–continuous variables, there always exist values for ε, small enough, such that all nondominated solutions are reachable (Alves and Clìmaco [JC00]); that is, the regularization term forces the resulting solutions to be nondominated.

Several forms of this function can be found in Benayoun et al.[RJJO71], Steuer and Choo [RE83], Nakayama and Sawaragi [HY84], Lewandowski et al.[ATTA88], Korhonen and Wallenius [PJ92], Costa and Clìmaco [PC94] etc.

An achievement scalarizing function $s(\mathbf{q}, \mathbf{z}, \lambda)$ projects the reference point $\mathbf{q}$ onto the set of nondominated criterion vectors $\mathbf{Z}$. With this function, the achievement scalarizing program is as follows:

$$\begin{aligned} \text{Min} \quad & \{\alpha + \varepsilon \sum_{i=1}^{k} \lambda_i(q_i - z_i)\} \\ \text{Subject to:} \quad & \alpha \geq \lambda_i(q_i - z_i), \ 1 \leq i \leq k \\ & \mathbf{z} \in \mathbf{Z} \end{aligned} \tag{2.2}$$

where $\alpha \in \mathbb{R}$ represents the maximum weighted deviation of an objective from a reference point i.e. $\alpha = \max_i \{\lambda_i(q_i - z_i)\}$, where λ is the weighting vector. The variable α can never be negative, although, the z_i variables are, in general, unrestricted.

[1]The regularization term is the second term in equation (2.1).

2.2 Value Function Methods

The MOLP problem mentioned in the previous chapter may be regarded as a value (utility) function program of the form:

$$\begin{aligned} &\text{Max} && V(\mathbf{z}) \\ &\text{Subject to :} && \mathbf{x} \in \mathbf{S} = \{\mathbf{x} \in \mathbb{R}^n : \ \mathbf{Ax} = \mathbf{b},\ \mathbf{b} \in \mathbb{R}^m\} \end{aligned} \tag{2.3}$$

where V is a real–valued function, which is strictly increasing in the criterion space defined at least in the feasible region $\mathbf{Z}= \{\mathbf{z} \in \mathbb{R}^k : \ \mathbf{z} = \mathbf{Cx},\ \mathbf{x} \in \mathbf{S}\}$. V maps the feasible region onto a one dimensional *value space* $\mathbb{R}$. The function V specifies the DM's preference structure over the feasible region. However, the key assumption in MOLP is that V is *not explicitly* known. Solutions for MOLP problems as in (2.3) are all those alternatives which can be the solutions of some value function $V : \ \mathbf{Z} \subset \mathbb{R}^k \rightarrow \mathbb{R}$.

Within the above mathematical framework, the primary objective is to find the "best compromise solution". Shin and Ravindran [SA91] define the "best compromise solution" as a nondominated solution that maximizes the DM's preference structure. They add that [SA91];

> research has been concerned with developing solution methods based on different assumptions and approaches to measure or derive the preference function ... Thus, the solution methods developed in multiple objective optimization problems can be categorized by the basic assumptions made with respect to the preference function when:
>
> (1) complete information of the preference function is available from the DM, usually by way of value (utility) functions (i.e preference structure elicited *a priori*).
>
> (2) no information is available. These methods do not explicitly take into account the DM's preference structure.
>
> (3) no partial information is obtainable progressively from the DM (i.e. no *a priori* preference information required).

On the other hand, in Hwang et al.[LRKM80] and Evans [W84] the third category in Shin and Ravandran is further split as follows:

> (1) Progressive articulation of the DM's preference by using an interactive procedure.
>
> (2) *A posteriori* articulation of the DM's preferences where he/she expresses his/her preferences for the efficient solutions generated by the algorithm.

It has to be clear that the value function is totally a decision maker–dependent concept. Different decision makers may have different value functions for the same problem.

2.3 Interactive Methods for Generalized Goal Programming Problems

In this section we discuss in more detail some interactive GP methods. The emphasis is on traditional interactive GP procedures and the methods that are implemented in the DSS.

2.3.1 STEM

STEM was proposed by Benayoun, Montgolfier, Tergny and Larichev [RJJO71]. It is a reduced feasible region method for solving MOLP problems. Conceptually, the method can be applied to both continuous and discrete problems, and both linear and nonlinear programming problems [ARL82]. STEM shares many features with the *compromise programming* approach. Stewart [J99b] defines the concept of *compromise programming* as simply the minimization of norms of the form:

$$\left[\sum_{i=1}^{m}\left[\lambda_i(q_i - z_i)\right]^p\right]^{1/p}$$

for some $p \geq 1$, which tends to the Tchebycheff norm:

$$\max_i \left[\lambda_i(q_i - z_i)\right]$$

as $p \to \infty$.

STEM algorithm is outlined as follows:

Step 1:

Let $z_{ij} = z_i(\mathbf{x}^j)$ $(i = 1, 2, ..., k;\ j = 1, 2, ..., k)$, where $\mathbf{x}^j$ is the solution to problem (1.2), define a payoff table. Construct a payoff table of the form:

	x^1	x^2	x^3	...	x^k
z_1	z_1^*	z_{12}	z_{13}	...	z_{1k}
z_2	z_{21}	z_2^*	z_{23}	...	z_{2k}
z_3	z_{31}	z_{32}	z_3^*	...	z_{3k}
$\vdots$	$\vdots$				
z_k	z_{k1}	z_{k2}	z_{k3}	...	z_k^*

Table 2.1: Payoff Table

where the rows are the criterion vectors formed by optimizing each of the

objectives. The z_i^* $(i = 1, 2, 3, \ldots, k)$ entries along the diagonal form the ideal criterion vector $(\mathbf{z}^* \in \mathbb{R}^k)$.

Evaluate the ideals z_i^* $(i = 1, 2, 3, \ldots, k)$ to form the ideal criterion vector $\mathbf{z}^* \in \mathbb{R}^k$. The ideals are used as the initial goals for each criterion. The ideal vector serves as a standard by which nondominated solutions can be evaluated.

Step 2:

Let iteration counter $r = 0$. Let the minimum value of each column of the payoff table be identified and labelled $\underline{z}_i$, $i = 1, 2, ..., k$. Calculate π_i values where

$$\pi_i = \begin{cases} \frac{z_i^* - \underline{z}_i}{z_i^*} \left[\sum_{j=1}^n (c_{ij})^2\right]^{-\frac{1}{2}}, & \text{when } z_i^* > 0 \\ \frac{\underline{z}_i - z_i^*}{z_i^*} \left[\sum_{j=1}^n (c_{ij})^2\right]^{-\frac{1}{2}}, & \text{when } z_i^* \leq 0 \end{cases}$$

The first term in the above equation places the most weight on the objective with the greatest relative ranges. The second term normalizes the gradients of the objective functions according to the L_2–norm.

Step 3:

Let $\mathbf{S}^{(1)} = \mathbf{S}$ and index $\mathbf{J}^* = \emptyset$.

$\mathbf{S}^{(1)} = \mathbf{S}$ means we begin the algorithm with the original (not yet reduced) feasible region. Index $\mathbf{J}^* = \emptyset$ designates the criterion values that are to be relaxed on the next iteration to allow achievement of others. At the beginning of the algorithm, $\mathbf{J}^*$ is empty because no solutions have yet been generated to relax.

Step 4:

Let $r = r + 1$. Calculate $\lambda_i^{(r)}$ minimax (Tchebycheff) weights, where

$$\lambda_i^{(r)} = \begin{cases} \frac{\pi_i}{\sum_{i=1}^k \pi_i}, & i \notin \mathbf{J}^* \\ 0, & i \in \mathbf{J}^* \end{cases} \tag{2.4}$$

The weights define the weighted Tchebycheff metric:

$$||\mathbf{z}^* - \mathbf{z}||_\infty^{\lambda_i^{(r)}} = \max_i \{\lambda_i^{(r)} |z_i^* - z_i|\}$$

On the first iteration $\lambda_i^{(r)}$ sum up to one, but on all subsequent iterations, $\lambda_i^{(r)}$ sum to less than one because $\mathbf{J}^* \neq \emptyset$.

Step 5:

Solve the weighted minimax (Tchebycheff) program:

$$\begin{aligned} \text{Min} \quad & \{\alpha\} \\ \text{Subject to}: \quad & \alpha \geq \lambda_i^{(r)}(z_i^* - z_i),\ i = 1,\ 2,\ ...,\ k \\ & \mathbf{x} \in \mathbf{S}^{(\mathbf{r})} \\ & \alpha \geq 0 \end{aligned} \tag{2.5}$$

where $\alpha \in \mathbb{R}$ represents the maximum weighted deviation of an objective from the ideal point i.e. $\alpha = \max_i \{\lambda_i(z_i^* - z_i)\}$. Here, we solve for the point in the reduced feasible region $\mathbf{S}^{(\mathbf{r})}$ whose criterion vector is closest to $\mathbf{z}^*$ according to the Tchebycheff metric defined by $\lambda^{(\mathbf{r})} \in \mathbb{R}^k$.

Step 6:

Let $\mathbf{z}^{(\mathbf{r})} = \mathbf{z}(\mathbf{x}^{(\mathbf{r})})$ (as obtained in step 5). Compare $\mathbf{z}^{(\mathbf{r})}$ with $\mathbf{z}^*$ since $\mathbf{z}^*$ is considered to be a good reference point for assessing the quality of a candidate criterion vector.

Step 7:

If all the values of $\mathbf{z}^{(\mathbf{r})}$ are satisfactory to the DM, stop. Otherwise continue with step 8.

Step 8:

Specify the index set $\mathbf{J}^*$ (if any) of satisfactory criterion values to be relaxed and specify the amounts ($\triangle z_j,\ j \in \mathbf{J}^*$) by which they can be relaxed.

Step 9:

Form a reduced feasible region

$$\mathbf{S}^{(\mathbf{r}+\mathbf{1})} = \left\{ \begin{array}{ll} \mathbf{x} \in \mathbf{S} \mid z_j(\mathbf{x}) \geq z_j\left(\mathbf{x}^{(\mathbf{r})}\right) - \triangle z_j, & j \in \mathbf{J}^* \\ \mathbf{x} \in \mathbf{S} \mid z_j(\mathbf{x}) \geq z_j\left(\mathbf{x}^{(\mathbf{r})}\right), & j \notin \mathbf{J}^* \end{array} \right\} \tag{2.6}$$

Then go to step 4.

The weights λ_j associated with the criterion $z_j,\ j \in \mathbf{J}^*$, are set equal to zero. This leads to smaller subsets of $\mathbf{S}$ as the process progresses.

Comments and Modifications

1. STEM was the first interactive procedure to have appeared in literature. It has opened a field of interactive methods' research.
2. STEM was initially proposed in the frame of multiobjective linear programming (MOLP), but the presentation of the method above is general enough to be adapted to other cases, such as problems involving an explicit list of alternatives.

3. The main disadvantage of the method is its *irrevocability*: when a special consideration is made on a criterion, it is definitively registered in the model. If the DM happens to change his/her mind, he/she is forced to start the procedure from the beginning. The method of Vincke [P76] (cf. [P92]), which performs an interactive sensitivity analysis using classical simplex properties, tries to overcome the problem of irrevocability in STEM. The sensitivity analysis helps the DM by giving lower and upper bounds for variations of the criteria due to a small change in one of them. The analysis proposed by Benayoun et al. (in the MOLP case) can produce bounds which do not correspond to feasible solutions.

4. It is not always easy for a DM to specify $\triangle z_j$, particularly if he/she knows the importance of this value in the procedure and the fact that it is irrevocable. Also, it would be more natural for the DM to specify the criteria to be improved, rather than those to be relaxed [P92].

5. Ringuest and Dowing [JC97] present an algorithm that operates in a similar way to STEM. STEM differs from their algorithm in that it is not based on any specific model of decision maker behavior.

2.3.2 Wierzbicki's Reference Point Method

Wierzbicki's reference point approaches were developed starting with research done at the International Institute for Applied Systems Analysis (IIASA), especially as a tool of environmental model analysis, although approaches found numerous other applications since that date (Wierzbicki [P99]). As mentioned earlier, Korhonen and Laakso consider the reference point methods as generalized GP. Their generalization tries to preserve main advantages of GP and overcome its basic disadvantages. According to its name, the *reference point method*, is based on the reference point. The reference point is a feasible or infeasible point in the criterion space considered by the DM to be reasonable or desirable. The reference point is used to derive an *achievement scalarizing function* as defined in section (2.1).

Wierzbicki's reference point approach is outlined as in the following steps:

Step 1:

Find the ideal criterion vector $\mathbf{z}^* = (z_1^*, z_2^*, ..., z_k^*)$ by maximizing each of k criteria individually.

Step 2:

Let counter $r = 1$. Let the DM express his/her reference point (aspiration levels) $q_i^{(r)}$ in relation to z_i^* , such that $q_i^{(r)} < z_i^*$ $(i = 1, 2, \ldots, k)$, $\mathbf{q}^{(r)} \in \mathbb{R}^k$. The vector $\mathbf{q}^{(r)}$ represents the reference point (aspiration levels for each criterion i.e. the levels of DM's achievement for each criterion).

Step 3:

Using $\mathbf{q^{(r)}}$ and $\mathbf{z^*}$ vectors, calculate λ–vector (representing the weights) corresponding to $\mathbf{q^{(r)}}$ as follows:

$$\lambda_i^{(r)} = \frac{1}{|z_i^* - q_i^{(r)}|} \left[\sum_{i=1}^{k} \frac{1}{|z_i^* - q_i^{(r)}|} \right]^{-1}, \; i = 1, 2, ..., k \qquad (2.7)$$

Note that λ_i's should sum to one $(i = 1, 2, ..., k)$ i.e. $\sum_{i=1}^{k} \lambda_i = 1$

Step 4:

Find $\mathbf{z^{(r)}}$ by solving the following achievement scalarizing program:

$$\text{Min} \qquad \alpha + \varepsilon \sum_{i=1}^{k} \lambda_i^{(r)} (q_i^{(r)} - z_i), \; \varepsilon > 0$$

$$\text{Subject to :} \quad \alpha \geq \lambda_i^{(r)} (q_i^{(r)} - z_i), \; i = 1, 2, ..., k \qquad (2.8)$$

$\mathbf{z^{(r)}}$ is essentially the projection of $\mathbf{q^{(r)}}$ onto the nondominated set of alternatives.

If the DM is satisfied with the solution $\mathbf{z^{(r)}}$ obtained, the process stops, otherwise go to step 5.

Step 5:

Let the DM update his/her aspirations and form yet another aspiration vector That is, let $r = r + 1$ and the process goes back to step 3

Comments and Modifications

1. Wierzbicki's reference point approach can be applied to any case, including problems involving an explicit list of alternatives (discrete MCDM problems), but are particularly well suited to application in problems with very large or infinite numbers of decision alternatives.

2. If the aspiration levels are not attainable, an achievement scalarizing function s (as defined earlier) generates an efficient point closest to the desired levels. If the aspiration levels are attainable with a surplus, s generates an efficient point while making the best possible use of that surplus. Wierzbicki [P83] calls this a $quasi - satisficing$ framework. The concept of quasi–satisficing of Wierzbicki basically describes how a computerized decision support system (DSS) should help a human DM [P99].

3. Reference point approaches require the measure of performance, z_i, to be available in a quantitative form [VJ02].

4. This approach follows a *learning–based* perspective (see the comments and modifications in interactive goal programming section for a learning–based interactive method).

5. The basic idea in Wierzbicki's reference point method is satisficing, rather than optimizing. Additionally, reference points are easy and intuitive for the DM to specify.

6. There is no clear strategy to produce a final solution since the method does not help the DM to find improved solutions.

2.3.3 Interactive Goal Programming

Interactive goal programming was developed for assisting decision makers in solving continuous multiple objective decision problems, even though it can now be used for all types of problems, including discrete decision problems. IGP has also been used for all types of goal programming models (linear GP, integer GP and non–linear GP). Any of the methods that are used to solve GP problems can be used as an interactive, sequential GP search methodology [J95]. The combination of these terms appeared in Masud and Hwang [ML81], which may have lead to the often used combination. A number of interactive goal programming ideas have been developed over the past years (although we shall give an overview of just a few).

Generalized Interactive Goal Programming

Reeves and Hedin [CR93] give the generalized interactive goal programming approach, where the nondominated solutions are guaranteed by using MINSUM forms of GP (which are fundamentally similar to the traditional GP model given earlier). Alternative solutions are obtained by adding constraints to assure that goals for each specific objective is attained for the subproblem. If none of the solutions obtained are satisfactory to the DM, the process involves setting new goal values, and regenerating the solutions. The generalization is outlined as follows:

Step 1:

Specify initial goal levels

Step 2:

Generate ideal solutions (primary initial solution)

Step 3:

If the solution is satisfactory, stop. Else go to step 4

Step 4:

Revise the goal levels

Step 5:

Regenerate the primary, alternative solutions

Tamiz and Jones [MF97] give a similar, but with some few modifications, general interactive goal programming process. Their process is outlined as follows:

Step 1:

Find the initial (feasible) solution

Step 2:

Present information from the current solution to the decision maker

Step 3:

If the DM is satisfied, stop. Otherwise go to step 4

Step 4:

Ask the DM to further express his/her preferences in some way

Step 5:

Reformulate the GP model in accordance with the information given in step 4

Step 6:

Reoptimize the GP model and go to step 2

Interactive Multiple Goal Programming

Interactive multiple goal programming (IMGP) method was introduced by Spronk [J81]. IMGP is based on reducing the set of feasible alternatives at each iteration until the DM is able and willing to choose the most satisfactory alternative among those remaining i.e. it is based on the pruning of the decision space.

IMGP algorithm is outlined as follows:

Step 1:

Construct a "potency matrix" P, consisting of the ideal $\mathbf{z}^*$ (a vector of maximum values for each criterion considered separately) and nadir (pessimistic) $\underline{\mathbf{z}}$ (a vector of minimum values for each criterion considered separately) values (which are usually infeasible) as depicted below:

$$P = \begin{bmatrix} z_1^* & z_2^* & \cdots & z_k^* \\ \underline{z}_1 & \underline{z}_2 & \cdots & \underline{z}_k \end{bmatrix} \tag{2.9}$$

The potency matrix has to be constructed at every current decision space.

Step 2:

Let the DM indicate the goal value(s) q_j (for j is the selected goal column), which should be improved and the value of improvement, looking at the pessimistic values (lower bounds) in the potency matrix.

If the DM is unable or unwilling to supply the exact improvement value, the goal q_j is assumed to be equal to $(z_j^* + \underline{z}_j)/2$.

Step 3:

Reduce the set of feasible alternatives by deleting all alternatives for which $z_j \geq q_j$ is not true (assuming z_j is to be maximized).

Step 4:

Present the new potency matrix resulting from the new reduced set of alternatives to the DM. If the DM is prepared to accept this restriction and its consequences on the other criteria values, this potency matrix becomes the new matrix to be considered with the second row as the new pessimistic starting solution. If the list of feasible alternatives has been reduced sufficiently to enable the DM to choose his/her most satisfactory solution from those remaining, the procedure terminates. Otherwise, go back to step 2.

Lexicographic (Preemptive) Goal Programming

Lee and Oslon [ML99] state that the *preemptive form of goal programming* can also be used interactively, as demonstrated by Franz and Lee [SM81]. Tamiz et al.[FE93] claim preemptive goal programming to be the most widely used form of GP. Romero [C04] adds that election of the superiority of preemptive goal programming's use over other GP variants is usually made in a rather mechanistic way without theoretical justification. Preemptive GP was introduced by Ijiri, Lee and Ignizio as mentioned before.

Comments and Modifications

1. The first interactive approach, specifically for goal programming was given by Dyer [S72]. This method finds a point for which an overall utility value of the solution is improved and moves part–way towards the new solution from the old solution in a manner which maximizes the overall utility value [MF97].

2. Interactive goal programming (IGP) is a relatively simple method which can be easily understood by the DM. This simplicity makes it to be

   easily computerized.

3. IGP tries to overcome the problem of standard GP, in which all objectives, target values, weights and priority levels should be set before solution

(i.e. the parameters are set *a priori* and there is no DM's intervention throughout the solution process).

4. Altering the current solution in most IGP problems involves the change in relative importance of different objectives to the DM. This importance is given by weighting or priority level structure. Another means of alteration involves the change of target (goals) of the objective.

5. There are several issues involved in the design of an interactive environment capable of processing problems solvable by current GP systems. Tamiz and Jones [MF97] outline this issues as:

   Choice of Initial Solution

   There are three approaches used to determine the starting solution.

   (a) *Infeasibly high (ideal) starting point.* From this point, the target (goal) values are progressively relaxed until feasibility is obtained. Its main disadvantage from the DM's point of view is that the solutions at each interactive iteration appear to be getting worse rather than better. Thus the DM may be unwilling to make the appropriate sacrifices.

   (b) *Infeasibly low (nadir) starting point.* In this case the minimal or low values for each objective are chosen. The DM then progressively improves the solution in accordance with his/her preferences e.g. Spronk's [J81] approach. It has an advantage of allowing the DM to 'build' the solution according to his/her preferences; however, for models with a large number of objectives, it is time-consuming to raise each objective from a low to a high value, as this requires many interactive iterations.

   (c) *Optimal starting point.* This approach either asks the DM for an initial estimate of the parameters or chooses them at random. The resulting goal program is then solved to give an initial point. It has an advantage of requiring fewer interactive iterations as it allows the DM's initial estimate. It has the disadvantage of psychologically narrowing the DM's focus to the area around the initial solution, since the DM may be unwilling to make major changes to his/her initial estimate.

   Terminating Conditions

   The final solution obtained is not a clear-cut issue. Thus, IGP (and other interactive) methods tend to lie somewhere between two extremes.

(a) *Search–based.* This approach tends to reduce the possible region in which the final solution lies at each iteration and is mathematically convergent. Thus the level of DM's satisfaction with current solution should improve at each iteration and backtracking is rarely allowed.

(b) *Learning–based.* This approach is designed to allow the DM the means of exploring the feasible region in a logical manner. It allows for backtracking in a systematic manner and terminates when the DM has decided that he/she is satisfied with the current solution or cannot improve upon a previous solution.

2.3.4 Interactive Weighted Tchebycheff Procedure

This method was proposed by Steuer and Choo [RE83]. It was basically developed for LP problems. The interactive weighted Tchebycheff procedure (IWTP) is a weight space reduction method. The set of nondominated solutions is generated and presented to the DM at each iteration of the solution process. The DM is asked to identify his/her most preferred solution from among the current set of nondominated solutions. Based upon the DM's response, the IWTP restricts the range of allowable weights for each objective, generates a new more concentrated set of nondominated solutions for DM consideration and the process continues.

The whole procedure is outlined in the following steps:

Step 1:

Let $r = 1$. Determine the ideal point $\mathbf{z}^*$. Let $\mathbf{z}^{**} = \mathbf{z}^* + \varepsilon$ ($\mathbf{z}^{**} \in \mathbb{R}^k$ is called the *utopian criterion vector*, it is the smallest value greater than $\mathbf{z}^*$), where ε is a vector of arbitrarily small positive values. The *utopian vector* is defined to be an infeasible criterion vector that strictly dominates every nondominated (Pareto optimal) solution.

Let

$$\Lambda^{(r)} = \{\lambda \in \mathbb{R}^k : \ \lambda_i \in [0,1], \ \sum_{i=1}^{k} \lambda_i = 1\}$$

be the initial set of weighting vectors (i.e $\lambda_i \in [l_i^{(r)}, u_i^{(r)}] = [0,1]$).

Step 2:

Randomly generate a large number ($\approx 50k$) of weighting vectors from $\Lambda^{(r)}$ (using a Steuer's [E86] computer program LAMBDA). Filter (using Steuer's [E86] computer program FILTER which can select a given number of λ–vectors, that are as far apart from one another as it is practicable

to compute) the weighting vectors to obtain a fixed number of representative weighting vectors. The proposed fixed number for the representative weighting vectors is $2R$ ($R \approx k$). Filtering tries to distinguish $2R$ vectors from one another.

Step 3:

For each representative weighting vector λ, solve the 2R weighted augmented weighted Tchebycheff program as follows:

$$\begin{aligned} \text{Min} \quad & \alpha + \rho \sum_{i=1}^{k} (z_i^{**} - z_i) \\ \text{Subject to:} \quad & \alpha \geq \lambda_i^{(r)} (z_i^{**} - z_i),\ i = 1,\ 2,\ ...,\ k \\ & \mathbf{x} \in \mathbf{S} \\ & \mathbf{z} \in \mathbf{Z} \\ & \alpha \geq 0 \end{aligned} \tag{2.10}$$

where $\alpha \in \mathbb{R}$ represents the maximum weighted deviation of the objectives from the utopian point i.e. $\alpha = \max_i \{\lambda_i (z_i^{**} - z_i)\}$ and ρ is a sufficiently small positive value.

Step 4:

Filter the criterion vectors resulting from Step 3 to obtain the R most different solutions. Present the R compromise solutions to the DM and ask him/her to select the most preferred solution. Let $\mathbf{z^{(r)}}$ be the selected point.

Step 5:

(a) If $r = t$ (where t is a prespecified number of iterations) then STOP, with $\mathbf{z^{(r)}}$ as the preferred solution. Otherwise if $r < t$,

(b) Let $\lambda^{(r)}$ be the weighting vector which generated $\mathbf{z^{(r)}}$ in step 4, with its components given as:

$$\lambda_i^{(r)} = \begin{cases} \frac{1}{|z_i^{**} - z_i^{(r)}|} \left[\sum_{i=1}^{k} \frac{1}{z_i^{**} - z_i^{(r)}} \right]^{-1} & , \text{ if } z_i^{(r)} \neq z_i^{**}\ i = 1,\ 2,\ ...,\ k \\ 1 & , \text{ if } z_i^{(r)} = z_i^{**} \\ 0 & , \text{ Otherwise} \end{cases}$$

Determine the reduced set of weighting vectors:

$$\Lambda^{(r+1)} = \{\lambda \in \mathbb{R}^k :\ \lambda_i^{(r+1)} \in [l_i^{(r+1)}, u_i^{(r+1)}],\ \sum_{i=1}^{k} \lambda_i = 1\}$$

where

$$[l_i^{(r+1)}, u_i^{(r+1)}] = \begin{cases} [0,\ g^r] & , \text{ if } \lambda_i^{(r)} \leq \frac{g^r}{2} \\ [1-g^r,\ 1] & , \text{ if } \lambda_i^{(r)} \geq 1 - \frac{g^r}{2} \\ [\lambda_i^{(r)} - \frac{g^r}{2},\ \lambda_i^{(r)} + \frac{g^r}{2}] & , \text{ Otherwise} \end{cases}$$

in which g $(0 \leq g \leq 1)$ is a prespecified *convergence/reduction factor* raised to the r^{th} power.

Let $r = r + 1$ and go to step 2.

Comments and Modifications

1. The method of Steuer and Choo can be applied in problems involving both implicit and explicit list of alternatives.
2. The method of Steuer and Choo is a *weighting vector space reduction* method for solving multiple objective problems.
3. The method has been designed to be easy to use for the decision maker, and, thus, complicated information is not required.
4. The preference information required from the DM is qualitative and rather natural, but however, it may become difficult to obtain as the number of criteria increases.
5. No assumption is made about any value function.
6. The numbers $50k$ and $2R$ are suggested as "rules of thumb". They can be changed by the analyst if so desired [E86].
7. The value of ρ between 0.0001 and 0.1 should normally suffice [E86].
8. The main disadvantage of the method is that many of its parameters (R, t, g) are prespecified without any intuitive meaning. This results in a somehow artificial stopping rule. Steuer [E86] suggests that the DM end the procedure when he/she is satisfied with the current solution, unlike when $r = t$ in step 5, i.e. whether $r < t$ or $r > t$. Thus it is believed that the parameter t should not be considered.
9. The correct selection of the reduction factor g is very important. The larger the reduction factor is, the faster the weighting vector space is reduced and the smaller the decision maker's possibilities for making errors and changing his/her mind about aspirations during the process.
10. It is suggested [RE83, E86] that the reduction factor g be such that $(1/R)^{1/k} \lessapprox g \lessapprox h^{1/(t-1)}$, where h is the final interval length of the weighting vectors, $\frac{1}{2k} \lessapprox h \lessapprox \frac{3}{2k}$, t is the number of prespecified iterations to be carried out, and $\lessapprox$ stands for "approximately equal or less".

11. The method is similar to Steuer's [E86] interactive weighted–sums/filtering method.

12. The weakness of this method is that too many calculations are needed at each iteration and many of the results are discarded. For large problems, where the evaluation of the values of the objective functions may be lengthy, the method of Steuer and Choo is not a realistic choice.

2.3.5 Visual Interactive Approach

The method was proposed by Korhonen and Laakso [JJ86]. This method extends the reference point approach by projecting a *direction* instead of a point onto an efficient frontier [AP01].

The method can be outlined as follows:

Step 1:

Let $r = 1$. Select an arbitrary point $\mathbf{z^{(0)}}$ in the criterion space.

Step 2:

Ask the DM to specify his/her aspiration levels (reference) point ($\mathbf{q^{(r)}}$) and take a vector $\mathbf{d^{(r)}} = \mathbf{q^{(r)}} - \mathbf{z^{(r-1)}}$ as a new reference point.

Step 3:

Solve the *achievement scalarizing parametric program* to project ($\mathbf{d^{(r)}}$, $\mathbf{z^{(r-1)}}$) onto the nondominated surface $\mathbf{Z}$:

$$\begin{aligned} \text{Min} \quad & \{\alpha + \varepsilon \sum_{i=1}^{k} z_i\} \\ \text{Subject to :} \quad & z_i + \alpha\lambda_i \geq z_i^{(r-1)} + \theta d_i^{(r)}, \text{ for } i = 1, 2, ..., k \\ & \mathbf{z} \in \mathbf{Z} \end{aligned} \tag{2.11}$$

as θ is increased from 0 to infinity.

α represents the maximum weighted deviation of an objective from a reference point.

Step 4:

Ask the DM to choose the compromise solution he/she most prefers. Let the chosen solution be $\mathbf{z^{(r)}} \in \mathbf{Z}$.

Step 5:

If $\mathbf{z^{(r-1)}} \neq \mathbf{z^{(r)}}$, let $r = r + 1$ and go to step 2. Else check the "optimality conditions" (cf. chapter 13 of Steuer [E86]). If the conditions are not satisfied, let $r = r + 1$, and let $\mathbf{d^{(r)}}$ be a new search direction identified by optimality checking procedure, and return to step 3.

Comments and Modifications

1. The visual interactive approach of Korhonen and Laakso can be applied to problems involving both implicit and explicit list of alternatives.

2. Korhonen and Laakso do not give a specific guidance as to the selection of an achievement scalarizing function s.

3. An "achievement scalarizing parametric program" projects a line segment (direction, instead of a reference point $\mathbf{q}$) that originates from the reference point $\mathbf{q}$ in the direction $\mathbf{d}$ onto the nondominated surface $\mathbf{Z}$.

4. The method mainly follows the *learning–oriented* perspective.

5. Optimality conditions are based on the assumption that the DM's value (utility) function is pseudo–concave and $\mathbf{S}$ is bounded and formed by linear constraints. The assumption is made only when the conditions have to be checked.

2.4 Interactive Methods Using Value Functions

In this section we discuss in more detail some interactive value function methods. The emphasis is on traditional value function methods and the methods that are implemented in the DSS.

2.4.1 Geoffrion–Dyer–Feinberg (GDF) Procedure

The method was proposed by Geoffrion, Dyer and Feinberg [AJA72]. The GDF method is used to solve value (utility) function problems as given below. It assumes that the arguments of the DM's value function consists of k objective functions.

$$\begin{aligned} &\text{Max} \quad V(\mathbf{z}) \\ &\text{Subject to}: \quad \mathbf{x} \in \mathbf{S} = \{\mathbf{x} \in \mathbb{R}^n : \ \mathbf{Ax} = \mathbf{b}, \ \mathbf{b} \in \mathbb{R}^m\} \end{aligned} \tag{2.12}$$

Since the value function is not explicitly known, the above problem cannot be solved directly [ARL82]. The solution to the above problem is accomplished by a series of linear approximations (Frank and Wolfe [MP56]) and through interaction with the DM. Solving decision problems with the GDF method requires the DM's assistance as the function and gradient evaluator. The Frank–Wolfe algorithm is chosen because of its simplicity and rapid initial convergence properties [E86]. Statements about convergence to a multiple criteria optimal solution can be made when:

1. $V(\mathbf{z}(\mathbf{x}))$ is concave

2. $V(\mathbf{z}(\mathbf{x}))$ is differentiable

3. the set $\mathbf{S}$ is convex

Definition

A subset $\mathbf{Z}$ of a vector space $\mathbb{R}^k$ is said to be *convex* if for every $z_1,\ z_2 \in \mathbb{R}^k$ and every real number α, $0 \leq \alpha \leq 1$:

$$M = \{u \in \mathbb{R}^k |\ u = \alpha z_1 + (1-\alpha)z_2\} \subset \mathbf{Z}$$

M is a closed, bounded segment, with any point $u \in M$ as its *interior point.* The definition means that a nonempty set $\mathbf{Z}$ is *convex* whenever all the points on the line segment joining any two points in $\mathbf{Z}$ also belong to the set $\mathbf{Z}$.

Definition

The function V: $\mathbb{R}^k \longrightarrow \mathbb{R}$ defined in a convex set $\mathbf{Z}$ is said to be *convex* if, for every $z_1,\ z_2 \in \mathbf{Z}$ and every $\alpha,\ 0 \leq \alpha \leq 1$, there holds:

$$V\left[\alpha z_1 + (1-\alpha)z_2\right] \leq \alpha V(z_1) + (1-\alpha)V(z_2)$$

If the above inequality holds as a strict inequality for $z_1 \neq z_2,\ 0 \leq \alpha \leq 1$, the function V is said to be *strictly convex.*

In other words, $V(\mathbf{z})$ is *convex* if it is "always bending upward", and for each pair of points joining on the graph of $V(\mathbf{z})$, the line segment joining these two points lies entirely "above" or on the graph of $V(\mathbf{z})$. To be more precise, if V has a second derivative everywhere, then V is convex if and only if $d^2V(\mathbf{z})/dz^2 \geq 0$ for all values of z [for which V(z) is defined]. Note that the second derivative can be used (when it exists everywhere) to check whether a function of a *single variable* is convex[2] or not, so second partial derivatives can be used to check functions of several variables, although it is a more complicated way.

Definition

The function V is said to be *concave* (*strictly concave*) if $-V$ is *convex* (*strictly convex*). $V(\mathbf{z})$ is *concave* if it is "always bending downward", and for each pair of points joining on the graph of $V(\mathbf{z})$, the line segment joining these two points lies entirely "below" or on the graph of $V(\mathbf{z})$. To be more precise, if V has a second derivative everywhere, then V is concave if and only if $d^2V(\mathbf{z})/dz^2 \leq 0$ for all values of z [for which V(z) is defined].

Basically, the GDF method follows the Frank–Wolfe algorithm, with the added feature of interaction with the DM ([FEL86]). The GDF algorithm would consist of the following steps:

[2]Bear in mind that this also applies to checking whether a function is concave or not

Step 1:

Find an initial arbitrarily chosen compromise solution $(\mathbf{x^{(1)}}, \mathbf{z^{(1)}})$.

Step 2:

Let $r = 1$. Determine the decision maker's local marginal rates of substitution (or tradeoffs) $\lambda_i^{(r)}$ between criterion z_i and z_1 (the reference criterion) at point $z^{(r)}$. This is done by pairwise comparison routine for obtaining the decision maker's preference weights. By definition, we have

$$\lambda_i^{(r)} = \frac{\partial V(\mathbf{z})/\partial z_i}{\partial V(\mathbf{z})/\partial z_1}$$

where $\partial V(\mathbf{z})/\partial z_i$ is the i^{th} partial derivative of the function V with respect to z_i.

For simplicity, the i^{th} criterion weight is calculated using the ratio

$$\frac{\Delta_1}{\Delta_i}$$

where Δ_i is the amount by which the i^{th} criterion is to be decreased to compensate for a Δ_1 increase in the first (reference) criterion, while holding all other criteria at their unperturbed values. For a given Δ_1, the purpose of the pairwise comparison process is to determine the Δ_i, $i \neq 1$, and hence the preference weights are (simply) given by

$$\lambda_i = \frac{\Delta_1}{\Delta_i}$$

The whole procedure is done interactively with the DM.

Step 3:

Determine a direction of improvement $\mathbf{d^{(r)}} = \mathbf{y^{(r)}} - \mathbf{x^{(r)}}$, where $\mathbf{y}$ is the solution to:

$$\begin{aligned} \text{Max} \quad & \nabla V(\mathbf{z}(\mathbf{x^{(r)}})) \cdot \mathbf{y} \\ \text{Subject to:} \quad & \mathbf{y} \in \mathbf{S} \end{aligned} \tag{2.13}$$

Using the chain rule of derivatives, the objective function can be expressed as:

$$\nabla V[\mathbf{z}(\mathbf{x^{(r)}})] \cdot \mathbf{y} = \left[\sum_{i=1}^{k} \left. \frac{\partial V}{\partial z_i} \right|_{\mathbf{z}=\mathbf{z}(\mathbf{x^{(r)}})} \cdot \nabla z_i(x^{(r)}) \right] \mathbf{y}$$

where $\partial V(\mathbf{z})/\partial z_i$ is the i^{th} partial derivative of the function V with respect to z_i, which is evaluated at the point $\mathbf{z} = \mathbf{z}(\mathbf{x}^{(r)})$. Dividing the objective function by the positive number $\partial V(\mathbf{z})/\partial z_1$ (the choice of the attribute to be the reference z_i is arbitrary), we get the problem:

$$\begin{aligned} \text{Max} \quad & \left[\sum_{i=1}^{k} \lambda_i^{(r)} \frac{\partial V}{\partial z_i}\bigg|_{\mathbf{z}=\mathbf{z}(\mathbf{x}^{(r)})} \cdot \nabla z_i(x^{(r)})\right] \mathbf{y} \\ \text{Subject to:} \quad & \mathbf{y} \in \mathbf{S} \end{aligned} \tag{2.14}$$

Step 4:

Determine the optimal solution to the one dimensional problem:

$$\begin{aligned} \text{Max} \quad & \nabla V[\mathbf{z}(\mathbf{x}^{(r)} + t\mathbf{d}^{(r)})] \cdot \mathbf{y} \\ \text{Subject to:} \quad & 0 \leq t \leq 1 \end{aligned} \tag{2.15}$$

The DM has to select the most preferred criterion vector, which provides the required value t^*. Let $\mathbf{z}^{(r+1)}$ be the selected point, which in turn defines the next feasible point $\mathbf{x}^{(r+1)}$, that is,

$$\mathbf{x}^{(r+1)} = \mathbf{x}^{(r)} + t^*\mathbf{d}^{(r)}$$

Step 5:

If $\mathbf{z}^{(r+1)} \approx \mathbf{z}^{(r)}$, then END with $(\mathbf{x}^{(r)}, \mathbf{z}^{(r)})$ as the final solution, ELSE let $r = r + 1$ and go to step 2.

Comments and Modifications

1. GDF method is based on the strong assumption that an implicit value (utility) function V, which must be maximized, pre–exists. All the answers given by the DM are to be consistent with the value function.
2. Many questions have to be answered at each iteration GDF method.
3. The only change at each iteration is in the objective function.
4. The main reason for selecting the Frank–Wolfe algorithm for GDF method is its (Frank–Wolfe algorithm) simplicity and robust convergence properties and the fact that, for the application of the method, only local information on the value function V is required. If the value functions were explicitly known, the Frank–Wolfe algorithm would terminate if $||\mathbf{x}^{(r-1)} - \mathbf{x}^{(r)}|| < \varepsilon$, where ε is a positive error bound, given *a priori*, without the DM's intervention throughout the whole solution process. Since V is not explicitly known, the DM's intervention is of great importance.

5. The solution can be found easily, since we end up with only one decision variable t, and the values of the functions $z_i(\mathbf{x}^{(\mathbf{r})} + t\mathbf{d}^{(\mathbf{r})})$ can either be calculated or plotted over the grid of points for $0 \leq t \leq 1$.

6. The major problem in the solution process is the DM's reluctancy to provide for the required preference information. DM's are usually reluctant to specify their tradeoffs. The determination of t^* also becomes difficult when the number of criteria k is too large.

7. Since preference information from one iteration is discarded before the next iteration in GDF method, Sakawa [M82] suggests the use of *proxy functions* as local estimates to value function. The proxy functions could be optimized directly instead of using the Frank–Wolfe algorithm, providing promise of much more rapid convergence to the DM's most preferred solution

8. Dyer [S72] presents a method called interactive goal programming, which is a combination of the GDF method and traditional goal programming.

2.4.2 Zionts and Wallenius (ZW) Method

The method was proposed by Zionts and Wallenius [SJ76]. The method also makes use of an implicit value (utility) function on an interactive basis. ZW method is dependent on the feasibility of using linear approximations to represent the constraint set and objective functions. The value function is assumed to be a linear function of the objective functions. It is essentially a *reduced weighting vector space* method for solving the MOLP problems. In solving ZW method, the following assumptions are made:

- All objective functions are concave
- The feasible set $\mathbf{S}$ is convex
- The value function V is a (pseudo)concave function over the objectives

In using the method, the DM chooses a set of positive weights and applies the weighting method to generate the nondominated solution. The subset of efficient variables (which cannot be increased without decreasing at least one other variable) is selected from the set of nonbasic variables, and for each efficient variable, a set of trade–offs is defined by which some objectives are increased and others decreased. These trade–offs are presented to the DM, who decides whether the trade–offs are desirable, undesirable or neither. From these answers a new set of weights is constructed, and the process starts from the beginning. The procedure terminates only if the DM finds a satisfactory solution.

The main steps of the algorithm are outlined below:

Step 1:

Let iteration $r = 1$.

An initial feasible set of weights is chosen as:

$$\Lambda^{(r)} = \{\lambda^{(r)} \in \mathbb{R}^k |\ \lambda_i \geq \varepsilon,\ \sum_{i=1}^{k} \lambda_i^{(r)} = 1\}$$

where ε is a sufficiently small positive value. The weights $(\lambda_i^{(r)})$ may be equal to $1/k$ for all i.

For $\lambda^{(r)} \in \Lambda^{(r)}$ solve the following linear program:

$$\begin{aligned} \text{Max} \quad & \sum_{i=1}^{k} \lambda_i^{(r)} z_i \\ \text{Subject to:} \quad & \mathbf{x} \in \mathbf{S} \end{aligned} \tag{2.16}$$

Let $\mathbf{x^{(r)}}$ be the optimal solution and $\mathbf{z^{(r)}}$ the corresponding (nondominated) criterion vector.

Step 2:

For each nonbasic variable x_j, and the parameters w_{ij} indicating the decrease in objective function z_i due to introducing a unit of the nonbasic variable x_j, test if the introduction of x_j into the basis leads to an efficient extreme point by solving the following linear program:

$$\begin{aligned} \text{Min} \quad & z_j = \sum_{i=1}^{k} w_{ij} \lambda_i^{(r)} \\ \text{Subject to:} \quad & \lambda_i^{(r)} \geq 0 \\ & \sum_{i=1}^{k} w_{ij} \lambda_i^{(r)} \ \geq\ 0 \ (i \neq j,\ j \in \mathbf{N}) \\ & \sum_{i=1}^{k} \lambda_i^{(r)} \ =\ 1 \end{aligned} \tag{2.17}$$

where $\mathbf{M}$ is the set of efficient[3] nonbasic variables, and $\mathbf{N}$ is the complement of $\mathbf{M}$ ($\mathbf{N}$ is the set of nonbasic variables that have not been declared

[3]Note that the words nondominated and efficient are sometimes used interchangeably

dominated).

Next, these tests are applied:

Test 1. If $z_j < 0$, then the nonbasic variable x_j leads to a nondominated solution.

Test 2. If $z_j > 0$, then x_j does not lead to a nondominated solution

Step 3:

This step of an algorithm involves an interaction with the DM.

For each of the efficient variables, the DM ought to answer whether he/she is willing to accept the trade–offs corresponding to simultaneous variations w_{j1}, w_{j2}, ..., w_{jk}; that is, the DM willing to accept the decrease in objective function z_1 of w_{j1}, and z_2 of w_{j2} and so on? The possible answer to this question are *yes, no* or *I don't know (indifferent to the trade–off).*

If the answer is *no* to all efficient variables, then the algorithm terminates, and the optimal solution of (2.17) above must be used as optimal weights, and the solution of problem (2.16) in step 2 with these weights will be accepted as the solution of the MOLP problem [($\mathbf{x}^{(r)}, \mathbf{z}^{(r)}$) is the final solution]. Otherwise, let indicator set $L = \mathbf{M}$ and reduce the set of weighting vectors $\Lambda^{(r)}$ to $\Lambda^{(r+1)}$:

- for each *yes* response and $x_j \in L$, $\lambda^{(r)} \in \Lambda^{(r)}$ by constructing an inequality

$$\sum_{i=1}^{k} w_{ij}\lambda_i^{(r)} \leq -\varepsilon \tag{2.18}$$

- for each *no* response and $x_j \in L$, $\lambda^{(r)} \in \Lambda^{(r)}$ by constructing an inequality

$$\sum_{i=1}^{k} w_{ij}\lambda_i^{(r)} \geq \varepsilon \tag{2.19}$$

- and for each *indifference* response and $x_j \in L$, $\lambda^{(r)} \in \Lambda^{(r)}$ by constructing an equation

$$\sum_{i=1}^{k} w_{ij}\lambda_i^{(r)} = 0 \tag{2.20}$$

for ε is a sufficiently small positive number.

Step 4:

Let $r = r + 1$. Find a new set of weights $\lambda^{(r)} \in \Lambda^{(r)}$, that are consistent with the DM's trade–off preferences by finding a feasible solution to the constraints (2.18, 2.19, 2.20) and

$$\lambda_i^{(r-1)} \geq \varepsilon, \ (i = 1, \ 2, \ ..., \ k)$$

$$\sum_{i=1}^{k} \lambda_i^{(r-1)} = 1$$

using linear programming with the objective function as

$$\text{Max} \quad \{\varepsilon\}$$

If $\Lambda^{(r)} = \emptyset$, progressively drop the oldest active constraints until $\Lambda^{(r)} \neq \emptyset$.

These new weights $\lambda^{(r)}$ are used in the next iteration (i.e. when $r = r+1$).

Solve the linear program in step 1 for the new weights $\lambda^{(r)} \in \Lambda^{(r)}$. Denote the corresponding solution $(\mathbf{x}^{(r)}, \mathbf{z}^{(r)})$.

Step 5:

Ask the DM to indicate which of $\mathbf{z}^{(r-1)}$ and $\mathbf{z}^{(r)}$ he/she prefers:

(a) if $\mathbf{z}^{(r-1)}$ is chosen, END with $(\mathbf{x}^{(r-1)}, \mathbf{z}^{(r-1)})$. In fact one could find better solutions but which are nonextreme points.

(b) if $\mathbf{z}^{(r)}$ is chosen, modify $\Lambda^{(r)}$ by adding a constraint

$$\sum_{i=1}^{k} (\mathbf{z}^{(r)} - \mathbf{z}^{(r-1)}) \lambda^{(r-1)} \geq \varepsilon$$

and go to step 2.

Comments and Modifications

1. The ZW method was restricted to MOLP problems. However, some extensions have been proposed in the multiple objective integer programming and discrete case (cf. Zionts [S77c, S89])

2. The method is based on the strong assumption of DM's implicit value function. The DM's answers have to be consistent with the value function.

3. Questions concerning trade–offs are generally considered to be too complicated. Hence, Zionts and Wallenius [SJ83] propose to replace trade–offs by pairwise comparisons between current and each of the adjacent non-dominated extreme solutions. The subsequent method seems to be easier to use.

4. Lara and Romero [PC92] successfully modify the ZW method for goal programming and apply it to a diet blending problem.

2.4.3 Method of Jacquet–Lagrèze, Menziani and Slowinski

The method was proposed by Jacquet–Lagrèze, Menziani and Slowinski [ERR87]. It is the methodology for multiobjective linear programming (MOLP) problems. The method consists of mainly three steps:

- Generation of a subset of feasible solutions (from 10 to 50).
- Assessment of an additive value (utility) function using an interactive method.
- Optimization of the additive utility function on the original set of feasible alternatives.

A detailed description of the method is outlined in the following steps:

Step 1:

Determine the payoff table of the MOLP problem and define the ideal point $\mathbf{z}^*$ and the nadir point $\underline{\mathbf{z}}$.

Calculate the weights λ_i such that:

$$\lambda_i = \frac{1}{z_i^* - \underline{z}_i},\ i = 1,\ 2,\ ...,\ k$$

Solve the weighted Tchebycheff problem as below to generate $\mathbf{x}^*$ and get the first (weakly) efficient point $\mathbf{z^0} = (z_1^0,\ z_2^0,\ ..., z_k^0)$:

$$\begin{aligned} \text{Min} \quad & \{\alpha\} \\ \text{Subject to}: \quad & \alpha \geq \lambda_i(z_i^* - z_i),\ i = 1,\ 2,\ ...,\ k \\ & \mathbf{x} \in \mathbf{S} \\ & \alpha \geq 0 \end{aligned} \tag{2.21}$$

where $z_i^0 = z_i(\mathbf{x}^*)$.

Considering s (s is arbitrarily chosen) interior points $\mathbf{z^0} - (r/s)(\mathbf{z^0} - \underline{\mathbf{z}})$, for $r = 1, 2, ..., s$, for each point, solve the following LP problem:

$$\begin{aligned} &\text{Max} && z_i(\mathbf{x}) \\ &\text{Subject to :} && z_j \geq z_j^0 - (r/s)(z_j^0 - \underline{z}),\ i, j = 1,\ 2,\ ...,\ k;\ i \neq j \\ & && \mathbf{x} \in \mathbf{S} \end{aligned} \tag{2.22}$$

This results in $k \times s$ weakly efficient points.

Step 2:

Ask the DM to estimate *piecewise–linear marginal value (utility) function*, v_i for each criterion z_i. This is achieved through interactive cycles of:

- direct estimations of v_i at some breakpoints using graphical displays (PREFCALC[4]).
- indirect estimations based upon ordinal regression methods (cf. Jacquet–Lagrèze and Siskos [EJ82] and Siskos [J85]). In this case the DM is first asked to rank the alternatives selected in step 1.

  The preference is assumed to be non–decreasing or non–increasing function for each criterion (i.e. $v_i(z_i)$ is monotone), and the overall preference can be represented by an additive piecewise–linear function of the form:

$$V(\mathbf{z}) = \sum_{i=1}^{k} v_i(z_i),\ v_i(z_i) \geq 0$$

Step 3:

Let $(z_{ih},\ v_{ih})$ be the break–points of $v_i(z_i)$ ($h = 1,\ 2,\ ...,\ \alpha_i;\ i = 1,\ 2,\ ...,\ k$). Then the objective function z_i and each partial utility function $v_i(z_i)$ can be expressed as a combination of the break–points:

[4]PREFCALC is a computer program that helps to estimate the DM's value (utility) functions

$$z_i = \sum_{h=1}^{\alpha_i} \lambda_{ih} z_{ih},$$

$$v_i(z_i) = \sum_{h=1}^{\alpha_i} \lambda_{ih} v_{ih}$$

$$\sum_{h=1}^{\alpha_i} \lambda_{ih} = 1,$$

$$\lambda_{ih} \geq 0 \; for \; h = 1,\ 2,\ ...,\ \alpha_i;\ i = 1,\ 2,\ ...,\ k$$

Determine the final solution by $\mathbf{z}$ by solving the following single objective value function problem:

$$\text{Max} \qquad V = \sum_{i=1}^{k} \sum_{h=1}^{\alpha_i} \lambda_{ih} v_{ih}$$

$$\text{Subject to}: \quad \sum_{j=1}^{n} a_{jl} x_j = b_l,\ l = 1,\ 2,\ ...,\ m$$

$$\sum_{j=1}^{n} c_{ij} x_j - \sum_{h=1}^{\alpha_i} \lambda_{ih} z_{ih} = 0,\ i = 1,\ 2,\ ...,\ k$$

$$\sum_{h=1}^{\alpha_i} \lambda_{ih} = 1,\ i = 1,\ 2,\ ...,\ k$$

$$\lambda_{ih},\ h = 1,\ 2,\ ...,\ \alpha_i;\ i = 1,\ 2,\ ...,\ k$$

$$x_j \geq 0,\ j = 1,\ 2,\ ...,\ n \qquad (2.23)$$

Comments and Modifications

1. Although the method of Jacquet–Lagrèze et al. was proposed for MOLP problems, it can be applied to problems involving an explicit list of alternatives.

2. The method is adapted for large scale linear programs where traditional multiobjective methods would be too costly to use, since the interactive phase is limited to step 2, using PREFCALC [ERR87].

3. If the values taken by different objective functions differ a lot for the same $\mathbf{x}$, the values have to be normalized by multiplying each objective i by $\alpha_i = 1/\left(\sum_{j=1}^{n} C_{ij}^2\right)^2$, where C_{ij} are the coefficients of decision variables x_j for each objective i $(i = 1, 2, ..., k; j = 1, 2, ..., n)$.

4. No irrevocability is imposed as long as the assessment is not completed i.e. the DM is free to adjust his/her value function.

2.4.4 Generalized Interactive Value Function Method

The generalized interactive value function method was introduced by Stewart [J99b]. The approach was introduced to overcome some shortcomings (see Belton and Stewart [VJ02], chapter 6 for details) of the method of Zionts and Wallenius. The generalized interactive value function method is based on a piecewise linear value function, similar to that in the previous section. The interactive part of the method is based on providing tradeoffs between criteria in the vicinity of a particular solution to the MOLP problem. One of the attributes from the provided solution is selected as a reference. Then the DM is asked to state for each of the other attributes, the amount that could be sacrificed in order to gain a specified increase on the reference attribute. The tradeoff information from the DM creates additional constraints on the value function model.

Belton and Stewart [VJ02] summarize the whole procedure as in below:

Step 1:

Let v_{id} $(i = 1, 2, ..., k; d = 1, 2, ..., p;$ where k is the number of criteria and p is the number of piecewise linear segments) be the values which fully define[5] the marginal value function $v_i(z_i)$ based on p piecewise linear segments. Select an arbitrary initial guess for the v_{id} values.

Step 2:

Maximize the value function $\widehat{\mathbf{V}}(\mathbf{z})$ defined by the current estimates v_{id} subject to $\mathbf{x} \in \mathbf{S}$. Denote the solution by $(\mathbf{x}^+, \mathbf{z}^+)$. If (in the judgement of the DM) the results seem to stabilize, then STOP; otherwise go to the next step.

Step 3:

If the tradeoffs have not yet been assessed at the point represented by $\mathbf{z}^+$, then set $\mathbf{z}^* = \mathbf{z}^+$. Otherwise select a distinctly different attribute vector $\mathbf{z}^*$ at which the tradeoffs have not been assessed.

[5]The v_{id} are the slopes of the marginal value function in each segment

Step 4:

Select one criterion (without loss of generality denoted as criterion 1) as a reference, then obtain tradeoffs on the other criterion corresponding to a fixed increment on the reference criterion (i.e. the amount, say δ_i which the decision maker would give up on criterion i in order to obtain a specified increase, say δ_1 on the reference attribute. This step results in additional constraints in the problem.

Step 5:

With the addition of the latest set of constraints on the v_{id} generated in step 4, solve the LP minimizing the maximum overall deviational variables subject to the non–negativity, scaling and shape constraints on v_{id} and return to step 2.

Comments and Modifications

1. The method can be used in both discrete and continuous MCDM problems.
2. The numerical results reported in [J93] indicate that the procedure tend to converge after about 6 or 7 iterations.

2.5 Applications of Interactive Methods

In this section we review and classify the (combinations of) interactive methods. We also take a look at some of interactive methods' applications reported in literature. The methods in consideration are those which use the DM's aspiration levels (reference point) and those which use the DM's underlying value (utility) functions as the DM's preference.

2.5.1 Applications of Generalized GP Methods

Buchanan [T97] presents a naïve MCDM solution technique called GUESS. The method is an interactive solution method designed for continuous multicriteria decision problems. It is a reference point method. He states that GUESS has been used typically as a benchmark in MCDM experiments where different solution methods have to be compared. In this method, the DM is required to provide for aspiration vectors (which Buchanan calls *guesses*). There is no guidance given to the DM other than the solutions which result from the guesses. The solution process terminates when the DM is, in some way, satisfied. The "engine" of GUESS solution method is a simple reference point method which seeks to maximize the minimum weighted achievement for the nadir (a *max–min approach*). Properties of max–min formulations can be found in Buchanan and Gardiner [TL95]. On the other hand, Stewart [J99a] points out that the application of this kind of procedure can lead to premature termination of procedure, at a relatively poor solutions (i.e. relative to the idealized preferences), leaving much of the decision space unexplored by the DM.

Kompan and Bera [MA95] applied GP in project selection decision for Indian coal mining industry. The proposed GP model was aimed at selecting the most suitable subset of nine projects among a set of feasible project proposals as an accepted investment plan with simultaneous attainment of all goals as best as possible. Two variants of GP models being preemptive GP and Archimedian model were basically employed in that project.

Nhantumbo et al.[IBG01] applied GP in the management of the Miombo Woodland in Mozambique. The essence of the project was to assist in the government's policy to manage the natural resources in partnership with the rural communities and private sector. The underlying principle in employing GP in woodlands management planning problem was to minimize the sum of deviations (under– or over–achievement) from goal levels set by the stakeholders[6]. Nhantumbo et al.[IBG01] state that the challenging aspect of many natural resources planning frameworks is that they inherently involve various goals within which each group of stakeholders and an integration of the interests of various users multiplies the number of potentially conflicting goals.

Urli and Nadeau [BR04] propose an interactive method for multiobjective stochastic linear programming problems. Their method can be decomposed into two main phases. The first phase (called the phase of modelling) transforms the initial stochastic program into a corresponding multiobjective deterministic program. In the second phase (called the resolution phase), the corresponding deterministic program is solved based on STEM [RJJO71], which is enriched with some parametric analyses. The analyses are relative to the improvement of criteria and the relaxation of constraints, and those tools are used to help the DM in the search for a compromise solution. They term their interactive approach PROMISE/scenarios (in French, PRogrammation Multiobjectif Interactive StochastiquE).

Novak and Ragsdale [CT03] develop a methodology for solving stochastic, multicriteria LP problems in spreadsheets. Their methodology solve a weighted Tchebycheff program to identify a nondominated solution for a current problem. The weights for the program are randomly generated. The method consists of what they call *re–evaluation*, which provides an indication of how robust a particular nondominated solution is to uncertainty.

Alves and Clímaco [JC00] propose an interactive reference point method for multiple objective (mixed) integer linear programming problems. Their model employs the use of branch–and–bound for solving the scalarizing programs. At each phase, the DM has to specify a criterion reference point or choose the objective function he/she wants to improve in respect to the previous nondomi-

[6] A stakeholder is someone with an interest or concern in business, enterprise, economy or society.

nated solution. Tchebycheff mixed–integer scalarizing programs are successfully solved by branch–and–bound.

Blake and Carter [TW02] propose a methodology for allocating resources in hospitals. Their methodology uses two linear GP models. The first GP model has two preemptive goals. The second GP model has six preemptive goals. Foued and Sameh [BM01] propose a multiobjective approach to solve the problem considering the releases, the pumping cost and salinity of water. Their model is used for the operation and control of multipurpose reservoir systems. The model employed is based on the stochastic GP approach.

Hajidimitriou and Georgiou [AC02] present a quantitative model based on the goal programming technique, which uses appropriate criteria to evaluate potential candidates and leads to the selection of the optima partner for International Joint Ventures. In their model, Hajidimitriou and Georgiou establish aspiration levels and set priorities for various goals that will be reflected in a lexicographic objective function.

Sun [M00] presents a multiple objective programming approach for determining faculty salary equity adjustments. The analysis and decision making process is interactive. Through its interactive process, the approach allows the DMs to formulate the model by adding, deleting and/or modifying constraints so as to reflect the budgetary and policy restrictions of the institution. The interactive solution process also incorporates the DM's preference information in the search process for the preferred solution. The model is in principle similar to the Combined Tchebycheff/Aspiration Criterion Vector Interactive Procedure (Steuer and Choo [RE83], Steuer et al.[EJW93]). He mentions that GP may also be used to solve the problem, but he does not recommend it due to its drawbacks.

Hämäläinen and Mäntysaari [PJ02] use dynamic goal programming model in the problem of space heating under time varying price electricity. In their model, goals are ideal temperature intervals and other criteria are costs and energy consumption. The use of interval goals to model the DM's preferences was recently used and strongly brought up by Martel and Aouni [JB90, JB98]. They discuss the modelling requirements in multicriteria problems with a dynamic structure and present (what they call) "a new relaxation method combining the traditional ϵ–constraint and goal programming methods". They comment that it is surprising that very few papers have suggested using GP in dynamic situations. They term this method an ϵ-constraint GP (GP_ϵ).

Lotfi et al.[VYS97] develop an interactive method for MOLP based on aspiration levels of a DM. The method assumes an unknown pseudo–concave preference structure of the DM throughout the decision process, and the decision maker's ability to select a preferred solution from $k+1$ alternatives, where k is the number of objectives. Their approach utilizes a Tchebycheff function that

facilitates attainment of an optimum at a non–extreme point solution. Their solution method is termed "Aspiration–Based Search Algorithm" (ABSALG) for MOLP. They conclude their research by comparing ABSALG with two other interactive MOLP methods; those of Reeves and Franz [RL85] (SIMOLP), and Steuer and Choo [E76, RE83]. The latter methods are comparable to ABSALG in that they are interactive and ask similar questions of the DM.

Gass and Roy [IG03] propose a method for ranking the full set or a subset of efficient extreme point solutions for MOLP problems with $k(\geq 2)$ objective functions. Their idea was to enclose the given efficient solutions as represented by k–dimensional points in the objective space within an annulus of minimum width, where the width is determined by a hypersphere that minimizes the maximum deviation of the points from the surface of the hypersphere. This is achieved by using the Tchebycheff method. They argue that the hypersphere represents a surface of compromise and that the point closest to its surface should be considered as the "best" compromise efficient solution. Their compromise procedure attempts to put a specific meaning to "somewhere in the middle", following Wierzbicki's [P99] statement that:

> by a *neutral compromise solution*, we understand typically in multicriteria analysis, a decision with outcomes located somewhere in the middle of the efficient set; a more precise meaning of 'somewhere in the middle' specifies the type of compromise solution.

Given a ranked (sub)set of efficient solutions, a procedure is given that associates to each efficient solution a set of k positive weights that causes the nondominated solution to be optimal with respect to the given set of nondominated solutions.

Sun et al [MAE00] develop an interactive multiple objective programming procedure that combines the strengths of the interactive Tchebycheff procedure (Steuer and Choo [RE83]) and the interactive FFANN (Feed–Forward Artificial Neural Networks) procedure (Sun, Stam and Steuer [MAE96]). In this interactive procedure, nondominated solutions are generated by solving the augmented weighted Tchebycheff programs. The DM indicates preference information by assessing "values" to or by making pairwise comparisons among these solutions. The revealed preference information is then used to train a feed–forward artificial neural network. The trained feed–forward artificial neural network is used to screen new solutions for presentation to the DM on the next iteration. They randomly generate their test problems using Steuer's ADBASE. In the computational experiments , linear, quadratic, L_4 and L_∞ (Tchebycheff metric) value functions were used to simulate a DM in evaluating the attractiveness of the solutions generated during the interactive solution process.

Gardiner and Steuer [RE94] propose a unified interactive multiple objective programming model. They demonstrate how the different interactive methods can be made to fit a single algorithmic outline. Their unified approach supports

what they call *procedure–switching*, thus enabling the user start with one procedure and switch to other procedures during the interactive process if so desired. A unified algorithm embracing STEM, interactive GP, Wierzbicki's reference point approach and five other interactive procedures, is used to show the degree to which the procedures of interactive multiple objective programming can be folded into one another when pursuing the unified strategy.

2.5.2 Applications of Value Function Methods

Ehrgott et al.[MKC04] propose a model for portfolio optimization extending Markowitz [H52, H59] model. Markowitz covariance model is based on two conflicting optimization criteria whereby the risk of a portfolio is represented by its variance to be minimized and the expected return of the portfolio is to be maximized. Conversely, the method of Ehrgott et al. has five specific objectives related to risk and return and allows consideration of the individual preferences through the construction of DM's specific utility (value) functions and a final additive global utility function[7]. The utility functions v_i in their model are monotone (which implies V to be quasi–convex). They randomly generate the breakpoints for the piecewise linear and piecewise quadratic value functions.

Gómez–Limón et al.[AMR03] present the methodology based on multiple criteria mathematical programming to obtain relative and absolute risk aversion coefficients. The method relies on multiattribute utility theory to estimate the risk aversion coefficients. Their model uses the additive linear utility function. They use GP to calculate the weights for the utility function. The proposed model is the modification of the model of Sumpi et al.[MFC93, MFC97]. Their method ensues in the additive function that can be used as an instrument which is capable of approximating the observed behavior of the DM.

Malakooti and Al–alwani [BaJE02] present the fundamental theory and algorithms for identifying the most preferred alternative for a DM having a non–centrist (or extremist) preference behavior. Their interactive method solves the MOLP problem in a way that the DM is requested to respond to a set of questions in the form of paired comparison of alternatives. The DM's underlying preference function is represented by a quasi–convex value (utility) function, which is to be maximized. Their method requires only a finite number of pivoting operations using a simplex–based method, and it asks only a limited number of paired comparison questions of alternatives in an objective space. They develop a branch–and–bound algorithm that extends a tree of solutions at each iteration until the MOLP problem is solved. At each iteration, the DM has to identify the most preferred alternatives from a given subset of efficient alternatives that are adjacent extreme points to the current basis. Through the branch–and–bound algorithm, without asking many questions from the DM,

[7] In an additive value function, V is basically a global value function composed of the sum of k (local) value functions for each objective i (i = 1, 2, ..., k)

all branches of the tree are implicitly enumerated until the most preferred alternative is obtained. Malakooti and Al–alwani state that in most interactive methods, the value (utility) function (that presents the DM's preference) is assumed to be linear or additive, concave, pseudo–concave, or quasi–concave. They add that, however there has not been any effort to recognize and solve the quasi-convex utility functions which present an extremist preferential behavior, while the other aforementioned methods represent a conservative behavioral preference.

Aghezzaf and Ouaderhman [BT01] present an interactive interior point algorithm for determining a best compromise solution to a multiple objective linear programming problem in situations with an implicitly defined utility function, which is illustrated by deriving the local tradeoffs with the decision maker. The partial information on the utility function of interest is the local trade–offs (marginal rates of substitution) between the objective functions, which are obtained by interacting with the DM. This follows the concept first proposed in MCDM by Geoffrion et al [AJA72]. The marginal rates of substitution are used in deriving the approximation gradient and to update the lower bounds of the objectives. The algorithm is based on the modification of the method of centers to MOLP problems proposed by Trafalis et al.[TTS90].

Chen and Lin [JS03] present an interactive approach for solving multiple criteria decision making (MCDM) problems based on Decision Neural Networks (DNN). The DNN is used to capture and represent the DM's preference. With the DNN, an optimization problem is solved to search for the most preferred solution. The DNN procedure is more similar to the way the DM presents his/her preference information. An interactive DNN approach developed in their paper can be used without assuming any specific function form of multi–attribute utility function (MAUF). With the DNN, the DM can answer much fewer questions regarding pairwise assessments comparing with the traditional interactive procedures.

Kim and Choi [KH01] suggest an interactive procedure for solving a multiattribute group decision problem using range–type utility information and develop an interactive group support system. To implement the suggested interactive procedure, a group decision support system (DSS) termed RINGS (utility Range based INteractive Group support System) was developed. Utility ranges are obtained by solving linear programming (LP) problems with incompletely specified information. RINGS finds out conflicting opinions among group members, compares each member's preferences with each other, and suggests a direction for consensus seeking. All members' utility values are represented by ranges within 0 and 1. As a way of combining individual utility ranges into a common range of the group, Kim and Choi define two types of utility ranges which are termed *total range* and *agreed range*. To make a group consensus, the procedure interactively modifies the DM's incomplete information to be concrete and complete.

Borges and Antunes [RH03] present the effects of uncertainty on MOLP models

using the concepts of fuzzy set theory. The proposed interactive decision support system (DSS) is based on the interactive exploration of the weight space, objective function space as well as numerical information obtained from the DM in each iteration. They find the comparative analysis of the weight spaces corresponding to distinct satisfaction degrees as a valuable tool for studying the fuzzy nondominated solution set. Among these fuzzy solutions, the DM may choose a satisfactory compromise solution according to his/her preference structure, which may change as more knowledge and understanding of the problem is acquired throughout the interactive decision aid process. There is no irrevocable decisions throughout the interactive process, and the DM is always allowed to revise prior preference information and exploit new search directions.

2.5.3 Combinations of Generalized GP and Value Function Methods

Stewart [J99a] compares the value functions and aspiration level approaches, and identifies the critical aspects of aspiration levels formulation. He proposes a refined interactive aspiration–based method for multicriteria problems. His method tries to prevent early termination of the solution process and provides a guidance for changes in aspiration levels in a more sensibly achievable directions, by minimizing a scalarizing function over the set of alternatives 'excluding' the most recent ones seen by the DM. This results in a broader exploration of the decision space. The simulated procedure stops when no improvements are reported by the DM over the specified number of iterations. The main disadvantage of this refined method is the inability to allow the DM's backtracking. Stewart further states that aspiration–based procedures can be applied with some confidence and convergence to reasonable solutions, even when there is no full understanding of all behavioral aspects of implementing reference point or goal programming methods.

Ballestero [E97] illustrates that a utility function with separate variables (presented in a form of a Taylor series around the ideal point) is reducible to a weighted sum of compromise programming (CP) distances. This follows the consideration of utility theory and compromise programming as different methodologies to measure preferences as well as to determine DM's optima on an efficient frontier. The linkage between utility and compromise methods (based on the main assumption in which the usual utility (value) function holds) leads to a method for specification and optimization of usual utility functions by operational technique, and a reformulation of standard CP with the advantage of determining the best CP solution from a utility perspective.

Romero [C04] points out that

> the system based upon penalty functions is the usual procedure to incorporate the DM's preferences into a GP model. However, there are other types of functions that can be used to address the same

> problem like utility and satisfaction functions ... It is straightforward to derive for each penalty function, the corresponding utility function, and vice–versa.

Tamiz et al.[MDC98] discuss the use of GP and utility (value) functions. They discuss the non–compatibility between LGP and utility functions. That is, an LGP model does not optimize the DM's utility function. The non–compatibility lies in the non–continuity of preferences inherent to lexicographic orderings. The non–continuity of orderings implies the impossibility of ordering the DM's preferences by a monotonic numerical representation or utility function [C91]. On the other hand Ignizio [P82a] argues in favor of a lexicographic utility function, claiming the utility function of a human life cannot be directly compared with the utility function of cash–savings when constructing a highway. Romero [C91] also defends lexicographic utility functions; he also explains that many of these cases of GP producing incorrect or unexpected results are due to incorrect modelling practices rather than a fundamental defect in the theory of GP.

Tamiz et al [MDC98] further state that MINIMAX GP can be considered as a specification of a utility (value) function known as Rawlsian function (only when the targets are fixed at an ideal values). This kind of utility (value) function is usually called a Rawlsian function because of the connections between it and the principles of justice introduced by Rawls [J72]. Generally, for any vector of targets (with no anchors), the solution provided by a MINIMAX GP model coincides with, or is the nearest possible solution to a Rawlsian perfectly equilibrated solution. Hence solutions of this form as well as the corresponding utility function are called *quasi–Rawlsian*. Tamiz et al. add that for different kinds of scalarizing functions (see section 2.1 of chapter 2), if $\varepsilon \to \infty$, the function becomes an *additive linear utility (value) function*. For $\varepsilon = 0$, the function becomes a pure *Rawlsian function*. For small values of ε, the function becomes a *quasi–Rawlsian* function.

Dyer [S77b] demonstrates how the underlying utility function of a weighted GP (WGP) is additive and separable. Moreover, if in WGP model, the targets have been set at their anchor values, a linear and additive utility (value) function is maximized. Therefore, WGP can be viewed as a specification of an additive utility (value) function, so the assumption of mutual preference independence must hold [L92].

Romero et al.[CMF98] look at the connections between multi–criteria techniques of goal programming, compromise programming and the reference point method. They demonstrate the means of reducing all compromise programming and reference point method models to corresponding GP formulations. They also give an interpretation in terms of utility functions of the solutions provided by each of the three approaches.

2.6 Concluding Remarks

It seems to be easier to work with generalized GP interactive methods, than working with value function interactive methods. This is justified by recent review of literature given in the previous sections. There appear to be a lot more researchers working on reference point methods than those working with value function methods. This might be due to a difficult task of estimating the value functions representing the DM's preference information in value function methods. Many researchers try to tackle this problem by using computer graphics, even though it is still not easy.

It is a common practice in MCDM to present models and methods in a disconnected way, giving the wrong impression that each approach is autonomous and completely independent from other MCDM approaches (Romero [C01]). Following Romero's statement and the literature review, it is evident that quite a small number of researchers is working on the combination of value function methods and (mainly) goal programming. The literature review further shows that there is a room for additional thought and research into a general framework or methodology for the use of value (utility) function theories and the GP approaches. It also appears that the use of value (utility) theories is only incorporated to standard GP in particular, and not to any other methods for generalized GP. It seems that there have been few real–world applications of such combinations. There also seem not to have been any computerization of such combinations. Only the theoretical aspects of the combinations appear in literature.

The combinations of generalized goal programming and value function method, which appear in literature, do not clearly illustrate how the methods can be used interactively. It would be of great importance to make further research on how to make use of such combinations in an interactive mode.

Chapter 3

Towards the Development of a Decision Support System

In the previous chapters, we discussed various interactive MCDM techniques. It is, however, of significant advantage that those techniques be incorporated into software system in an integrated manner, so that the decision maker could use such a system to investigate his/her decision problems using several methods without being bothered by the mathematics involved.

Decision making often requires the analysis of large amount of data and complex relations. Computerized tools designed and implemented for such purposes are called decision support systems (DSS). A computer, for its part, is obviously much faster and more accurate than a human in handling massive quantities[1] of data, hence the goal of a DSS is to supplement the decision powers of the human with the data manipulation capabilities of the computer [J87]. A DSS, which is typically a problem specific tool, usually helps in the evaluation of consequences of a given decision and may advise what decision would be the best for achieving a given set of goals [M94a]. In such cases, an analysis of a mathematical model can support rational decision making [M94a]. Dutta [A96] further notes that mathematical models, particularly optimization models, have been, and continue to be the workhorses for decision support. Licker [S97] adds that the most common form of DSS is a spreadsheet.

Hence the basic purpose of this chapter is to deal with the theoretical background of the development of what is known as integrated multiple criteria decision support system (IMC–DSS) [PJ98]. Decision support systems are con-

[1] Handling massive quantities of data should be understood not only in a traditional sense (as data processing done by a Data Base Management System) but it can also analyze a large amount of logical relations and/or solve mathematical programming problems [M94a]

sidered as a subset of Management Information Systems (MIS), which include all systems that support any management decision making. Sometimes different terms such as Management Information Systems (MIS), Strategic Information Systems (SIS), Expert Systems (ES), Intelligent Decision Support Systems (IDSS) or DSS are used for similar methodology and/or type of application [M94a]. The development of a DSS includes planning, design, implementation, testing and incorporation into the user's work. There are various definitions of decision support systems, and each stresses a different aspect. Since there are a lot of ways to make decisions, there are a lot of DSS models as well, but most of them are activated to produce the results that assist rather than take over decision making in an unstructured problem-solving environment. The real purpose of a DSS is to provide support to the decision maker (DM) in the process of making decisions [S97].

3.1 Definitions of a Decision Support System

As mentioned above, there are essentially many definitions of a DSS, however the different definitions address some common themes and a focus on those themes should yield a good working definition of the concept. Some of the various DSS definitions are given below as outlined in [S97]

- A model–based set of procedures for processing data and judgements to assist a manager in decision making.
- An interactive computer–based system intended to help decision makers utilize data and models to solve unstructured problems.
- A system coupling intellectual resources of individuals with the capabilities to computers to improve the quality of decisions; a computer–based support system for management DMs who deal with semi–structured problems.
- A system under the control of one or more DM's that assists in the activity of decision making by providing an organized set of tools intended to impart structure to portions of the decision making situation and to improve the ultimate effectiveness of the decision outcome.

Following the above definitions, a DSS can be seen as the combination of the strengths of computers and the strengths of the human DMs.

3.2 Types of Decision Support Systems

Many different functions that can or should be provided by a DSS can be divided into two sets [M94a]:

1. Data processing in the traditional sense: These functions provide selection retrieval, and presentation of information previously stored in a data base. Such functions are typically supported by a Data Base Management System (DBMS) and are useful for many everyday managerial activities such as producing various reports, answering *ad hoc* queries, presenting information in different forms, etc.
2. Model processing: Quite often, it is desired to predict consequences of some actions (for example implementing a decision or making a choice) or events (actions that are not controlled by a DM). In such cases, a mathematical model for a real situation is built and such a model is used for analysis of predicted consequences.

A DSS that supports only functions from the first set is called a data–oriented DSS. A DSS that uses an underlying mathematical model is called a model–based DSS. A mathematical model is built for a part of the decision making process (DMP) for which it is possible to implement an abstract (mathematical) model that is good enough to represent the available knowledge and experience of a DM in order to support his/her intuition [M94a]. Makowski further points out that quite often, DMP requires not only data processing in the traditional sense, but also requires the analysis of a large number of logical and analytical relations and processing– in the sense of solving an underlying mathematical model – a large amount of data. He adds that in such situations, a properly designed and implemented model–based DSS not only performs cumbersome data processing, but it also provides relevant information that enables a DM to concentrate on those parts of the DMP that cannot be formalized. We shall basically focus on the model–based DSS in this dissertation.

3.3 Model–based DSS

A mathematical model describes the modelled object by means of variables, which are abstract representations of the elements of the object that the users of the model want to study [MP03]. Actual model building is still a mixture of art and science that requires knowledge and experience, including a good understanding of the problem, good knowledge of model building methodology, and understanding of solution techniques that will be used for processing the model [M94b]. When a model–based DSS is desired, it can be achieved only for a problem that is understood sufficiently well to build a mathematical programming model, which can adequately represent a decision situation [M94a]. Such a model – that implicitly defines a set $\mathbf{S}$ of alternatives – is typically composed of [PM92]:

- decision variables that represent actual decisions (alternatives, choices, options etc.).

- potential objectives (goals, performance indices) which can be used for evaluation of consequences.
- various intermediate and parametric variables (balance and/or state variables, external decisions).
- constraining relations (inequalities, equations etc.) between variables that indirectly determine the set of feasible decisions. Some of the constraints may reflect the logic of handling events represented by variables.
- outcome relations that define goals as functions of decision relations.

More generally, the mathematical programming model to be solved mainly consists of two parts [M94a]:

- A constant and usually large *core model*. The so–called *core model* accounts only for logical and physical relations between all the variables. The core model implicitly defines the set of feasible solutions **S**. It should not contain any constraints corresponding to a preferential structure of the user. It is typically composed of decision variables that represent actual decisions (alternatives, choices, options, etc.), variables defining potential criteria (objectives), constraints (inequalities and equations) that reflect the logical relations between all variables represented by the model.
- A part that corresponds to a current specification of values such as goals and trade–offs set by a user. This specification is interactively changed, often drastically, by the user. During an interactive procedure a DM specifies goals and preferences such as values of objectives that he/she wants to achieve and/or to avoid. Such a specification may result in the generation of additional constraints and variables, which are added to the *core model* forming the optimization problem.

In his other paper, Makowski [M94b] lists the advantages of the two–stage approach over the traditional approach in which both preferential structure of a user and logical and physical relations between variables are specified and implemented together. The advantages are outlined below:

- A properly defined core model should always have a feasible solution.
- A traditional model quite often has an unnecessarily narrow set of admissible solutions, which is caused by adding constraints aimed at making a solution not only feasible but also acceptable. Such additional constraints correspond to a preferential structure of a user and therefore should be implemented as *soft* constraints but in many applications they are implemented as *hard* constraints (i.e. in a way similar to the constraints representing logical and physical constraints). This in turn leaves out many interesting solutions beyond analysis (because such solutions are not considered to be feasible in the strict sense of mathematical programming).

- The generation of a core model is problem specific and is usually done by a problem specific problem generator. A verification of a core model can (and should) be done before starting an interactive analysis of a model.

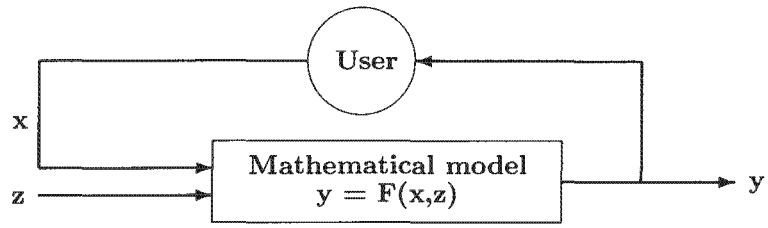
- Interactive analysis of the model is aimed at generation and analysis of rational solutions. Therefore the DM specifies interactively preferences, goals and/or additional constraints that narrow the set of acceptable solutions. In other words, the DM examines solutions that fulfill both constraints specified by the core model and additional requirements specified by the DM. The DM typically changes those requirements substantially upon analysis of previously obtained solutions. Contrary to the constraints specified by a core model (which can be interpreted as hard constraints that must not be violated) additional requirements are very often not attainable therefore they should not be represented as hard constraints. Hence, a properly designed interactive procedure should never generate an optimization problem that is infeasible.

- An interactive analysis of the model can be done with the help of modular tools that are not specific and can be used for a class of problems, e.g. Lindo's *What's Best!* can be used for any linear programming (LP) model. Hence, software development is easier because one can reuse whole modules. Moreover, different methodologies and corresponding software modules for interactive analysis can be used without changing a core model formulation.

- A number of additional constraints and variables generated during an interactive analysis of the model is typically a small fraction of a number of constraints and variables of a core model. Therefore handling the corresponding modifications are much easier from both logical and technical points of view. The latter includes using the last solution for a warm start of the next optimization run.

- There is no need to generate soft constraints in the core model because all the criteria can be treated in a uniform way. Generation of soft constraints is a sound idea but in practice the handling of a prior specification of soft constraints is cumbersome and therefore rarely used. However, one can easily handle soft constraints within multi-criteria model analysis (see Makowski [M94b], section 6 for details).

Makowski and Wierzbicki [MP03] give the concepts essential for explanation of model–based decision support as depicted in the figure below.



Decisions (inputs) **x** are controlled by the user. External decisions (inputs) **z** are not controlled by the user. From the mathematical programming point of view, the external decisions are represented as given fixed values of variables, and therefore are treated as parameters of the model. These are the two types of inputs to the core model. The outcomes (outputs) **y** are used for measuring the consequences of implementation of inputs. In practice, inputs **z** may include representations of various quantities that substantially influence the values of outcomes **y** although they are not controlled by the user [MP03]. The mathematical model F, which represents the relations between decisions **x** and **z**, can be symbolically represented by: $\mathbf{y} = \mathbf{F(x,z)}$.

3.3.1 Types of Model–based DSS

There is a general agreement [A90] that model–based DSS are basically of two types that correspond to the ways of a model analysis: *descriptive* and *prescriptive*. The *descriptive* DSS are used for prediction of the modelled system behavior without an attempt to influence it. The *prescriptive* DSS are aimed at providing information about controls (in managerial situations called decisions). In other words, a descriptive DSS helps to answer a question such as *"what will happen if"*, whereas a prescriptive DSS supports answers to questions like *"what decisions are potentially good"* [M94a]. For the prescriptive type of DSS, optimization techniques are widely considered as good tools for selecting a nondominated solution which is considered *potentially good*. The term *potentially good* corresponds to a solution that provides best value for the objective function

Makowski [M94a] further states that:

> it is desirable to use a model–based DSS in both (i.e descriptive and prescriptive) modes interchangeably. For example, before even trying to find prescriptions, one should verify the model also in the descriptive mode. The model should not only conform to the formal specification but also all discrepancies between the intuitive judgement of a DM and analytic results obtained from the model must be resolved.

3.4 Actors in a DSS

Several kinds of participants are usually involved during the development process of a DSS. Following Zaraté [P89], we have *initiators* or *implementors*[2], who had the initial idea to build a DSS and constitute the first party involved in the DSS development process. This may be an Operational Research/Management Science (OR/MS) analyst. The analysts analyze the model and conduct some studies above its future version. Walkenbach [J04] divides initiators/developers into two primary groups:

- *Insiders* are the developers who are intimately involved with the users and thoroughly understand their needs. In many cases, these developers are also users of the application. Often, they develop the DSS in response to a particular problem.
- *Outsiders* are the developers who are hired to produce a solution to a problem. In most cases, developers in this category are familiar with the business in general but not with specifics of the DSS they are developing. In other cases, these developers are employed by the company that requests the DSS (but they normally work in different department).

Developers ought to have a thorough understanding of their development environment. DSS developers are typically involved in the following activities:

- Determining the needs of the user
- Planning a DSS that meets these needs
- Determining the most appropriate user interface
- Testing the DSS under all reasonable sets of conditions
- Documenting the development effort
- Distributing the DSS to users
- Updating the DSS if and when it is necessary

Another key actor/participant in consideration is the *decision maker* (DM), also known as the *user*. This is a person who makes real decisions (depending on an application it may be a manager or an engineer or an operator) or an expert who may be his/her advisor. The user may have no previous knowledge of OR/MS tools [SCF04]. Decisions are made within a *Decision Making Process* (DMP), which, in situations that justify usage of a DSS, is rather complex sequence of tasks [M94a].

[2]Initiators or implementors are also known as developers

3.5 Characteristics of a DSS

Despite all the different definitions of DSS, there has been research [W81, S77a] indicating that systems which everyone agrees to be DSS share the same characteristics. Finlay [P95] sum up the characteristics that define DSS as:

- they tend to be aimed at the less well structured, underspecified problems that upper level management face;
- they attempt to combine the use of models or analytic techniques with traditional data access and retrieval functions;
- they specifically focus on features which make them easy to use by non–computer people in an interactive mode; and
- they emphasize flexibility and adaptability to accommodate changes in the environment and the decision making approach of the user.

Makowski [M94a] give characteristics of a DSS that implicitly define a class of model–based DSS as outlined below:

- A DSS is a supportive tool for the management and the processing of large amounts of information and logical relations that helps the DM to extend his habitual domain and thus help him/her to reach a better decision. In other words, a DSS can be considered as a tool that, under full control of a DM, performs the cumbersome task of data processing and provides relevant information that enables a DM to concentrate on this part of decision making process (DMP) that cannot be formalized.
- A DSS is a problem dedicated system designed for a specific DMP and its environment. The functioning of a DSS should be consistent with the actual environment of a DMP. A DSS is often tuned for a specific DM.
- A DSS is not a *black box type* tool. The structure and functioning of a DSS (including explicit and implicit consequences of assumptions adopted for its design) must be such that a DM understands and accepts them. The user interface of a DSS is designed in such a way that a DM may obtain, from the DSS, information and answers for questions that he considers to be important for a DMP.
- A DSS is not intended to solve a decision problem. Therefore it should not support reaching a single or unique decision nor should it restrict a possible range of decisions.
- A DSS should support a user during a DMP by providing two main functions. First, it allows for examination of consequences of any (feasible) decision. Second, it helps in finding decisions that are the best for attaining goals specified by a user.

3.6 Essential Requirements for a DSS

According to Holsapple et al.[WSB93], three primary categories of system requirements for the DSS can be established as: (1) *functional requirements*, (2) *interface requirements* and (3) *coordination requirements.*

3.6.1 Functional Requirements

This set of specifications for the DSS focuses on its capacity for the storage, recall and production of knowledge useful to the problem context. An example of this type of requirement might be the capability of a particular DSS to store a variety of product sales projections, recall specific components or characteristics of one or more projections, and estimate the effect on sales volume of one or more changes to the underlying assumptions or variables contained therein [M99].

3.6.2 Interface Requirements

The interfaces for preference acquisition and interactive decision making for MCDM methods have to be designed with their own features. Through these interfaces, the user is able to use an integrated multicriteria decision support system (ICM–DSS) to deal with his/her decision problem even though he/she may not know the details of the mathematics involved in the MCDM methods. Thus, the heart and soul of a DSS is arguably related to the quality and usefulness of its interface [M99]. The interface requirements focus on the communication capabilities of the DSS. During this phase, the designer must determine the various channels and methods of communication that will be made available to the DSS user and the conditions under which they will be made available. Menuing, report structures, command interface, and output formats are all identified during this phase of the development process. The developer must also identify the various types of requests that might be made of the DSS by the DM and appropriate types of responses that the DSS must be able to make as a result.

In other words, in designing the user interface (UI), screens as well as what will happen if error occurs should be considered. The UI should be clear and consistent. The design of the screen should be easy to understand and comfortable with the user. The best way to accomplish these goals is to follow industry standards in relation to color, size and placement of controls [CC01]. For example, most Windows applications are predominantly gray. One reason is that some people are color blind, and gray is considered the easiest and best color for users. Although a designer may personally prefer brighter colors, if he/she wants the applications to look professional, he/she should stick to gray [CC01]. The white background for text boxes should be used to indicate that the user is to input information. Gray background for labels which the user cannot change should be used. Controls that aid the user should be grouped together. It is astute to create frames to hold related items, especially those

controls that require user input. This visual aid helps the user to understand the information that is being presented or requested.

3.6.3 Coordination Requirements

In addition to establishing the specific functions and communication mechanisms associated with the proposed DSS, the designer must determine the coordination requirements as well. This include timing of events associated with DMP, the facilitation of access to relevant information, and the integration of various modelling tools contained within the DSS.

3.7 The Modelling Process

Modelling is a network of activities, often referred to as a *modelling cycle*, or a *modelling process* [MP03]. The modelling process starts with an analysis of the problem, including the description of objectives and questions to be answered. A conceptual (qualitative) version of a model is set up to support further discussions between the modeler and user. This is where the selection of types of variables and mathematical relations between them is made. The modeler then translates this conceptual model into a *model specification*. The model specification is composed of mathematical (symbolic) relations (i.e definitions of variables and relations between them). For non–trivial problem, model specification is an iterative process which involves a series of discussions between developers (typically OR/MS specialists) and users until a common understanding of the problem and its model representation is agreed [MP03].

The next phase of the modelling process is *model analysis*. A properly organized analysis of the model is the essence of any model–based problem support [MP03]. Properly organized means that the user is supported in using all relevant methods of analysis, comparing the results, documenting the modelling process, and also in moving back to the first stage, whenever he/she wants to change the type of the model. During the model analysis, different computational tasks are generated and solved by *solver*, which are software specialized for specific types of mathematical programming problems.

Makowski and Wierzbicki summarize the modelling process as outlined below [MP03]:

- problem formulation,
- model (symbolic) specification,
- collection, cleansing, and verification of data,
- model implementation, verification and validation,
- model analysis,
- model management, including documentation of the whole modelling process

The process of specifying the requirements to be met by the modelling process or establishing the specifications that the modelling process must fulfill is called metamodelling [M94a].

3.8 The Systems Development Life Cycle

A common perspective with regard to system design and development is found in the *system development life cycle* (SDLC) approach. The SDLC portrays the process of a series of recursive phases, each with its own set of required inputs, activities, and outputs [M99]. The stages of SDLC are outlined below:

1. *Problem Definition:* In this phase, analysts assess the nature of the organization's problem and develop documentation describing the problem context and the organizational environment.

2. *Feasibility Survey:* Feasibility analysis is performed to assess whether the problem context outlined in stage 1 can be effectively addressed by one or more applications integrated into the information systems. Activities during this stage include conversion of the existing physical system into a series of logical models to look at the embedded processes in an implementation–independent manner. Initial determinants of feasibility with regard to available technologies, human and economic, and political and regulatory constraints are conducted during this phase. The output will be a feasibility report.

3. *Systems Analysis:* It focuses on the investigation of the situation to determine what needs to be done. This includes specifying to management what the new system will do for them, what they and their stuff will be required to do, and what the system will look like. The output is a system specification.

4. *Systems Design:* The phase involves the creation of a detailed model. This will define and identify all data items, all file structures and the programs and program structure. The systems software that will be needed is also specified in this phase. Validation procedures are also defined.

5. *Systems Programming:* In this phase, the logical model for the new system is converted into a physical system through the development and acquisition of software and hardware necessary to meet the needs of the system's user(s).

6. *Systems Testing:* This is where the system is put through a series of trial runs to ascertain its performance relative to the stated requirements. Not only must the correctness of the product be tested, but also its robustness: Intentionally, erroneous input data are submitted to see whether the product will crash, or whether its error handling capabilities are adequate for dealing with bad data.

7. *Systems Documentation:* This involves writing down the system works, the names and definition of variables, collating program listings, job descriptions for personnel involved in the maintenance of the system.

8. *Systems Implementation:* It involves the final installation activities of the system.

9. *Systems Review:* It involves the evaluation of the stages of the system development life cycle and of the system itself.

10. *Systems Maintenance:* It involves various refinements and incremental enhancements to the operational system to effect smooth performance and functionality to the system. Maintenance is not an activity that is grudgingly carried out after the product has gone into operations mode. It is an integral part of the life cycle that must be planned from the beginning.

An alternative to the SDLC approach is ROMC analysis [JC82]. ROMC approach focuses the analysis effort on developing the understandings of *Representations, Operations, Memory aids* and *Controls.* A brief explanation of ROMC analysis for a DSS design is given below.

The analyst characterizes the various *representations* available for use as methods of communication between the DSS, the user and the DSS application. This include graphical displays, charts, tables, lists, stocks, input forms, menus etc.

In the *operations* analysis, the analyst is focused on identifying those activities necessary for the DSS to perform or facilitate the generation and delivery of the various representations contained within the system. This includes the interpretation, production, and packaging of relevant knowledge contained in the DSS.

Memory aids are the components intended to provide support to the use of the various identified representations and operations. The memory aids include databases, workspaces or blackboard systems, and embedded triggers intended to alert DSS users of the need to perform specific operations relevant to the task at hand.

Controls are mechanisms that help the DSS to extract or synthesize a particular decision–making process from a number of available representations, operations and memory aids. Controls include mechanisms to assist the user in submitting specific requests or queries to the DSS, functions that assist the user in tailoring the DSS to his/her unique problem–solving style or tendencies, and modules that assist the user learning to use specific elements within the DSS through examples rather than trial–and–error discovery.

3.8.1 Analysis and Design of a DSS

DSS analysis should be thought of as business problems solving, independent of technology. It is driven by system users' concerns. The analysis phase usually begins with one or two interviews with the project requester (user). The interviews include the facts such as list of problems and opportunities that led to the project request, and any relevant constraints, ideas and opinions. Based on these facts from the initial interviews, the systems analyst should attempt to define the initial scope of the proposed project. If the analysis of the interviews' facts recommends the development of a new DSS, the systems analyst (or project manager) needs to develop a project plan. That is, a first–draft master plan and schedule for completing the entire project should be developed. This schedule will be modified at the end of each phase of the project.

DSS design may be seen as the evaluation of alternative solutions and specification of a detailed computer–based solution [LDM94]. Whereas DSS analysis focused on the logical–independent aspects of a DSS (the requirements), DSS design deals with the physical or implementation–dependent aspects of a DSS (the system's technical specifications). DSS design builds on the knowledge derived from systems planning and systems analysis.

Many analysts are turning to *prototyping*, a modern engineering–based approach to design. According to Schach [R90] a *prototype* is a working model that is functionally equivalent to a subset of the product. The purpose of a *prototype* is to provide a user with an understanding of the software. It enables the user and the developer to agree as quickly as possible on what the software is to do. Any imperfections in the prototype may be ignored, provided that they do not seriously impair the functionality of the prototype and hence give a misleading impression of how the product will behave. Prototyping approach has several advantages as outlined in [LDM94]:

- Prototyping encourages and requires active user participation. This increases the user's morale and support for the project. User's morale is enhanced because the system appears real to him/her.
- Iteration and change are a natural consequences of systems development – that is, users tend to change their minds. Prototyping better fits this natural situation since it assumes that a prototype evolves, through iteration, into the required system.
- It has often been said that users do not fully know their requirements until they see them implemented. If so, prototyping endorses this philosophy.
- Prototypes are an active, not passive, model that users can see, touch, feel, and experience.
- Prototyping can increase creativity because it allows for quicker user feedback which can lead to better solutions.

- Prototyping accelerates several phases of the SDLC. In fact, it consolidates parts of the phases that normally occur one after another. These phases include the following:
  - *Definition.* Prototyping can be used to quickly experiment with different requirements. Each prototype can change not only the design, but the actual requirements– until the users accept the requirements. Requirements can be defined more quickly with this approach.
  - *Design.* Screen and report layouts can be very quickly changed until users accept their design. Terminal dialogue can be user–tested for friendliness and completeness. In most cases the design, as development through prototyping, can be completed faster than one developed with paper and pencil. The working prototype has been seen by users; therefore, it is less likely to be redesigned after it has been implemented in final form.
  - *Construction.* The very act of prototyping requires construction, which is also known as programming. Many prototypes are being implemented in the prototyping languages. This can significantly reduce implementation time and effort.

There are also disadvantages to using prototyping approach. Whitten et al [LDM94] sum up most of these disadvantages in one statement:

> prototyping encourages ill–advised shortcuts through the SDLC.

From many different approaches of DSS development, two common strategies can be identified as:

1. programming a customized DSS and
2. employing a DSS generator

Each approach is unique and the choice of which one to adopt is often contingent on the organizational setting or problem context. It is quite possible for a complex DSS undertaking to use both approaches in combination during the various design stages of the project [M99]. Programming a customized DSS employs either a general purpose programming language (GPL) such as PASCAL or COBOL or a fourth–generation language (4GL) such as Visual C++ or Delphi. 4GL are non–procedural languages that are often considered to be more productive than traditional (procedural) programming languages [M94a].

A *DSS generator* is an application system that eliminates the need for programming thousands of individual instructions or code in the design and construction of a DSS [M99]. Most common forms of DSS generators are the electronic spreadsheets such as Excel, Lotus 1–2–3 and Quattro Pro. Several commercial add–ins such as Lindo's *What's Best!*, are available to assist in the development of various types of DSS analysis mechanisms using electronic spreadsheet as the primary DSS generator.

3.8.2 The Quality of Software

Software related problems are key components of any decision related to design and implementation of any DSS. There is probably not one bug–free software package, therefore it is important to use software engineering techniques that ease the software development and result in better end products like modular structure of each DSS component and the application of object oriented programming languages, as well as utilization of well–tested software tools [M94a]. Makowski further states that:

> Software development and maintenance is a major cost component of probably any DSS. Direct costs of development of any dedicated software are higher than those of purchasing a *ready from the shelf* package whereas the relation of their functionality are the opposite. Direct costs of software development may be remarkably reduced by two factors: first, using appropriate software tools (both as parts of DSS and for software development), second, by reusability of the developed software. There are two key principles that allow the re–use of software. First, the software should have a *modular structure* and should be well tested. This implies that the popular *"quick and dirty"* programming techniques should never be used in DSS development (no matter how far behind schedule the project is). Second, the software should be developed in such a way that it is *portable* between many possible architectures.

The above arguments can be rounded up with the general advice that one should select the programming languages and tools that allow for fast development of software that is *robust (reliable), reusable, portable, allowing for fast prototypying, allowing for modifications, testing and can be used easily.* Software tools are generally of three types: programming languages, general purpose tools (like utilities for development of the user interface or for handling typical data structures) and tools for a given category of problems (either mathematical programming problems or substantial problems). Tools that allow for fast development of software are briefly discussed below.

Software Robustness (Reliability)

Software robustness is understood as its ability for functioning according to the specification and to the needs of the user. This includes proper behavior for possibly any data supplied for the program, any action of the user and for typical hardware problems. The software robustness is also the necessary condition for the software reuse, and justifies the careful design and testing of all software components [M94a]. Marakas [M99], further points out that for software robustness, there should be a degree of: (1) *completeness*: that is, all of the software components are present and each is fully developed, (2) *accuracy*: that is, software outputs are sufficiently precise to satisfy their intended use and

(3) *consistency*: that is, software contains uniform notation, terminology, and symbology within itself, and its content is traceable back to the requirements.

Software Reusability

A large part of the software should be reusable both in different applications and by different teams without the necessity of any modification of the available software. This requires some additional programming effort, but it is enough to consider how much time is necessary to debug and test new software to justify this effort [M91]. The reusable software should be carefully designed, tested and converted into libraries that are well documented. It must contain enough information for the user to determine its objectives, assumptions, constraints, inputs, outputs, components and status. The reuse of some specific software (e.g solvers) requires an additional agreement on the common structure of data for a specific class of mathematical programming problems.

Software Efficiency

The software should have the degree of device efficiency: that is, the software should fulfill its purpose without any waste of resources. It should also have a degree of accessibility: that is, the design of the software should facilitate the selective use of its components. A choice of an algorithm for solving a problem is critical for software efficiency.

Software Portability

Software should be device independent. It should be executed on a variety of hardware configurations. Nowadays, there exist many portable tools (especially for Graphical User Interface (GUI)) for software development. This is mainly due to today's' availability of several more operating systems than in the past. Therefore, portability is now essentially a question of a careful design of the software and of a proper selection of software tools used for implementation of a DSS.

Software Modifiability

Well–designed and implemented software should be easily modifiable for specific purposes. This includes modification of the user interface for a specific decision–making problem or translation to another spoken language or extension for supplying additional options. It should as well easily accommodate expansions in data storage requirements and component computational functions.

3.8.3 Validation of a DSS

Validation of a DSS is defined by Finlay [P95] as "the process of testing the agreements between behavior of the DSS and that of the real world system

being modelled". It is the process of defining whether the model behavior represents the real world system in a particular problem domain. Finlay points out that DSS validation is not concerned with proving that a DSS is a truthful representation of the real world– since this is impossible– but with demonstrating that the DSS has appropriate underlying relationships to permit an acceptable representation. Very few model–based DSS have been properly validated [JMEE90].

Validation has two dimensions– *verification* and *substantiation* [JMEE90]. Verification is defined as "the process of testing the extent to which the model has been faithful to its conception, whether or not its conception are valid" [JS88]. Substantiation is defined as "the demonstration that a computer model, within its domain of applicability, possesses a satisfactory range of accuracy consistent with the intended application of the model" [OA81].

The validation process is explicitly incorporated into the development life cycle of the DSS prototype under constrained resources– time and costs. A DSS is evaluated through a two–stage procedure– laboratory testing (that ensures face validity, subsystems 'verification and validation' and predictive validation) and field testing [JMEE90]. The two–stage validation occurs iteratively throughout the system development. The results from any stage (or substage) may require changes (reformulations, redesign, and refinements) in the prototype. Whenever the prototype is modified or expanded, the system must be re–evaluated.

As mentioned above, there are two clear stages involved in the validation [D98] approach:

1. An internal stage within the *laboratory tests*. The laboratory experiments take place in settings constructed by the development team for evaluating the DSS. Some of these tests may involve potential users that will contribute to the validation through questionnaires and interviews. In laboratory testing, *predictive validation* should only be executed after *face validation* and *subsystems 'verification and validation'* have been successfully carried out. The main objective of face validation is to achieve consistency between the designer's view and the potential user's view of the problem in a timely and cost–effective way. Face validation also ensures that the formulated problem contains the entire actual problem and is sufficiently well structured that a credible solution can be derived before extensive and detailed software development proceeds [JMEE90].

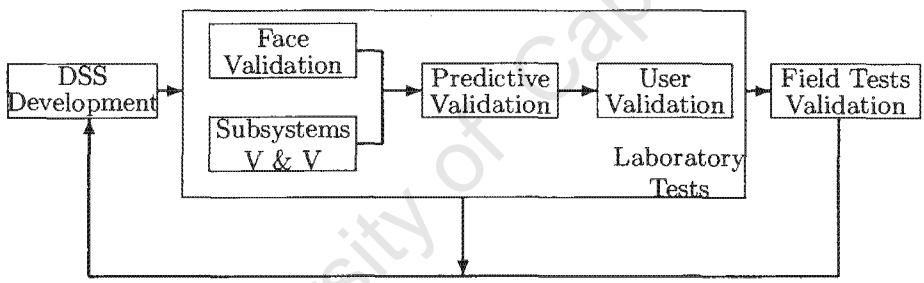
   Subsystem 'Verification and Validation' (V & V) consists of testing, verifying and/or validating the DSS modules one at a time as they are developed. The main objective is to guarantee the quality of each model in the sub–model component of a DSS. Predictive validation consists of validating systems using laboratory test cases in which the results are known. A DSS is driven by past data from the test cases, and its results are com-

pared with corresponding known results.

User assessment is carried out at last, after a successful predictive validation. User assessment can be defined as 'the process by which interested parties (who are not involved in a model's origins, development, and implementation) can determine, with some level of confidence, whether or not the model's results can be used in decision–making [I83]. The main objectives of the user assessment testing are: (a) to obtain a statement of the applicability of the system by possible users; (b) to assess the impact of the computational system's assumptions, simplifications, methods and generic structure from an independent source.

2. Only when the DSS has been internally verified by laboratory experiments should the external stage of extensive *field tests* be carried out. The field tests place the DSS in the field, and seek to identify those performance errors that occur [MOP87]. A field testing validation of a computer based system is a very desirable step to take before a full implementation [JS88].

The DSS life–cycle with emphasis on the validation process is given in the figure below.



3.8.4 Implementation of a DSS

There is no general specification of sufficient conditions for the implementation of a DSS since this obviously depends on a particular environment of a DMP [M94a]. The implementation of a DSS can be seen as a set of activities which are focused on the successful introduction of a DSS into the organizational environment [M99]. It is basically an introduction to a change in an organizational environment. Schein [H56] gives a descriptive framework (not intended to be prescriptive) to develop an understanding of the change process as it occurs during a typical DSS implementation. According to Lewin–Schein theory of change [H56], the process of change occurs in three basic steps:

1. ***Unfreezing.*** The creation of an awareness of the need for change and a climate receptive to change.

2. ***Moving.*** The changing of magnitude and/or direction of the forces defining the initial need for change through the development of new methods and the learning of new attitudes and behaviors.
3. ***Refreezing.*** Reinforcement of the desirable changes that have occurred and the establishment of a maintainable and stable new equilibrium.

A successful implementation of a DSS can occur only when the DSS is fully validated. To achieve a successful implementation, it is not just sufficient for upper management to accept the model; the people who will run it every day must be thoroughly trained in its use. At the very least, they should understand how to enter appropriate inputs, run the analyses, and interpret the model's outputs correctly.

Once the organization sees the benefits of a useful model (and of MCDM techniques in general), it is likely to expand the model (or create new models) for uses beyond those originally intended [LC97]. Thus, the software qualities mentioned in the previous section ought to be implemented for the expansion of the implemented model.

The general belief is that software is "never" ready on time [M94a]. Makowski further states that very often the testing phase of software development is under–evaluated despite the commonly known fact that there is probably never enough software testing. Therefore quite often either the software is not ready on time or that an inadequately tested version of the software is released.

3.8.5 DSS Maintenance

Once the DSS has passed the acceptance test, it is handed over to the users. Any changes after the client has accepted the product constitute *maintenance.* In maintenance, as in new systems development, it is essential to go through a series of steps that involve understanding the problem, analyzing needs and opportunities, then devising solutions and designing them technically before going ahead with implementation. If maintenance is treated on a "quick–and–dirty" basis, there is a great danger that the systems being maintained will, in some way, be destroyed by those who want to enhance them. Therefore, it follows that all of the skills acquired in connection with the study of systems development are equally appropriate in maintenance projects [JHG90]. There are three main reasons for making changes to a product after it has been delivered to the client [R90].

- The first reason is to correct any residual faults; specification faults, design faults, coding faults, documentation faults, or any other types of faults. This is called the *corrective maintenance.*

- The second type of maintenance is known as *perfective maintenance*. In this type of maintenance, changes are made to the code because the client thinks that these changes will improve the effectiveness of the product. The client may, for example, wish to have additional functionality added or might request that the product be modified so that it runs faster. Users become progressively more sophisticated as systems are developed and used. The more experienced they become, the more opportunities they are apt to discover for enhancing the systems they have.

- The third type of software maintenance is known as *adaptive maintenance*. Here changes are made to the software in order to react to changes in the environment in which the software operates. The software will almost certainly have to be modified if, for example, it has to be ported to a new compiler, operating system, and/or hardware. Computers are at the hub of one of the most dynamic fields of endeavour in the world [JHG90]. Whole new generations of concepts and equipment are brought into use every now and then.

Maintenance has become an increasingly important segment of the activities of systems analysts, systems designers, and programmers [JHG90]. Powers et al.[JHG90] further state that maintenance projects can be greater factors than systems development projects in terms of working days of systems and programming time consumed. Chaudhry et al.[SLM96] emphasize that the design of the system, choice of development tool, and programming skills of the developer, can also dramatically affect the maintenance costs.

3.9 Selection of MCDM Technique for a DSS Engine

Gershon and Duckstein [ML83, ML84], and Ozernoy [W92] suggest that the choice of MCDM technique for a specific task is actually a multi-criteria problem itself. Several MCDM methods have been developed, and their number continues to grow. Different methods often represent completely different approaches to decision making, and choosing among methods may depend on the particular problem under consideration [W92]. Belton and Stewart [VJ02] further state that the implementation of MCDM covers a range of different ways of working, having different implications for the nature of support required in terms of facilitation and analysis, as well as attendant facilities and technology. Belton and Stewart discuss the issues that are important when seeking to use in practice any of MCDM approaches. The issues addressed span a broad range of considerations, such as conceptual foundations of modelling, skills of facilitation, etc. (see Belton and Stewart [VJ02], chapter 9 for more detail). Consideration of matters discussed in their book is essential in ensuring effective implementation of MCDM approaches.

Sen and Yang [PJ98] provide for general questions asked for the selection of an appropriate MCDM method that can be used in a DSS. The questions are as outlined as follows:

1. How is the preference information elicited?
   A priori articulation or *Progressive articulation*

2. What type of preference information is available?
   Ideal design, goal values, implicit trade–off or *explicit trade–off*

3. Which decision rule is opted?

3.10 Concluding Remarks

This chapter provides a vehicle for creating a structured, yet stimulating, environment in which to learn the fundamental concepts of decision support systems' development. Following all the concepts of DSS development in this chapter, and an overview of interactive MCDM techniques in the previous chapters, we are now in the position to move forward to the practical development of an integrated multiple criteria decision support system.

Chapter 4

Development of an Integrated Multiple Criteria Decision Support System

In this chapter we discuss issues related to the development of an integrated multicriteria decision support system(IMC–DSS) using the framework presented in chapter 3. The aim of developing an integrated multicriteria decision support system is to support and facilitate the process of constructing a good decision, when such decision involves multiple and conflicting goals. A number of tools have been used to facilitate improvement in decision making. These tools include spreadsheets, optimization tools for OR/MS (such as LINDO) and MCDM methods. Although these tools have played an important role in solving decision problems, each tool has its own limitations and could not be used alone to reach the best (compromise) solution. This poses the challenge of integrating these decision tools for better decision solutions.

4.1 A Framework for Development of a DSS

A generalized approach to the design, development and implementation of a DSS is proposed in this section, and is based loosely on the systems development life cycle (SDLC) concept and prototyping. DSS designers have long recognized the need for departure from the traditional SDLC and recommend an iterative approach, with extensive use of prototyping [SLM96]. In real life development of a DSS, the framework requires more active user[1] participation in all stages. A fundamental assumption of the traditional SDLC methodology is that the requirements can be completely specified during the analysis phase. This does not necessarily translate well to DSS development since the user may not fully understand or be able to articulate requirements early in the life cycle. The

[1] In our case the user is conjectured during the development process of a DSS

process of requirements specification for a DSS is best characterized as a learning experience which takes place continually during the development process.

4.2 DSS Design

The proposed DSS is model–based. A model-based DSS often uses multi–criteria optimization for selecting nondominated solutions. Such a selection is based on the interactive specification of user preferences. The DSS is basically a spreadsheet application. That is, linear programming problems are solved using Microsoft Excel spreadsheet. The problems are solved using a special purpose optimization package called What'sBest![2] (Lindo Systems, 2003) which is capable of solving linear optimization problems. The detailed design focus on the design of the database, the model base, and the user interface. The model is designed in such a way that it provides the user with the ability to integrate mathematical (MCDM) models (model base) with available data with a click of a button. The user interface is designed in such a way that it is clear for the user to use the DSS, and the format of the results facilitate decision–making, and are consistent with the user's mental model of the system. To integrate the database, What'sBest and MCDM models the DSS is coded in Visual Basic for Applications (VBA). The primary reason for using VBA is to hide the integration from the user (DM) who is not interested in the technical background of DSS.

4.3 DSS Overview

4.3.1 Getting Started

The DSS can be run from the compact disk (provided *What'sBest* is installed). It is advisable that the DSS files be copied to the hard drive for better performance. However, before running the DSS there should be some settings made to the spreadsheet (Microsoft Excel). Firstly macro security should be changed. This is done by checking *Medium* under *Tools|Macro|Security Level.* If the security level higher than *Medium* is selected then the macros would not run. Secondly *What'sBest* reference should be activated. This is done by checking *wba.xls* or just *wba* under *Tools|References* in the Visual Basic Editor window. Visual Basic Editor window can be accessed under *Tools|Macro|Visual Basic Editor* from the Excel window. If the reference to *What'sBest* is not made, *What'sBest* attributes or procedures (i.e. calling *What'sBest* from the VBA code) will produce error.

After all the settings are completed then the DSS can be opened. Every time the DSS is opened a pop-up for 'security warning' appears. Then the macros should be enabled by clicking on "Enable Macros"

[2]What'sBest! is basically an Excel add-in

4.3.2 Problem Formulation

Before the problem is solved the model needs to be formulated. The formulation menu can be accessed by clicking "Formulation" on the main menu. The user starts by giving the number of objectives, number of decision variables and the number of constraints in the model. The user is guided throughout the process of formulation. In some cases, the process is hierarchically structured. For example, the number of constraints cannot be given before the number of objectives is given. If that hierarchy is not followed the user is warned or guided as to find the direction. The model formulation is independent of the three methods implemented. That is, there is no need to reformulate the problem if the user wants to change the method he/she is using. The formulated model can be deleted and yet another model be formulated. The DSS gives the user the choice of deleting the entire model without saving or saving the model before deleting it. For details of problem formulation see appendix (A).

4.3.3 Method Selection

The DSS provides for three MCDM methods selection for a decision problem. All the user has to do is select the method to use in the "Main Menu". The MCDM method selection depends on the individual DM who may choose to use one or more of the three for different insights. The selection of which MCDM method to use is itself a multi–criteria problem. No method is inferior to the other, and it depends on how comfortable the DM feels in using each of the three methods.

The method to use can be selected under different criteria such as ease of use and confidence in the solution obtained. Elicitation of preferences change from one interactive MCDM procedure to another implying differences in the difficulty of the information demanded at one iteration and the number of iterations [YTE96].

4.4 MCDM Model Generation

The three alternative MCDM models which have been selected for implementation in a prototype DSS for MOLP problems are:

- Wierzbicki's [A.P80, P99] reference point method
- An interactive weighted Tchebycheff method [RE83]
- An interactive value function method [VJ02]

The main reason for selecting these three methods is that they are all somewhat different in nature, although they are all interactive.

The reference point method makes use of an achievement scalarizing function

and requests the DM to specify his/her reference point (aspiration levels) before each solution procedure. The weighted Tchebycheff procedure is a weight space method in which the DM selects the solution that he/she thinks is the best among the generated solutions, then the procedure is updated in the vicinity of the selected solution to contract the weight space. The value function method also requires the DM to select between alternative solutions, but the information is now used to provide estimates of the underlying value function. The next subsections (4.4.1, 4.4.2, 4.4.3) include technical details of proposed models implementation, assuming that problem formulation has been done.

4.4.1 Reference Point Model

The reference point model for a DSS is constructed in a similar way to the traditional (original) model (see section 2.3.2). There are no modifications made to the model. The steps are coded as outlined below:

Step 1:

Find the ideal criterion vector $\mathbf{z}^* = (z_1^*, z_2^*, ..., z_k^*)$ by maximizing each of k criteria individually.

Step 2:

Let $r = 1$. Let the DM express his/her reference point (aspiration levels) $q_i^{(r)}$ in relation to z_i^* , such that $q_i^{(r)} < z_i^*$ $(i = 1, 2, \ldots, k)$, $\mathbf{q^{(r)}} \in \mathbb{R}^k$. The vector $\mathbf{q^{(r)}}$ represents the reference point (aspiration levels for each criterion i.e. the levels of DM's achievement for each criterion).

Step 3:

Using $\mathbf{q^{(r)}}$ and $\mathbf{z}^*$ vectors, calculate λ–vector (representing the weights) corresponding to $\mathbf{q^{(r)}}$ as follows:

$$\lambda_i^{(r)} = \frac{1}{|z_i^* - q_i^{(r)}|} \left[\sum_{i=1}^{k} \frac{1}{|z_i^* - q_i^{(r)}|} \right]^{-1}, \; i = 1, 2, ..., k \qquad (4.1)$$

Note that λ_i's should sum to one $(i = 1, 2, ..., k)$ i.e. $\sum_{i=1}^{k} \lambda_i = 1$

Step 4:

Find $\mathbf{z^{(r)}}$ by solving the following achievement scalarizing program:

$$\text{Min} \qquad \alpha + \varepsilon \sum_{i=1}^{k} \lambda_i^{(r)} (q_i^{(r)} - z_i), \; \varepsilon > 0$$

$$\text{Subject to :} \quad \alpha \geq \lambda_i^{(r)} (q_i^{(r)} - z_i), \; i = 1, 2, ..., k \qquad (4.2)$$

$\mathbf{z^{(r)}}$ is essentially the projection of $\mathbf{q^{(r)}}$ onto the nondominated set of alternatives.

If the DM is satisfied with the solution $\mathbf{z}^{(\mathbf{r})}$ obtained, the process stops, otherwise go to step 5.

Step 5:

Let the DM update his/her aspirations and form yet another aspiration vector. That is, let $r = r + 1$ and the process goes back to step 3

Comments

At each iteration of the reference point method implemented in a DSS, the user is only requested to provide for his/her reference point (aspiration levels). He/she is not involved in any other steps outlined above.

4.4.2 Interactive Weighted Tchebycheff Model

Some modifications have been made to the interactive weighted Tchebycheff model described in section (2.3.4). The major problem with the traditional interactive weighted Tchebycheff procedure [RE83] is the filtering of weights, which makes it time consuming to reach the final solution. The weights in this procedure are not uniformly generated over the (weight) space.

Uniform weights can however be generated directly according to the following procedure:

- generate uniformly distributed weights $\lambda_i^{(U)}$ on (0, 1) $(i = 1, 2, ..., k)$,
- transform uniformly distributed weights $\lambda_i^{(U)}$ to be exponentially distributed by taking the negative logarithm of each weight $\lambda_i^{(U)}$, i.e $\lambda_i^{(E)} =$ -ln($\lambda_i^{(U)}$), where $\lambda_i^{(E)}$ are the exponentially distributed weights. This results in exponentially distributed weights $\lambda_i^{(E)}$ with mean 1,
- normalize the exponentially distributed weights, i.e. $\sum_{i=1}^{k} \lambda_i^{(E)} = 1$

With uniformly distributed weights, filtering is unnecessary, and it is only necessary to progressively screen weights as new information is received. This method reduces the time taken through the solution process (since there is only one set of weight vectors generated, and no filtering is needed).

The steps for the modified interactive weighted Tchebycheff model implemented in the DSS are outlined as in below:

Step 1:

Let $r = 1$. Determine the ideal point $\mathbf{z}^*$. Let $\mathbf{z}^{**} = \mathbf{z}^* + \varepsilon$ ($\mathbf{z}^{**} \in \mathbb{R}^k$ is called the *utopian criterion vector*, it is the smallest value greater than $\mathbf{z}^*$), where ε is a vector of arbitrarily small positive values; k is the number

of objectives. The *utopian vector* is defined to be an infeasible criterion vector that strictly dominates every nondominated (Pareto optimal) solution.

Let

$$\Lambda^{(r)} = \{\lambda \in \mathbb{R}^k : \ \lambda_i \in [0,1], \ \sum_{i=1}^{k} \lambda_i = 1\}$$

be the initial set of weighting vectors (i.e $\lambda_i \in [l_i^{(r)}, u_i^{(r)}] = [0,1]$).

We suppose that the number of iterations will not exceed $t = 10$. If we aim to have weights within an interval width h where Steuer [E86] suggests

$$\frac{1}{2k} \lessapprox h \lessapprox \frac{3}{2k}$$

then at each iteration the interval needs to be reduced by the factor g, where

$$(1/k)^{1/k} \lessapprox g \lessapprox h^{1/(t-1)}$$

Step 2:

Randomly generate $2500k$ (where k is the number of objectives) weight vectors with uniformly distributed weights over $\Lambda^{(1)}$ as described above.

Store (save) the weight vectors for future use.

Step 3:

Randomly select any five weight vectors from those generated in step 2. Mark the weight vectors so that they cannot be used again in the next iterations. This guarantees that different solutions are generated at each iteration during the solution process. Solve 5 solutions using the selected weight vectors. That is, determine 5 solutions using the augmented weighted Tchebycheff program as follows:

$$\begin{aligned} \text{Min} \quad & \alpha + \rho \sum_{i=1}^{k} (z_i^{**} - z_i) \\ \text{Subject to:} \quad & \alpha \geq \lambda_i^{(r)} (z_i^{**} - z_i), \ i = 1, \ 2, \ ..., \ k \\ & \mathbf{x} \in \mathbf{S} \\ & \mathbf{z} \in \mathbf{Z} \\ & \alpha \geq 0 \end{aligned} \tag{4.3}$$

where $\alpha \in \mathbb{R}$ represents the maximum weighted deviation of an objective from the utopian point i.e. $\alpha = \max_i \{\lambda_i (z_i^{**} - z_i)\}$ and ρ is a sufficiently small positive value.

Step 4:

Present the 5 compromise solutions to the DM. Ask the DM to select the solution that he/she most prefers. Let $\mathbf{z^{(r)}}$ be the selected solution. If the DM is satisfied with $\mathbf{z^{(r)}}$ then STOP, with $\mathbf{z^{(r)}}$ as the solution to the problem. Else go to step 5.

Step 5:

Let $\lambda^{(r)}$ be the weighting vector which generated $\mathbf{z^{(r)}}$ in step 4, with its components given as:

$$\lambda_i^{(r)} = \begin{cases} \frac{1}{|z_i^{**} - z_i^{(r)}|} \left[\sum_{i=1}^{k} \frac{1}{z_i^{**} - z_i^{(r)}} \right]^{-1} & , \text{ if } z_i^{(r)} \neq z_i^{**},\ i = 1,\ 2,\ ...,\ k \\ 1 & , \text{ if } z_i^{(r)} = z_i^{**} \\ 0 & , \text{ Otherwise} \end{cases}$$

Determine the reduced set of weighting vectors:

$$\Lambda^{(r+1)} = \{\lambda \in \mathbb{R}^k :\ \lambda_i^{(r+1)} \in [l_i^{(r+1)}, u_i^{(r+1)}],\ \sum_{i=1}^{k} \lambda_i = 1\}$$

where

$$[l_i^{(r+1)}, u_i^{(r+1)}] = \begin{cases} [0,\ g^r] & , \text{ if } \lambda_i^{(r)} \leq \frac{g^r}{2} \\ [1 - g^r,\ 1] & , \text{ if } \lambda_i^{(r)} \geq 1 - \frac{g^r}{2} \\ [\lambda_i^{(r)} - \frac{g^r}{2},\ \lambda_i^{(r)} + \frac{g^r}{2}] & , \text{ Otherwise} \end{cases}$$

in which g $(0 \leq g \leq 1)$ is a prespecified *convergence/reduction factor* raised to the r^{th} power.

Let $r = r + 1$. Eliminate all the weight vectors that do fall in the reduced interval presented above and return to step 3.

Comments

No specific number of iterations is specified (in the model implemented in the DSS), allowing the DM to stop when he/she feels that he/she is satisfied with the solution provided. Steuer and Choo [RE83] do not specify any specific value for the reduction factor, but they suggest that the reduction factor g be within the interval $(1/R)^{1/k} \lessapprox g \lessapprox h^{1/(t-1)}$, where $R \approx k$, k is the number of objectives and t the number of iterations. Steuer [E86] suggests the final interval width h to be $\frac{1}{2k} \lessapprox h \lessapprox \frac{3}{2k}$. The interval width is arbitrarily generated within the above specified boundaries. Similarly in the model used in the DSS, the reduction factor g is arbitrarily generated within the above specified boundaries. Once the reduction factor and the interval width are generated, they are stored throughout the whole solution process, unless the problem is reformulated.

At each iteration the user is provided with five solutions. The user is only requested to choose the solution which he/she most prefers, and the process continues. The user is not involved in any other technicalities outlined above.

4.4.3 Value Function Model

The basis of this model is that there exists a value function V(**z**) such that if the DM (or the user of the DSS) is able to express a definite preference for a solution $\mathbf{z}^a$ over another solution $\mathbf{z}^b$, then $V(\mathbf{z}^a) > V(\mathbf{z}^b)$. The value function model implemented in the DSS uses a piecewise linear approximation.

For any objective i, let $\underline{z}_i$ and z_i^* be the nadir and ideal respectively. The value function is approximated by p (usually equal to 4) piecewise linear segments. The piecewise linear value function procedure begins by splitting each objective range (between the nadir and the ideal) into p intervals defined by the breakpoints $z_i^0, z_i^1, ..., z_i^p$ (such that $z_i^0 < z_i^1 < ... < z_i^p$) where z_i^0 and z_i^p are the endpoints (the nadir and ideal respectively). Each objective function is split by introducing p new variables z_{id} (i=1, 2, ..., k; d=1, 2, ...,p) such that

$$z_i = \underline{z}_i + \sum_{d=1}^{p} z_{id}, i = 1,\ 2,\ ...,\ k \tag{4.4}$$

where z_{id} represent the value increment on the d^{th} interval, constrained by

$$0 \leq z_{id} \leq \frac{z_i^* - \underline{z}_i}{p} = L_i$$

and $z_{id} \geq 0$ only if $z_{i,d-1} = L_i$, $d = 2,\ 3,\ ..., p$

Define

$$\delta_{id} = \frac{z_{id}}{L_i},\ 0 \leq \delta_{id} \leq 1$$

which is then such that $\delta_{id} \geq 0$ only if $\delta_{i,d-1} = 1$.

Equation (4.4) can be written in terms of δ_{id}'s as follows[3]

$$z_i = \underline{z}_i + L_i \sum_{d=1}^{p} \delta_{id}, i = 1,\ 2,\ ...,\ k$$

We shall now assume that $V(\mathbf{z})$ can be expressed in the additive form as

$$V(\mathbf{z}) = \sum_{i=1}^{k} v_i(z_i)$$

[3]Note that any z_i implies the δ_{id} values and vice versa.

where $v_i(z_i)$ is the partial value function. We scale the partial value function such that $v_i(\underline{z}_i) = 0$ and $v_i(z_i^*) = 100\lambda_i$, where λ_i is the weight on objective i (normalized such that $\sum_{i=1}^{k} \lambda_i = 1$).

Then the piecewise linear approximation is given by

$$v_i(z_i) = \sum_{d=1}^{p} U_{id}\delta_{id}, \ i = 1, \ 2, \ ..., \ k$$

where U_{id} are the marginal value increments over each interval for z_i. The overall value function becomes

$$V(\mathbf{z}) = \sum_{i=1}^{k}\sum_{d=1}^{p} U_{id}\delta_{id}$$

which is a linear function in z_i

We still need to ensure that $\delta_{id} > 0$ only if $\delta_{i,d-1} = 1$. However if the value function is concave such that $U_{id} < U_{i,d-1}$ for each d, then the maximization of V($\mathbf{z}$) will automatically ensure the condition is satisfied.

The normalization of V($\mathbf{z}$) then also requires

$$\sum_{i=1}^{k}\sum_{d=1}^{p} U_{id} = 100$$

Our problem is thus to estimate the U_{id} from available preference information, and then to find $\mathbf{z}$ by maximizing $V(\mathbf{z}) = \sum_{i=1}^{k}\sum_{d=1}^{p} U_{id}\delta_{id}$ subject to

$$z_i = \underline{z}_i + L_i \sum_{d=1}^{p} \delta_{id}, i = 1, \ 2, \ ..., \ k$$

and other constraints

The steps for the simplified interactive piecewise linear value function model implemented in the DSS are outlined below:

Step 1

Let r=0. Generate $1000k$ (where k is the number of objectives) weight vectors of length $4k$, uniform on $\sum_{i=1}^{k}\sum_{d=1}^{4} U_{id}$; $U_{id} \geq 0$. Eliminate the weight vectors which do not satisfy the concavity constraint ($U_{id} < U_{i,d-1}$). Store (save) the weight vectors for future reference.

Step 2

Randomly select a set of $5 - r$ (since the maximum number of iterations is 5 in this method) weight vectors generated in step 1 and solve the LP problems i.e maximize V($\mathbf{z}$) and obtain $5 - r$ solutions.

Step 3

Present $5 - r$ solutions to the DM. Ask the DM to rank order the current solutions. The rank ordering of solutions should be such that the most satisfactory solution is ranked 1 and the least satisfactory solution is ranked $5 - r$. If the DM is satisfied with one of the solutions presented then STOP. Otherwise go to the next step.

Step 4

If the DM (or the user of the DSS) is able to express a definite preference for a solution $\mathbf{z}^a$ over another solution $\mathbf{z}^b$, then we can work in terms of δ^a and δ^b. The DM's preference induce the following constraint on the U_{id}

$$\sum_{i=1}^{k}\sum_{d=1}^{4}(\delta_{id}^{a} - \delta_{id}^{b})U_{id} \geq 0 \tag{4.5}$$

Eliminate the weight vectors which do not satisfy constraint (4.5). Let $r = r + 1$ and go back to step 2.

Comments

The procedure starts by generating five feasible solutions. If the user is not satisfied with any of the solutions provided, he/she is only requested to rank–order the solutions provided in every iteration. The rank–ordering is done until there is only one pairwise comparison of solutions provided.

4.5 Graphical User Interface

The graphical user interface screens and all the buttons in the DSS have been arranged in meaningful hierarchical structure so that their functionality can be easily accessible to the user. Table (4.1) presents the functionalities of each screen in a DSS. Figure (4.1) below presents the accessible switch between the screens of the designed DSS.

Screen No.	Screen Name	Function
1	Main Menu	Starts the DSS application
2	Description Menu	Enables the user to describe the problem
3a	Formulation Menu	Gives access to Formulation Inputs depending on what the user wants to enter
3b	Formulation Inputs	Enables the user to provide for data
4a	Solutions	Enables the user to view the solutions and interactively modify the solutions
4b	View Solutions	Enables the user to view all the solutions after each iteration.

Table 4.1: Screen–Function

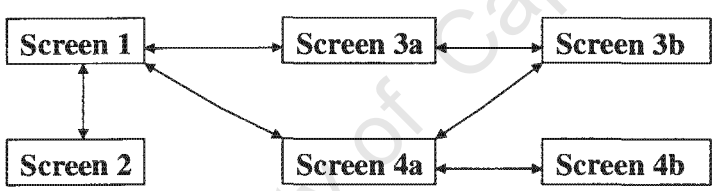


Figure 4.1: Screen–Switch

4.5.1 Main Menu

The main menu allows the user to go to description menu where he/she can describe the whole problem for report purposes. The user can go to formulation menu (where the whole problem can be formulated) through the main menu. The main menu also enables the user to choose the method he/she wants to use. The method to use can be chosen by simply clicking on either of the three methods displayed on the main menu. The menu appears as in the figure below.

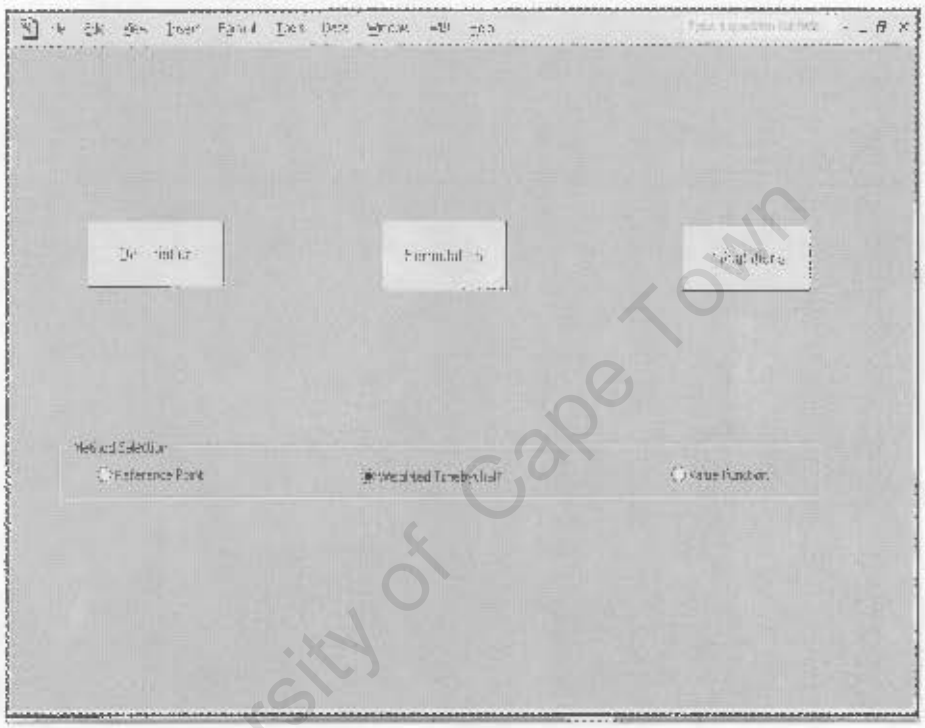


Figure 4.2: Main Menu

4.5.2 Description Menu

The description menu allows the user of a DSS to provide for his/her personal information. It also enables the user to give a brief explanation of the nature of the problem. The purpose of this menu is basically for reports. The description menu looks as in the figure below.

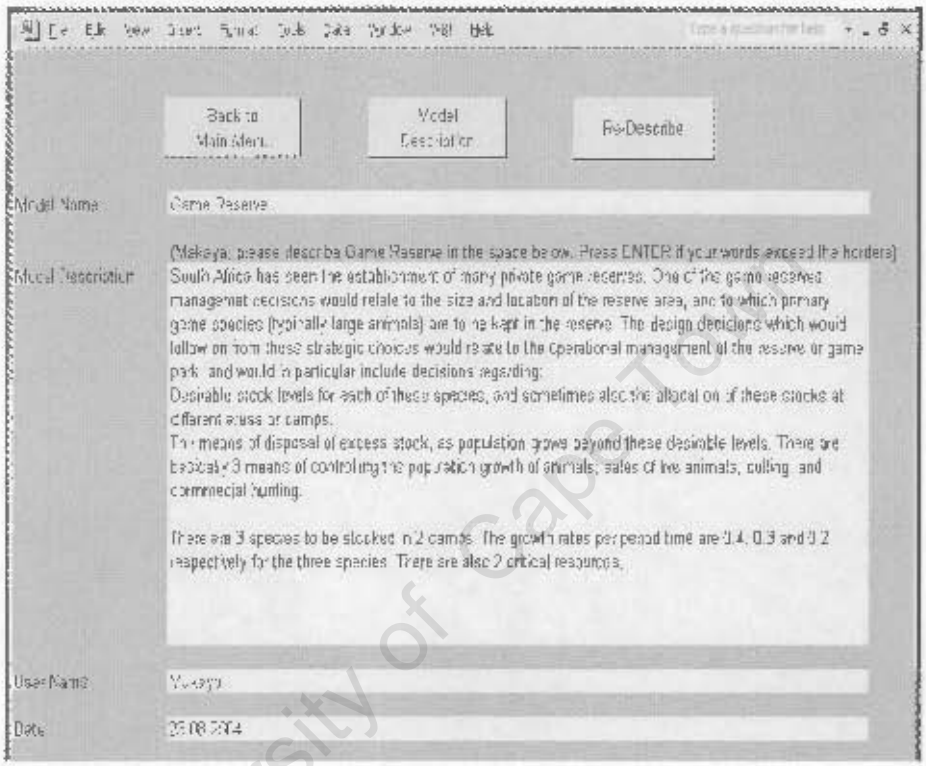


Figure 4.3: Description Menu

4.5.3 Formulation Menu

The formulation menu allows the user of a DSS to provide for the data, basically numerical values of the model to be solved. The formulation menu looks as in the figure below.

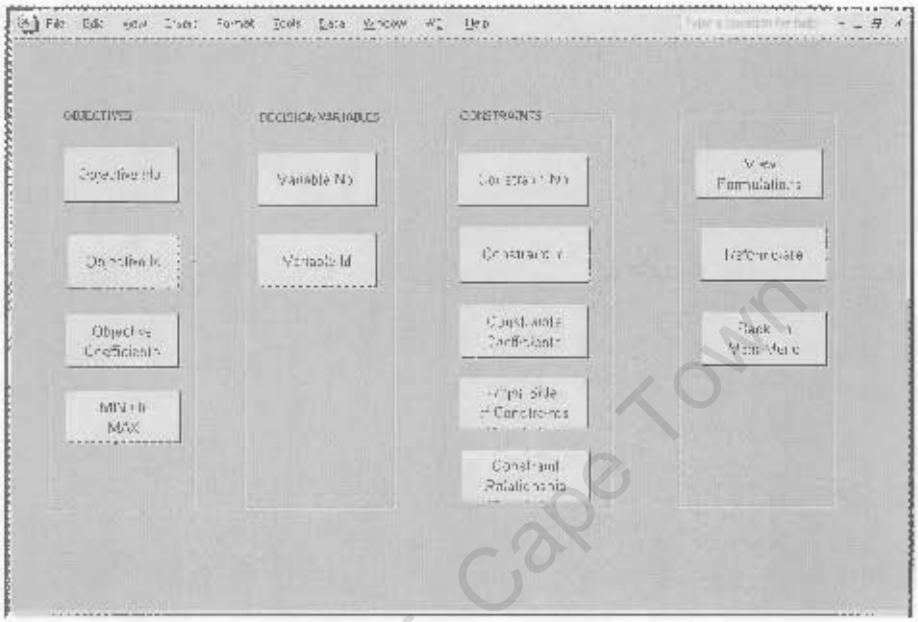


Figure 4.4: Formulation Menu

4.5.4 Formulation Inputs

Suppose, for example, that the user has a problem with three objectives, three decision variables and four constraints. Then the "Formulation Inputs" Menu would appear as in the figure below. In the following figure whether the objectives are to be maximized or minimized is not specified. The figure only gives a rough idea of how the menu looks like. This menu enables the user to formulate the problem by inserting the numeric values in the white spaces, that is, the coefficients of decision variables of the objectives as well as of the constraints. The user can also provide for the values for available resources. The formulation inputs menu looks as in the figure below.

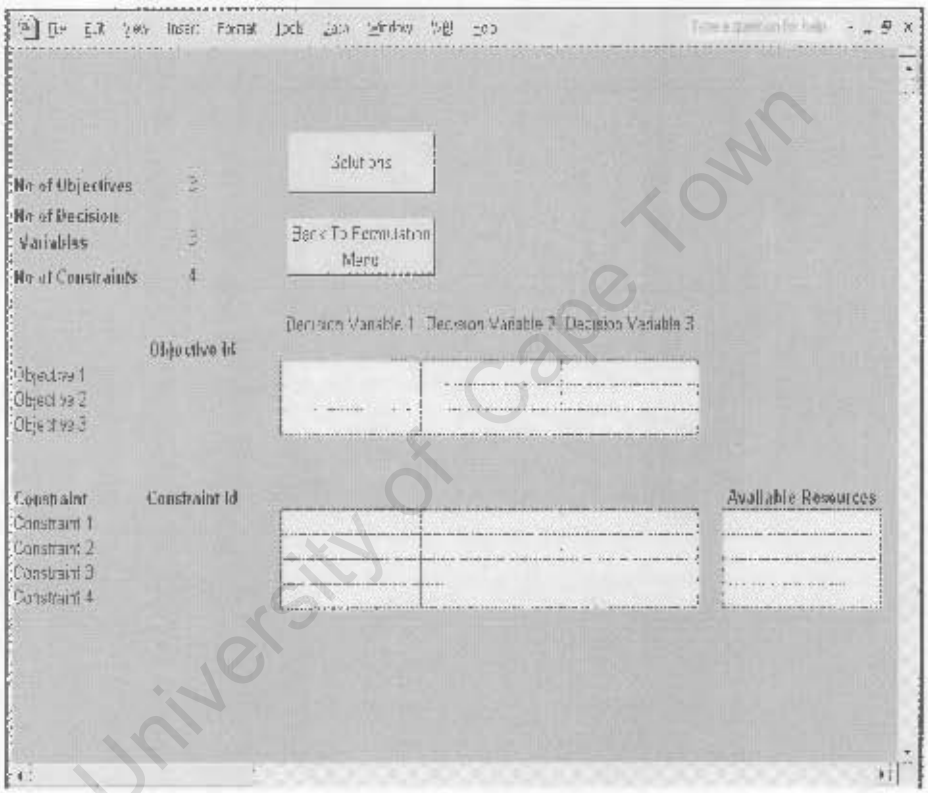


Figure 4.5: Formulation Inputs Menu

4.5.5 Solutions

The solutions menu allows the user to start solving multicriteria problems depending on the method selected. All the user has to do is click on any of the three buttons depending on what he/she wants. This menu allows the user to view the solutions of the current iterations only (except for when the reference point method is selected). The user is guided throughout the whole process. The solutions menu looks as in the figure below.

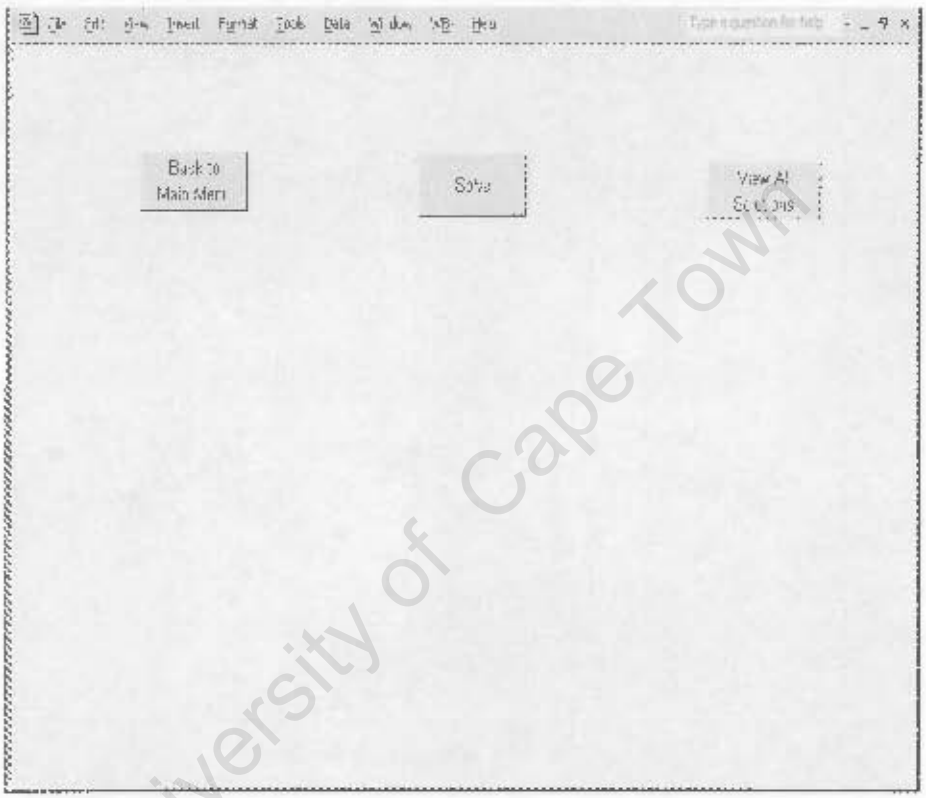


Figure 4.6: Solutions Menu

4.5.6 All Solutions

The all solutions menu enables the user to view all the solutions after all the iterations are performed or during the solution process. This screen shows the solutions depending on the selection of the method. The user can view different solutions by just selecting the method and clicking on "view all solutions"

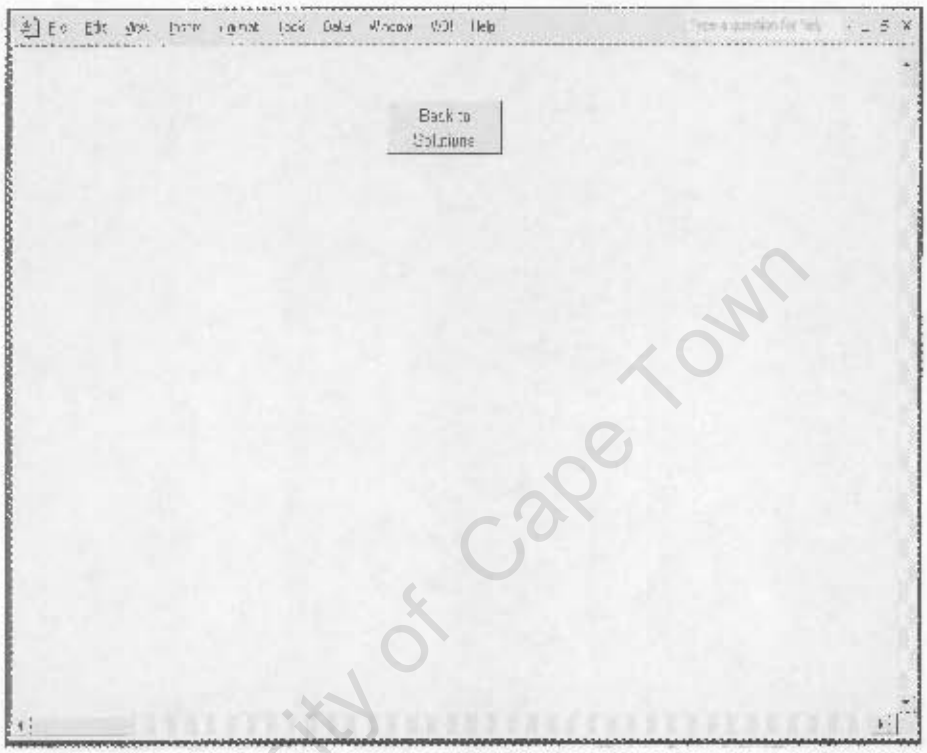


Figure 4.7: All Solutions Menu

4.6 Decision Making Process

The decision making process through a DSS occurs in several phases explained in the following sections. The decision making process is essentially done through the aid of a decision support system.

4.6.1 Problem Description

The DSS enables the user to describe the model for report purpose. This can be done by clicking "Description" on the main menu. Clicking "Description"

takes the user to description menu. The user can click on "Model Description" to get the pop-up and provide for the models name as in figure (4.8).

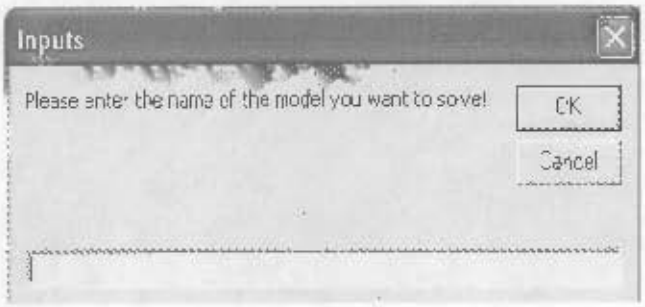


Figure 4.8: Model Name

Then the user can provide for his/her name as in figure (4.9).

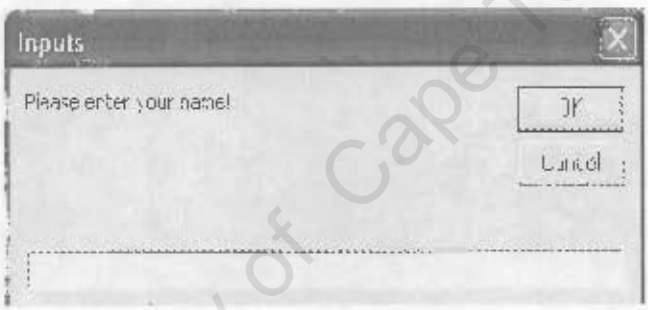


Figure 4.9: User Name

Then the user can provide for the date of DSS use as in figure (4.10).

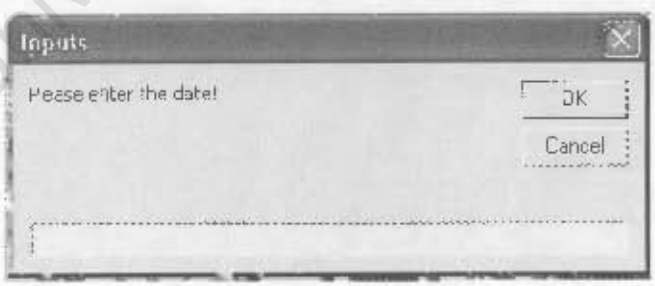


Figure 4.10: Date

The space is then created as in figure (4.3) for the user to briefly and thoroughly describe the model. If the user feels like describing the model once again, he/she can click on "Re–Describe" to clear everything on "Description" menu. When clicking on "Re Describe", the pop–up as in figure (4.11) appears which asks the user whether he/she is positive about re describing the model.

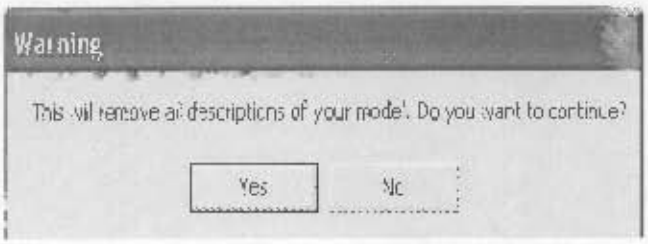


Figure 4.11: Re Description Warning

In this case the user has two options, to clear all the contents in "Description Menu" or to select "No" and leave the "Description Menu" as it is.

4.6.2 Solution Process

Once the method is selected and the problem is formulated, the interactive algorithm (of the method selected through a DSS) request the DM to provide input to the solution algorithm and the results are continually displayed throughout the solution process. Data can be interchanged through different methods depending on the choice of the DM. In other words, once the problem is formulated, there is no need to reformulate the problem for a different MCDM method. After selecting the MCDM method to use, the solution process proceeds in relation to the method selected. The DSS allows the user to change his mind (switch through the methods) at any time during the solution process.

Reference Point Method Solution Process

When the reference point method is selected, the user has to click on "Solve" to follow the reference point method solution process. The ideal values are displayed and the user is requested to provide for his/her aspiration levels for each criterion/objective. The aspiration levels should be less than the ideal values. The DSS warns the user if the aspiration levels are not acceptable and requests the user to review his/her aspiration levels. After providing for the aspiration levels the user has to click on "Solve" once again to get the solution. If the user is not satisfied with the solution he/she can click on "Solve" once again to provide for different aspiration levels and so on and so forth. The solutions in every iteration are saved for final solution selection. The user is guided throughout the reference point solution process. Pictorial examples of the reference point method solution process are found in appendix (C)

Interactive Weighted Tchebycheff Method Solution Process

When the interactive weighted Tchebycheff method is selected, the user clicks on "Solve" in the "Solutions" menu to get the initial solution. After some time a number (5) of different solutions is displayed on the screen. If the user is not satisfied with any of the solutions displayed, he/she can click on "Solve" once again. At this stage the user is requested to provide for a number corresponding to the solution he/she most prefers among the displayed solutions. Yet another set of five solutions is displayed on the screen for the user to select the best solution. If the user is still not satisfied, the process continues likewise until he/she finds the solution that satisfies his/her needs. The solutions in every iteration are saved for future reference. The user is still guided throughout the solution process. Pictorial examples of interactive weighted Tchebycheff method solution process are depicted in appendix (C).

Value Function Method Solution Process

When the value function method is selected, the user has to click on "Solve" in the "Solutions" menu. After some time a number (5) of solutions is displayed on the screen. If the user is not satisfied with the solutions provided he/she has to click on "Solve" once again. Then the user is asked whether he/she is satisfied with the initial solution. If not, the system can provide for yet another set of different initial solutions. From here the user is requested to rank order the solutions provided from 1 to 5 (where 1 refers to the most preferred solution and 5 refers to the least preferred solution). Unlike in the other two methods' solution processes, the value function method solution process has a limited number of iterations. As the number of iterations increases the number of solutions that need to be rank ordered decreases. The interactive solution process terminates after the user has rank ordered only two solutions. At this stage if the user is still not satisfied with the solutions provided, on clicking "Solve" he/she can start the whole solution afresh with a different set of starting solutions. Pictorial examples of the solution process through the value function method are found in appendix (C)

4.7 Concluding Remarks

This chapter covers issues related to development and implementation of a DSS. The development relied, mainly, on prototyping which is characterized as an iterative process of DSS development. Having developed the DSS, we are in the position to test the DSS before implementing it.

Chapter 5

DSS Testing and Evaluation

This chapter deals with the testing and evaluation of an integrated multicriteria decision support system. Such testing and evaluation of a DSS are essential before the implementation of the system. The participants as well as the methodologies for evaluating the DSS are discussed in this chapter.

5.1 Decision Makers

The DSS was tested with human decision makers playing the role of real decision makers. There is a distinction between a real decision maker and a role player. A real DM would be someone who has a multiple criteria decision making problem and is responsible not only for solving it but also for implementing the best compromise solution found [YTE96].

5.2 Test Problems

MCDM problems can be categorized depending on the number of objective functions, decision variables and constraints. Two test problems have been used for testing the DSS. A *laboratory test problem* [see appendix (B) for description and mathematical representation of the problem] was the *game reserve planning problem*[1] taken from Belton and Stewart [VJ02]. The game reserve problem had 6 objectives, 15 decision variables and 8 constraints. A *field test problem* for the DSS had 4 objectives, 20 decision variables and 21 constraints. It was an investment problem (see appendix B for description and mathematical representation of the problem).

[1]The main reason for selecting the game reserve planning problem is that some basic calculations such as for the payoff table, nadir, ideal etc. are known for reference

5.3 Laboratory Test

The intention of this stage is to validate the system in the sense of evaluating the extent to which plausible solutions are generated. In order to verify the internal system structure and to guarantee the system accuracy, the V&V (Verification and Validation)[see section (3.8.3)] was carried out for each MCDM model implemented in the DSS. Validation examines whether the system achieves the project's stated purpose related to helping the user reach decisions. Details of decision support system validation are found in section (3.8.3).

The laboratory test (using the problem described in appendix B) illustrates the solution process throughout the DSS. The same problem was solved using different MCDM methods implemented in a DSS, but with the decision maker's responses simulated by the researcher. All the tests outlined in the next sections were carried out after the problem description and formulation. Detailed steps for taking laboratory tests through three implemented MCDM methods are outlined in the next sections.

5.3.1 Reference Point Test

The laboratory test through the reference point method followed the algorithmic steps outlined in section (4.6.2). The solutions provided below were copied from the DSS "Solution Menu".

Firstly, the reference point method was selected in the "Main Menu". Then clicking on "Solve" in "Solutions menu" provided for ideals and requested the DM to give the reference point. After providing for the reference point, the user had to click on "Solve" again, and the DSS provided for solutions for the first iteration. The reference point and solutions for the first iteration are given in table (5.1).

Objective	Ideal	Ref. Point 1	Solution 1
1	16.8	15	7.7
2	31.2	30	18.95
3	120	110	18.02
4	77.67	75	50.47

Table 5.1: Reference Point Solution For Iteration 1

Since some of the solutions provided in iteration 1 appeared to be below half of the ideal, the user did not find the first solution satisfactory. He had to click on "Solve" again to provide for new aspiration levels (reference point). The new

reference point adjusted all aspirations downward (to be more realistic), with greater adjustments to the second two objectives (indicating that they may have been of lesser importance). Yet a different set of solutions was displayed to the user. The solutions for iteration 2 are outlined in table (5.2).

Objective	Ideal	Ref. Point 2	Solution 2
1	16.8	10	5.12
2	31.2	20	11.97
3	120	80	51.31
4	77.67	60	47.33

Table 5.2: Reference Point Solution For Iteration 2

The solution provided in iteration 2 was still not satisfactory, and the process had to go to the next iterations. In the next iterations, the user still had to provide for the aspiration levels (reference point). The results for iteration 3 to 5 are depicted in table (5.3).

By iteration 4, there was a good match between the DM's aspirations and the solution, with little further improvement at iteration 5. The simulated solution process through the reference point method might thus end at this stage.

5.3.2 Interactive Weighted Tchebycheff Test

The laboratory test through the interactive weighted Tchebycheff method followed the algorithmic steps outlined in section (4.6.2). The solutions provided below were copied from the DSS "Solution Menu".

Firstly, the interactive weighted Tchebycheff method was selected in the "Main Menu". Then clicking on "Solve" in "Solutions menu", the utopian and a set of 5 solutions was displayed on the screen. The solutions for the first iteration are depicted in table (5.4).

After the first iteration the DM did not find any of the solutions to be satisfactory since most of the objectives had the lowest achievable values. Nevertheless, he still needed to select the least unsatisfactory solution of the five, which was solution 5, before clicking on solve to go to iteration 2. The solutions for iteration 2 are depicted in table (5.5).

There was still no satisfactory solution found in iteration 2 since there was no balance in the objective values (some objectives were awarded too high values

		Iteration 3		Iteration 4		Iteration 5	
Objective	Ideal	Ref. Point 3	Solution 3	Ref. Point 4	Solution 4	Ref. Point 5	Solution 5
1	16.8	8.5	7.64	9.24	11.27	12	11.61
2	31.2	16	14.43	12.6	13.95	14	15.41
3	120	55	48.27	52.4	52.4	55	49.73
4	77.67	45.9	42.61	38.4	38.4	40	36.95

Table 5.3: Reference Point Solutions For Iteration 3–5

Objective	Utopian	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
1	16.8	9.81	8.86	5.99	7.49	11.83
2	31.2	15.74	0	0	6.71	17.58
3	120	0	59.4	100.13	93.15	49.69
4	77.67	65.35	61.46	42.53	33	33

Table 5.4: Interactive Weighted Tchebycheff Iteration 1 Solutions

Objective	Utopian	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
1	16.8	3.39	13.3	3.3	6.58	1.04
2	31.2	9.37	2.9	7.95	16.15	2.3
3	120	87.31	10.52	93	42.58	100.87
4	77.67	33	73.48	33	42.95	41.31

Table 5.5: Interactive Weighted Tchebycheff Iteration 2 Solutions

while others had too little). The DM had to go to the next iteration with solution 4 selected as the least unsatisfactory solutions. The solutions for the 3^{rd} iteration are given in table (5.6).

Objective	Utopian	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
1	16.8	6.98	4.69	7.67	4.34	10.62
2	31.2	17.14	11.42	7.18	8.39	23.61
3	120	56.23	79.12	91.28	90.37	0
4	77.67	33	33	33	33	49.23

Table 5.6: Interactive Weighted Tchebycheff Iteration 3 Solutions

The DM was still not fully satisfied with any of the solutions in iteration 3, but selected solution 1 as the least unsatisfactory. The solutions obtained in iteration 4 are shown in table (5.7).

Objective	Utopian	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
1	16.8	10.19	1.08	5.84	1.18	10.04
2	31.2	16.6	2.4	2.59	2.66	17.01
3	120	54.88	100.3	109.64	53.04	0
4	77.67	33	41.41	33	67.48	62.7

Table 5.7: Interactive Weighted Tchebycheff Iteration 4 Solutions

At this stage the DM was satisfied with solution 1. He wanted however, to see if there could be any improvement in the next solutions provided. Solution 1 was selected as the most preferred and the process moved to the next iteration. Iteration 5 solutions are shown in the figure below.

Objective	Utopian	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
1	16.8	6.05	5.73	12.36	4.75	5.02
2	31.2	14.82	14.01	24.73	11.56	12.26
3	120	65.5	68.74	23.47	78.54	75.76
4	77.67	33	33	33	33	33

Table 5.8: Interactive Weighted Tchebycheff Iteration 5 Solutions

The process ended up with 5 iterations. The DM found solution 1 in iteration 4 as the most satisfactory solution even after the 5^{th} iteration.

5.3.3 Value Function Test

The laboratory test through the value function method followed the algorithmic steps outlined in section (4.6.2). The solutions provided below were copied from the DSS "Solution Menu".

Firstly, the value function method was selected in the "Main Menu". Then clicking on "Solve" in "Solutions menu", the ideals and a set of 5 solutions were displayed on the screen. The solutions for the first iteration are depicted in table (5.9).

Objective	Ideal	Solution 1	Solution 2	Solution 3	Solution 4	Solution 5
1	16.8	4.8	9.89	1.11	6.47	3.12
2	31.2	0	2.57	2.46	4.17	7.51
3	120	120	55.91	97.22	103.31	94.76
4	77.67	33	57.66	43	33	33

Table 5.9: Value Function Iteration 1 Solutions

The DM was not satisfied with any of the solutions provided in iteration 1. There was no balance in the values for each objective. He had to click on "Solve" again. He was requested to rank–order the solutions provided. The rank ordering–was in such a way that the most satisfactory solution was ranked 1 and the least satisfactory solution was ranked 5. The DM's rank–ordering for solutions 1, 2, 3, 4 and 5 was 4, 2, 5, 1 and 3 respectively (i.e. solution 1 was ranked 4, solution 2 was ranked 2 and so on) from table (5.9). The solutions for iteration 2 are shown in table (5.10).

Objective	Ideal	Solution 1	Solution 2	Solution 3	Solution 4
1	16.8	11.57	13.11	13.8	13.19
2	31.2	7.8	15.6	17.93	16.41
3	120	40.07	30	23.91	30
4	77.67	55.33	45.65	44.17	44.17

Table 5.10: Value Function Iteration 2 Solutions

The solutions in iteration 2 seemed to be quite good but the DM wanted to explore further. Thus he had to proceed to the next iteration[2]. The DM had to rank–order the solutions in table (5.10). The rank–ordering for solutions 1, 2, 3 and 4 was 4, 1, 2 and 3 respectively. The solutions for iteration 3 are outlined in table (5.12).

After the 3^{rd} iteration the DM found solution 2 essentially satisfactory. However, to see if there could be any improvement he continued with the rank–ordering although he had been happy with solution 2 in iteration 3. The rank–ordering for solutions 1, 2 and 3 was 2, 1 and 3. Then the process went to iteration 4. The solutions for iteration 4 are shown in table (5.12).

[2]The number of solutions provided decreases as the number of iterations increases in the value function method

Objective	Ideal	Solution 1	Solution 2	Solution 3
1	16.8	16.14	13.8	13.8
2	31.2	23.4	15.6	19.6
3	120	0	20.8	0
4	77.67	45.1	49.87	55.33

Table 5.11: Value Function Iteration 3 Solutions

Objective	Ideal	Solution 1	Solution 2
1	16.8	15.58	7.8
2	31.2	17.82	3.39
3	120	0	44.32
4	77.67	55.33	66.5

Table 5.12: Value Function Iteration 4 Solutions

The laboratory test through the value function method ended after 4 iterations with solution 2 in iteration 3 as the most satisfactory solution to the DM.

All the three methods generate quite similar solutions in 3 or 4 iterations. This is a good validation that all the three methods implemented in the DSS do what is desired.

5.4 Field Test

The aim of field testing is to gain confidence that the DSS possesses the required properties. The field testing validation is a very desirable step taken before full implementation of the system. Only when the DSS has been internally verified by laboratory tests should the field test be carried out. The *field test* (using the problem described in appendix B) was carried out by a group of students. The primary reason for performing the field test was to evaluate the DSS so that there could be some modifications (if any) before the DSS is implemented.

5.5 DSS Evaluation

For DSS evaluation, a group of six students simulating real decision makers were given a multicriteria problem (see appendix B) to solve. The nature of a problem was discussed with each of the users before the actual use of the DSS. The problem was formulated and described before the evaluation process (i.e. the users were only requested for input information needed for the solution process). Buttons for implemented methods selection were placed differently for all the 6 users to avoid bias of method selection. All the steps taken in using the DSS were also explained to the users before the use of the DSS. The DSS was evaluated on 4 criteria being ease of use, graphical user interface, methods' evaluation and time. Each criterion was decomposed to sub–criteria as outlined below:

1. **User's Background**
   - (i) MCDM familiarity
   - (ii) Computer literacy
2. **Ease of use**
   - (i) User friendliness
   - (ii) Error tolerance
   - (iii) Ease of getting to the solution
   - (iv) Flexibility of changing one's mind during the solution process
   - (v) Ease of interaction
   - (vi) Getting lost during the solution process

(vii) Flow of steps

(viii) Comfort in using the DSS

(ix) Guidance

(x) Terminology

(xi) Ease of Use in general

3. **Graphical user interface**

   (i) Screen appearance

   (ii) Menus placement

   (iii) Button placement

4. **Implemented methods' evaluation**

   (i) Method preference

   (ii) Method giving satisfactory solution

   (iii) Difference in solutions provided

   (iv) Confidence in solutions

   (v) Method reinitialization during solution process

5. **Time**

   (i) Time taken to reach a satisfactory solution using value function method

   (ii) Time taken to reach a satisfactory solution using interactive weighted Tchebycheff method

   (iii) Time taken to reach a satisfactory solution using reference point method

   (iv) General speed of a DSS

All the sub–criteria measures were obtained by having the users complete questionnaires (see appendix D) after using the DSS. Each subcriteria was measured on a 1 to 5 scale [except for subcriteria 4(i), 4(ii), 4(iii) and 4(v) which requested qualitative resposes] with "5" for the strongest response (i.e. if the user marked "5" for sub–criteria 2(i), he/she would be extremely satisfied with the user friendliness of the DSS). Qualitative comments were also solicited.

5.5.1 Evaluating the Decision Makers

Criterion 1 illustrated above was mainly for evaluating the decision makers not necessarily the DSS. The responses of each student on subcriteria 1(i) and 1(ii) are shown in the table (5.13).

	Sub-criteria	
User	1(i)	1(ii)
1	3	3
2	1	4
3	1	4
4	1	2
5	3	4
6	1	3

Table 5.13: Users' Scores for Sub–criteria 1(i) and 1(ii)

Descriptive statistics for sub–criteria 1(i) and 1(ii) performance and each users' scores on the sub–criteria are respectively depicted in tables (5.14) and (5.15).

Sub-criteria	1(i)	1(ii)
Mean	1.667	3.333
Standard Deviation	1.033	0.816
Variance	1.067	0.667
Minimum	1	2
Maximum	3	4

Table 5.14: Descriptive Statistics for Sub–criteria 1(i) and 1(ii) performance

	User 1	User 2	User 3	User 4	User 5	User 6
Mean	3	2.5	2.5	1.5	3.5	2
Standard Deviation	0	2.121	2.121	0.707	0.707	1.414
Variance	0	4.5	4.5	0.5	0.5	2
Minimum	3	1	1	1	3	1
Maximum	3	4	4	2	4	3

Table 5.15: Descriptive Statistics for Users scores on subcriteria 1(i) and 1(ii)

The level of MCDM familiarity for students who tested the model ranged from 1 to 3. 67% of the students had very little familiarity with MCDM. The level of computer literacy of the students ranged from 2 to 4.

5.5.2 Evaluating the DSS

The analysis of the evaluation was drawn from the questionnaire completed by the students (users) after using the DSS. Only criteria 2, 3, part of 4 and 5 were used for evaluating the DSS. The responses of each student on different sub-criteria are given in the table (5.16). Descriptive statistics for each sub–criteria performance and each users' scores on the sub–criteria are respectively depicted in tables (5.17) and (5.18).

The average matrix for each user's response on each criteria (i.e. criteria 2, 3, part of 4, and 5) used in evaluating the DSS is depicted in table (5.19). The elements of the matrix are means of each user's response on each of the sub-criteria (which constitute criteria scores).

For DSS evaluation process, a two–way analysis of variance (ANOVA) was performed to compare the criteria (2 to 5) performance and users responses towards the criteria. The results of the ANOVA are depicted in the table (5.20).

The ANOVA results show that there is no significant difference in the students responses towards the criteria evaluating the DSS. This is justified by the *P–value* ($P\text{–}value = 0.691 > 0.05$) for users in ANOVA table. On the other hand the ANOVA results show that there is a highly significant difference in criteria performance used for evaluating the DSS. This is as well justified by the *P–value* ($P\text{–}value = 0.003 < 0.01$) in ANOVA table. The above interpretations of the results are further justified by the summary statistics for both users responses and criteria performance in table (5.21). Some of the significant difference in criteria performance might be due the graphical user interface (GUI) issues. The GUI has the mean score of 4.44, which is higher than the score of the other 3 criteria used in evaluating the DSS.

User	Sub-criteria																		
	2(i)	2(ii)	2(iii)	2(iv)	2(v)	2(vi)	2(vii)	2(viii)	2(ix)	2(x)	2(xi)	3(i)	3(ii)	3(iii)	4(iv)	5(i)	5(ii)	5(iii)	5(iv)
1	4	4	5	4	5	4	4	4	5	5	4	5	5	5	3	3	3	4	4
2	4	5	5	3	4	3	4	4	3	3	4	5	4	4	3	5	2	3	4
3	3	3	3	4	2	5	4	3	2	4	3	5	3	4	3	5	4	4	4
4	4	3	3	3	2	3	4	4	3	5	3	4	4	5	4	2	2	5	3
5	3	5	3	3	4	2	4	4	3	4	4	4	5	5	4	3	2	5	3
6	4	4	2	3	3	2	5	3	4	3	3	5	4	4	3	2	3	5	3

Table 5.16: Users' Scores for Sub–criteria

Sub-criteria	2(i)	2(ii)	2(iii)	2(iv)	2(v)	2(vi)	2(vii)	2(viii)	2(ix)	2(x)	2(xi)	3(i)	3(ii)	3(iii)	4(iv)	5(i)	5(ii)	5(iii)	5(iv)
Mean	3.667	4	3.5	3.333	3.333	3.167	4.167	3.667	3.333	4	3.5	4.667	4.167	4.5	3.333	3.333	2.667	4.333	3.5
Standard Deviation	0.516	0.894	1.225	0.516	1.211	1.169	0.408	0.516	1.033	0.894	0.548	0.516	0.753	0.548	0.516	1.366	0.816	0.816	0.548
Variance	0.267	0.8	1.5	0.267	1.467	1.367	0.167	0.267	1.067	0.8	0.3	0.267	0.567	0.3	0.267	1.867	0.667	0.667	0.3
Minimum	3	3	2	3	2	2	4	3	2	3	3	4	3	4	3	2	2	3	3
Maximum	4	5	5	4	5	5	5	4	5	5	4	5	5	5	4	5	4	5	4

Table 5.17: Descriptive Statistics for Sub–criteria Performance

	User 1	User 2	User 3	User 4	User 5	User 6
Mean	4.211	3.789	3.579	3.474	3.684	3.421
Standard Deviation	0.713	0.855	0.902	0.964	0.946	0.961
Variance	0.509	0.731	0.813	0.930	0.895	0.924
Minimum	3	2	2	2	2	2
Maximum	5	5	5	5	5	5

Table 5.18: Descriptive Statistics for Users' Scores on Criteria 2 to 5

		Criterion			
		2	3	4	5
User	1	4.36	5.00	3.00	3.50
	2	3.82	4.33	3.00	3.50
	3	3.30	4.00	3.00	4.25
	4	3.36	4.33	4.00	3.00
	5	3.55	4.67	4.00	3.25
	6	3.27	4.33	3.00	3.25

Table 5.19: Users' Average Score Matrix for 4 Criteria

Source of Variation	SS	df	MS	F	P-value	F crit
Users	0.636	5	0.127	0.615	0.691	2.901
Criteria Performance	4.527	3	1.509	7.294	0.003	3.287
Error	3.103	15	0.207			
Total	8.266	23				

Table 5.20: ANOVA Results

	Mean	*Variance*
User		
1	3.966	0.793
2	3.663	0.313
3	3.638	0.342
4	3.674	0.364
5	3.866	0.380
6	3.464	0.351
Criterion		
2	3.611	0.177
3	4.444	0.119
4	3.333	0.267
5	3.458	0.185

Table 5.21: Summary Statistics

5.5.3 Evaluating MCDM Methods

In evaluating the methods implemented, only evaluation criteria 5 scores were used. For sub–criteria 5(i) and 5(ii), a point (given the value of 1) was given for each user's preference of the implemented method and for the method that gave the satisfactory solution. The average matrices for sub–criteria 5(i) and 5(ii) are respectively depicted in tables (5.22) and (5.23).

		Method		
		Value Function	Reference Point	Weighted Tchebycheff
User	1	1	0	0
	2	0	1	0
	3	0	0	1
	4	1	0	0
	5	1	0	0
	6	0	0	1

Table 5.22: Method Ease of Use Matrix

		Method		
		Value Function	Reference Point	Weighted Tchebycheff
User	1	1	0	0
	2	0	1	0
	3	0	0	1
	4	1	0	0
	5	0	1	0
	6	0	1	0

Table 5.23: Satisfactory Solution Matrix

The summary statistics for matrix (5.22) and (5.23) are respectively depicted in tables (5.24) and (5.25).

Method	*Mean*	*Variance*
Value Function	0.5	0.3
Reference Point	0.166667	0.166667
Weighted Tchebycheff	0.333333	0.266667

Table 5.24: Summary Statistics for Method Ease of Use

Method	*Mean*	*Variance*
Value Function	0.333333	0.266667
Reference Point	0.5	0.3
Weighted Tchebycheff	0.166667	0.166667

Table 5.25: Summary Statistics for Method Giving Satisfactory Solution

Using a chi–square test with 2 degrees of freedom to compare the three methods ease of use, it was found that there is no significant difference in the three methods ease of use. Chi–square test also shows that there is no significant difference in the three methods satisfactory solutions. The $P - value$ for both tests is 0.6 (i.e $P - value \approx 0.6 > 0.05$).

Finally 83% of the students found substantial differences in the solutions provided by the three methods. 67% of the students found that all methods implemented in the decision support system were fail-safe (i.e. there was no need to reinitialize any of the methods during the solution process). Unfortunately, the remaining 33% who found other methods not fail–safe did not specify the methods.

5.6 Concluding Remarks

After testing and evaluating the DSS, some aspects suggested by the students were looked at. Qualitative comments helped in suggesting for modifications of a DSS for better performance. Error-tracking in the reference point method had to be looked upon. Guidance through the reference point method solution process was also refined. The way the user has to proceed from the first iteration in value function method was also changed to suit the users understanding. There were also other minor changes made to the DSS after the field tests.

The results obtained in the laboratory and field tests provide for the validation of the DSS. Having verified and validated the DSS, we are confident that the MCDM methods were properly implemented. The suggested issues for further research and development of the DSS are addressed in the next chapter. The issues include the extension of the DSS by implementing extra MCDM methods. They also include finding ways of making the DSS easily accessible to the users.

Chapter 6

Conclusions and Further Research

In this dissertation, multicriteria decision making, mainly multiobjective linear programming concepts are covered. The study of multiobjective linear programming is made in parallel with the study of integrated multicriteria decision support systems. The main objective of the study was to construct a user–friendly decision support system for solving MOLP problems. Three MOLP methods, being the *reference point, value function and interactive weighted Tchebycheff* methods, are implemented as the model base for the DSS. Some modifications are made to value function and interactive weighted Tchebycheff models. The DSS is coded in Visual Basic for Applications to mask the complexity of mathematics behind the model base. The DSS is basically a spreadsheet application (i.e. all the calculations are done by Microsoft Excel). The system was tested by a group of students for evaluation. Some conclusions were drawn from the responses of the students after using the DSS.

For further research, the value function method is suggested in which the DM's interaction would be provided in such a way that there is a tradeoff between some criteria (the research in this dissertation did not address this issue). Discrete model base are also suggested so that the DSS could solve almost all MCDM problems (not only continuous problems). It is also suggested that the whole system be Microsoft Excel's add–in. This research did not attempt to compare our DSS with available multicriteria decision support systems, which could be done as further research.

Appendix A

Problem Formulation

As mentioned earlier the problem formulation begins by providing for the number of objectives. This is achieved by clicking on "Objective No" in the "Formulation Menu". An iputbox as in the picture below appears:

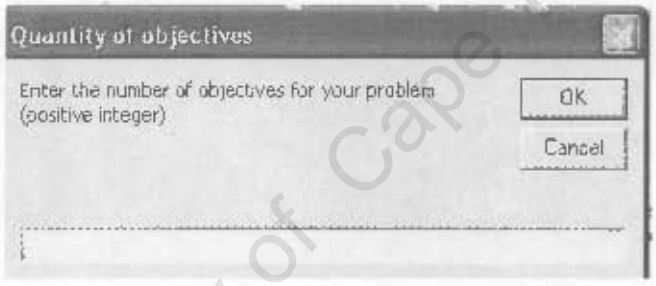


Figure A.1: Objective Number Input

The DSS is capable of trapping any erroneous input information. Suppose the user inserts anything else which is not numeric, then the massage box appears in the figure below:

Figure A.2: Error Message

The user can provide for the number of decision variables by clicking on "Variable No." in the "Formulation Menu". Then the inputbox as in the picture below appears:

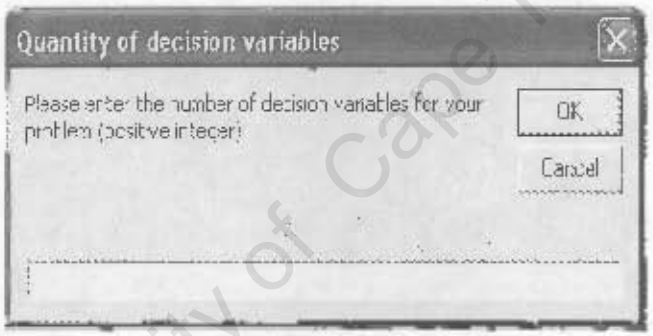


Figure A.3: Decision Variable Number Input

The erroneous information can be trapped as in providing for the number of objectives.

The user can also provide for the number of constraints by clicking on "Constraint No." in the "Formulation Menu". Then the inputbox as in the figure below appears:

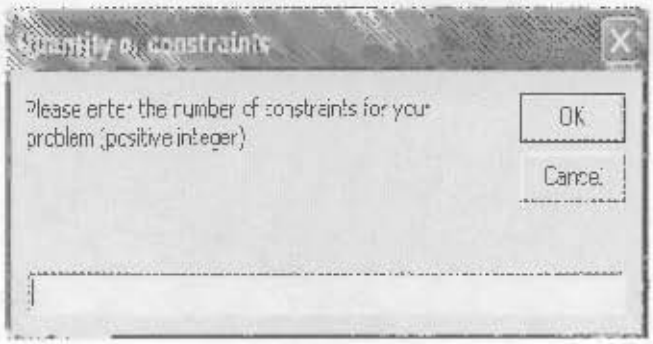


Figure A.4: Error Message

The erroneous information can be trapped as in providing for the number of objectives. In this case the number of constraints cannot be given before the number of objectives is given. In case where it happens that the user decides to provide for the number of constraints before the number of objectives the DSS will inform the user to provide for the number of objectives first as in the figure below:

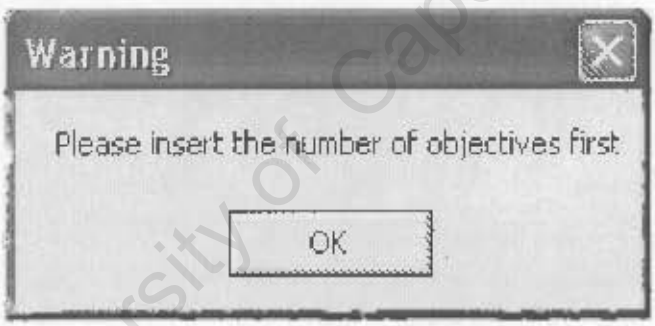


Figure A.5: Error Message

The user can identify (give a name to) each of the objectives by clicking on "Objective Id" in "Formulation Menu". Then the sequence of inputboxes appear until all the objectives are identified. The inputboxes appear as in the figure below:

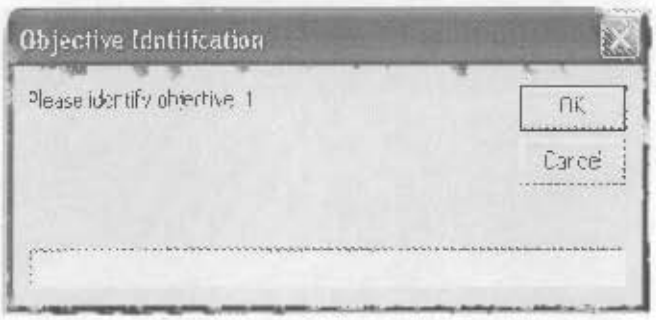


Figure A.6: Objective Identification

Similarly for decision variables and constraint identification, the user has to click on "Variable Id" and "Constraint Id" respectively in the "Formulation Menu". The sequence of inputboxes also appear until all decision variables and constraints are identified.

At this stage the user can click on "Objective Coefficients" in the "Formulation Menu" to provide for the coefficients of decision variables for the objectives. That is, the user has to construct a **C** matrix (see chapter 1). Then the inputbox for guidance appears as in the figure below:

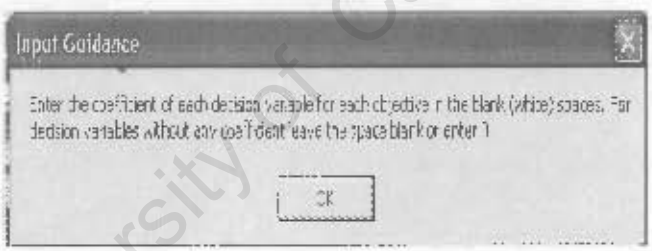


Figure A.7: Objective Coefficients

Then the user is automatically taken to "Formulation Inputs Menu" (see figure 4.5). Similarly for the **A** matrix (coefficients of decision variables for the constraints) and vector **b** (right side of the constraints) formulation, the user has to click on "Constraint Coefficients" and "Right Side of Constraints" respectively in the "Formulation Menu". Likewise the user is guided by message boxes and taken to the "Formulation Inputs Menu". After the construction of **C** matrix, **A** matrix and **b** vector the user has to click on "Back To Formulation Menu" if there is still something else to formulate.

The user is also permitted to provide for the relationship between the left and right sides of the constraints. This can be done by clicking on "Constraint Rela-

tionship" in "Formulation Menu". Then a sequence of inputboxes appear until all the relationships between the left side and the right side of each constraint are provided. The inputboxes appear as in the figure below:

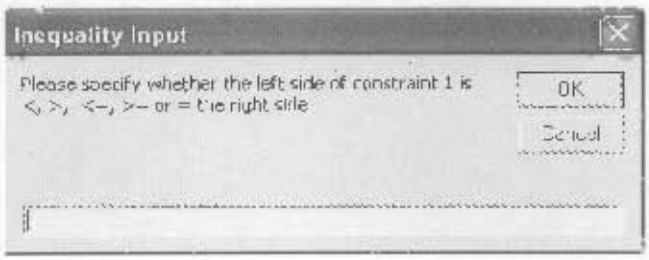


Figure A.8: Constraint Relationship

The user can only type in the signs $<$, $>$, $<=$, $>=$ or $=$ in the input boxes, else the user is informed if ever anything else is inserted.

The user can also specify whether each of the objectives is to be maximized or minimized by clicking on "MIN Or MAX" in "Formulation Menu". Then a sequence of inputboxes appears until all the objectives are stated whether to be maximized or minimized. The inputboxes appear as in the figure below:

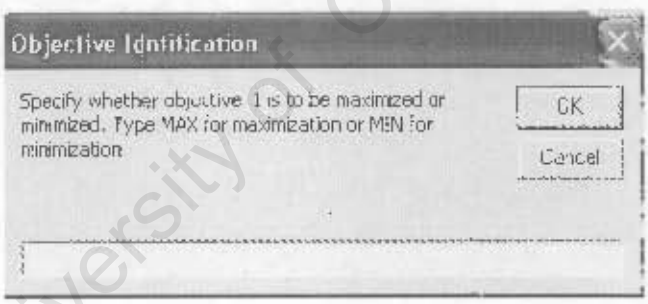


Figure A.9: Objective Status

The user has to type either "min" or "max" (not case sensitive) in the inputbox. Otherwise if something else other than the words "min" and "max" is typed the user is warned.

Clicking on "View Formulations" in "Formulation Menu" allows the user to view the formulated problem in "Formulation Inputs Menu". This is possible only when there is something formulated. Clicking on "Back To Main Menu" takes the user back to "Main Menu".

The DSS also allows the user to reformulate the whole problem. This erases all the formulations and any solutions saved within the model. This is achieved by clicking on "Reformulate". Since deleting might happen mistakenly, the warning appears as in the figure below to ask whether the user really wants to reformulate the problem:

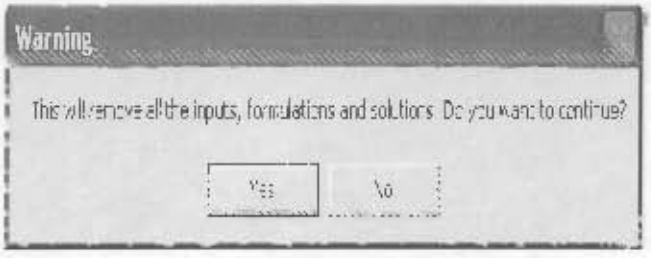


Figure A.10: Delete Warning

Clicking on "No" takes the user back to the "Formulation Menu" without reformulating the problem. Clicking on "Yes" allows the user to reformulate the problem, but also gives the user a chance to save the problem by asking yet another question through the message box. The question appears as in the figure below:

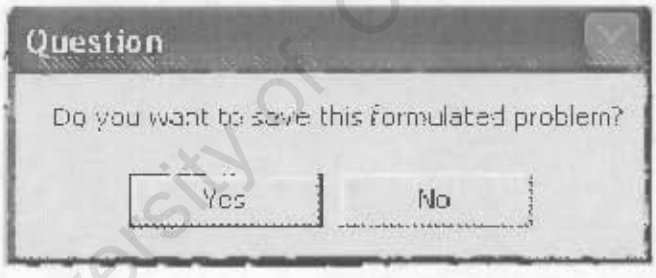


Figure A.11: Delete Question

Clicking on "No" deletes the problem without saving it. Clicking on "Yes" gives the user the chance to specify where to save the problem through the "Save As" inputbox as in the figure below:

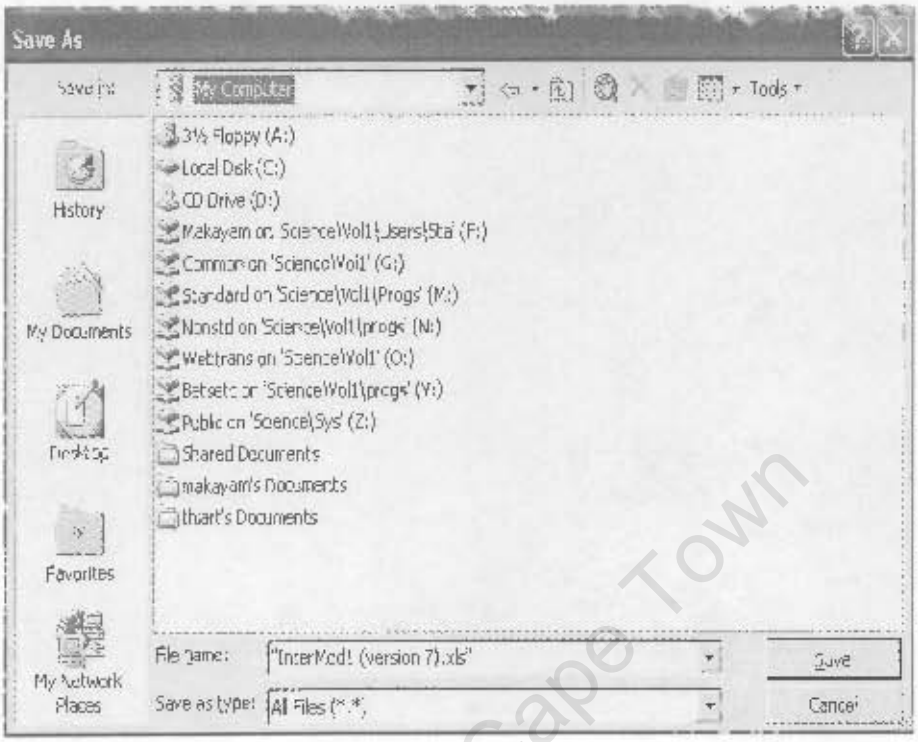


Figure A.12: Save Destination

Appendix B

Test Problems

B.1 Game Reserve Planning Problem

The Game reserve planning problem was taken from Belton and Stewart [VJ02].

B.1.1 The background

South Africa has seen the establishment of many private game reserves. One of the game reserves management decisions would relate to the size and location of the reserve area, and to which primary game species (typically large animals) are to be kept in the reserve. Many different issues for a new game reserve planning such as the following need to be taken into consideration:

i The availability of food and water supplies for the animals in various parts of the reserve: This also requires consideration of an adequate balance between various types of herbivores such as browsers and grazers;

ii Direct and indirect economic benefits derived from tourism, which includes game viewing and hunting (generally according to strictly controlled permits): Such benefits come from entry fees, sale o hunting permits, sale of curios, and by provision of accommodation, meals and refreshments;

iii Direct economic benefits form the sale of live animals (to other reserves, or to zoos) or of meat;

iv Maintenance of good relations with surrounding rural communities, often by permitting grazing of cattle and/or free hunting by traditional methods, and/or providing contributions in cash or kind (e.g. meat) to local villagers: Where these communities do not see a direct benefit to themselves, they may resent the loss of what they often perceive to be their traditional hunting or grazing areas to wealthy interests, and this can lead to problems with poaching or even security for visitors;

v Effects on contributions to national conservation goals, either out of general conservation interests, or because the potential for spin–offs through public image and general increases in eco–tourism.

There are 3 species to be stocked in 2 camps. The growth rates per period time are 0.4, 0.3 and 0.2 respectively for the three species. There are also 2 critical resources.

B.1.2 The Decision

The design decisions which would follow on from the above strategic choices would relate to the operational management of the reserve or game park, and would in particular include decisions regarding:

- Desirable stock levels for each of these species, and sometimes also the allocation of these stocks at different areas or camps.
- The means of disposal of excess stock, as population grows beyond these desirable levels. There are basically 3 means of controlling the population growth of animals; sales of live animals, culling, and commercial hunting.

B.2 Game Reserve Planning Problem Mathematical Representation

$$\begin{aligned}
\text{Max } z_1 &= 2Y_{12} + 2Y_{13} + 3Y_{22} + 6Y_{23} + Y_{32} + 8Y_{33} \\
\text{Max } z_2 &= 5Y_{12} + 3Y_{22} + 2Y_{32} \\
\text{Max } z_3 &= 20Y_{11} + 15Y_{21} \\
\text{Max } z_4 &= 3X_{11} + 15X_{21} + 6X_{31} + X_{12} + 5X_{22} + 2X_{32}
\end{aligned} \tag{B.1}$$

Subject to:

$$\begin{aligned}
-0.4X_{11} - 0.4X_{12} + Y_{11} + Y_{12} + Y_{13} &= 0 \\
-0.3X_{21} - 0.3X_{22} + Y_{21} + Y_{22} + Y_{23} &= 0 \\
-0.2X_{31} - 0.2X_{32} + Y_{31} + Y_{32} + Y_{33} &= 0 \\
X_{11} + 3X_{21} + 8X_{31} &\leq 14 \\
2X_{11} + X_{21} + 4X_{31} &\leq 12 \\
X_{12} + 3X_{22} + 8X_{32} &\leq 25 \\
2X_{12} + X_{22} + 4X_{32} &\leq 30 \\
X_{31} + X_{32} &\geq 14
\end{aligned}$$

where

z_1 = Revenues from Sales and Hunting
z_2 = National Conservation Effort
z_3 = Food Value to Local Communities
z_4 = Viewing Pleasure to Visitors
X_{11} = Quantity of Species 1 to be Placed in Camp 1
X_{12} = Quantity of Species 1 to be Placed in Camp 2
X_{21} = Quantity of Species 1 to be Placed in Camp 1
X_{22} = Quantity of Species 1 to be Placed in Camp 2
X_{31} = Quantity of Species 1 to be Placed in Camp 1
X_{32} = Quantity of Species 1 to be Placed in Camp 2
Y_{11} = Quantity of Species 1 to be Disposed in Manner 1
Y_{12} = Quantity of Species 1 to be Disposed in Manner 2
Y_{13} = Quantity of Species 1 to be Disposed in Manner 3
Y_{21} = Quantity of Species 2 to be Disposed in Manner 1
Y_{22} = Quantity of Species 2 to be Disposed in Manner 2
Y_{23} = Quantity of Species 2 to be Disposed in Manner 3
Y_{31} = Quantity of Species 3 to be Disposed in Manner 1
Y_{32} = Quantity of Species 3 to be Disposed in Manner 2
Y_{33} = Quantity of Species 3 to be Disposed in Manner 3

B.3 Multicriteria Investment Problem

B.3.1 The background

You have just been informed that you have a bequest of R175 000 from a recently deceased relative whom you hardly knew. The terms of the will are that you may use this money for a car and/or for an educational holiday abroad after graduation. For purposes of this test problem, let us assume the following:

- You have use of your father's car for the next year. But after one year you will need to purchase some sort of car, as you will not otherwise have transport available. Assume that the minimum price for a usable second hand car at that time will be R25 000. You are, of course, free to buy a more elaborate vehicle then if you wish. At the end of 3 years you might wish to replace that car or simply to stick with the one bought after one year.
- Study commitments mean that you will be unable to undertake an extensive holiday abroad until three years from now.
- At the end of 3 years, you must make at least a short overseas trip to visit relatives, which can be funded from the bequest. Assume that at least R30 000 must be available for the bequest at that time to cover such a trip. However, you can incorporate this visit into a more extensive holiday at that time.

B.3.2 The Decision

The primary decision that you need to make at this stage is how to invest the money for now, seeing as you will not be using any of it for at least one year. It turns out that there are only four investment options open to you, which differ in terms both of risk and availability of funds after one year. Assessments of cash available per R1 000 invested after one year or after three years, for each of two possible scenarios for economic trends over the three years have been made.

Investment Option	Cash available per R1 000 invested:			
	Scenario I		Scenario II	
	1 Year	3 Years	1 Year	3 Years
A	1070	1240	1120	1440
B	1095	1150	1150	1500
C	850	1100	1175	1650
D	700	950	1000	2000

Experts we have consulted have asserted that there is no good reason to indicate that any scenario is more or less likely than any other, but have not been prepared to associate any probabilities with this outcomes.

Note: The investment decisions have to be made now; you are not permitted to switch funds between investment options at a later stage. The assumption is that the cash you take out after one year will be used for the car, and will not be available for subsequent re–investment. You are, however, allowed to defer the decision as to how much to take out in cash at the end of year 1 until that time (i.e. by the time you know how much each investment has earned for the first year).

B.3.3 The Emphasis

The decision variables are the initial amounts invested in each of the four investment options, the amounts to be withdrawn from each investment after year 1 under each scenario and the amounts to be withdrawn from each investment after year 3 under each scenario.

Your problem is to find the solution which gives the best balance on the following criteria:

- Cash available at the end of year 1 under scenario I
- Cash available at the end of year 1 under scenario II
- Cash available at the end of year 3 under scenario I
- Cash available at the end of year 3 under scenario II

B.4 Multicriteria Investment Mathematical Representation

$$\begin{aligned}
&\text{Max } z_1 = Y_{AI} + Y_{BI} + Y_{CI} + Y_{DI} \\
&\text{Max } z_2 = Y_{AII} + Y_{BII} + Y_{CII} + Y_{DII} \\
&\text{Max } z_3 = P_{AI} + P_{BI} + P_{CI} + P_{DI} \\
&\text{Max } z_4 = P_{AII} + P_{BII} + P_{CII} + P_{DII}
\end{aligned} \tag{B.2}$$

Subject to:

$$\begin{aligned}
X_A + X_B + X_C + X_D &= 175000 \\
Y_{AI} + Y_{BI} + Y_{CI} + Y_{DI} &\geq 25000 \\
Y_{AII} + Y_{BII} + Y_{CII} + Y_{DII} &\geq 25000 \\
P_{AI} + P_{BI} + P_{CI} + P_{DI} &\geq 30000 \\
P_{AII} + P_{BII} + P_{CII} + P_{DII} &\geq 30000 \\
Y_{AI} &\leq 1.07X_A \\
Y_{AII} &\leq 1.12X_A \\
Y_{BI} &\leq 1.095X_B \\
Y_{BII} &\leq 1.15X_B \\
Y_{CI} &\leq 0.85X_C \\
Y_{CII} &\leq 1.175X_C \\
Y_{DI} &\leq 0.7X_D \\
Y_{DII} &\leq X_D \\
P_{AI} &= 1.24X_A - 1.159Y_{AI} \\
P_{AII} &= 1.44X_A - 1.286Y_{AII} \\
P_{BI} &= 1.15X_B - 1.05Y_{BI} \\
P_{BII} &= 1.5X_B - 1.304Y_{BII} \\
P_{CI} &= 1.1X_C - 1.294Y_{CI} \\
P_{CII} &= 1.65X_C - 1.404Y_{CII} \\
P_{DI} &= 0.95X_D - 1.357Y_{DI} \\
P_{DII} &= 2X_D - 2Y_{DII}
\end{aligned}$$

where

z_1 = Amount Available to Withdraw After Year 1 Under Scenario I
z_2 = Amount Available to Withdraw After Year 1 Under Scenario II
z_3 = Amount Available to Withdraw After Year 3 Under Scenario I
z_4 = Amount Available to Withdraw After Year 3 Under Scenario II

X_A = Amount to Invest in Option A
X_B = Amount to Invest in Option B
X_C = Amount to Invest in Option C
X_D = Amount to Invest in Option D
Y_{AI} = Amount to Spend after Year 1 From Option A Scenario I
Y_{AII} = Amount to Spend after Year 1 From Option A Scenario II
Y_{BI} = Amount to Spend after Year 1 From Option B Scenario I
Y_{BII} = Amount to Spend after Year 1 From Option B Scenario II
Y_{CI} = Amount to Spend after Year 1 From Option C Scenario I
Y_{CII} = Amount to Spend after Year 1 From Option C Scenario II
Y_{DI} = Amount to Spend after Year 1 From Option D Scenario I
Y_{DII} = Amount to Spend after Year 1 From Option D Scenario II
P_{AI} = Amount to Spend after Year 3 From Option A Scenario I
P_{AII} = Amount to Spend after Year 3 From Option A Scenario II
P_{BI} = Amount to Spend after Year 3 From Option B Scenario I
P_{BII} = Amount to Spend after Year 3 From Option B Scenario II
P_{CI} = Amount to Spend after Year 3 From Option C Scenario I
P_{CII} = Amount to Spend after Year 3 From Option C Scenario II
P_{DI} = Amount to Spend after Year 3 From Option D Scenario I
P_{DII} = Amount to Spend after Year 3 From Option D Scenario II

Appendix C

Solution Process

C.1 Solution Process Through Reference Point Method

As mentioned earlier, the reference point method starts by selecting "Reference Point" in the "Main Menu" followed by clicking "Solutions" in the same menu. Then the user is taken to "Solutions Menu" where he/she can click on "Solve" to start solving the problem. Immediately after clicking on solve, the message box guiding the user appears as in the figure below:

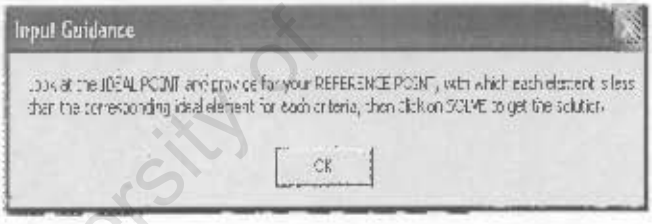


Figure C.1: Reference Point Input Guidance

The user is then taken to "Formulation Inputs Menu" where he/she can provide for his/her aspiration levels (reference point). Figure (C.2) shows the reference point input space in a DSS:

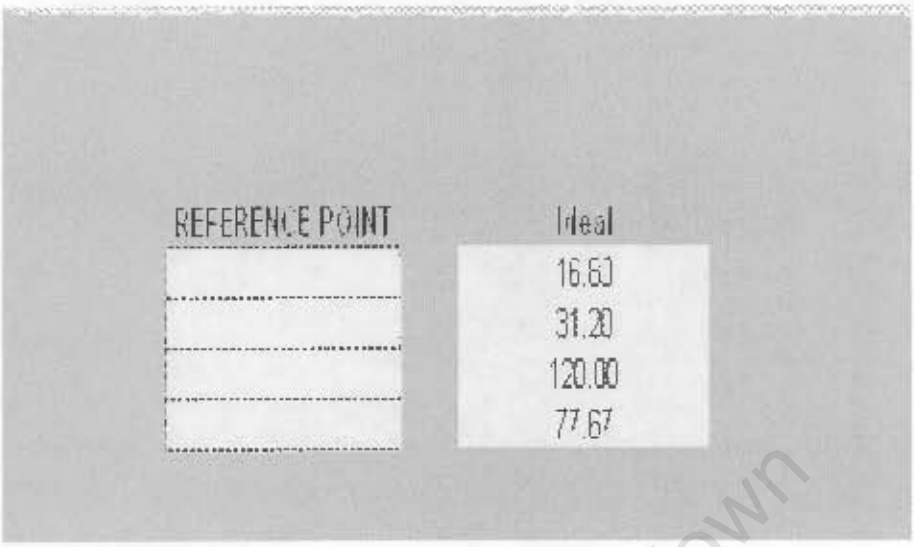


Figure C.2: Reference Point Input

The user has to click on "Solutions" in the "Formulation Inputs Menu" to go back to "Solutions Menu", then click on "Solve" once again. If ever there are errors in the aspiration levels (reference point) provided, the user is informed before the calculations proceed. If there are no errors in the reference point provided by the user, the solution process continues. After every solution calculation a message box as in the figure below appear:

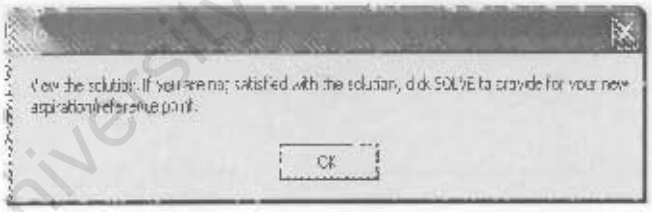


Figure C.3: Reference Point Input Guidance

This procedure goes on and on until the user is satisfied with one of the solutions provided.

C.2 Solution Process Through Interactive Weighted Tchebycheff Method

After the selection of Interactive Weighted Tchebycheff Method in the "Main Menu", the user can click on "Solutions" to go to "Solutions Menu". He/she can click on "Solve" in "Solutions Menu". After some time the solutions will be displayed and the message box as in the figure below appears to guide the user with the next step.

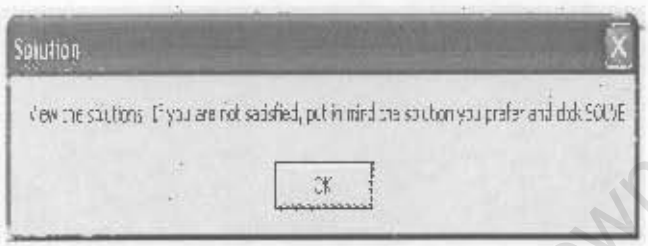


Figure C.4: Iteractive Weighted Tchebycheff Guidance

If the user is not satisfied with the solutions provided, he/she can click on "Solve" once again and the inputbox (with guiding message) as in the figure below appears.

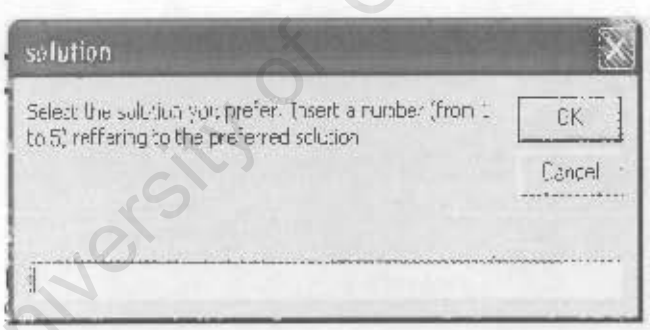


Figure C.5: Interactive Weighted Tchebycheff Input

The solution process continues similarly until the user is satisfied with one of the solutions provided.

C.3 Solution Process Through Value Function Method

The solution process with this method also starts by selecting "Value Function" in the "Main Menu" followed by clicking "Solutions" to get to "Solutions Menu". At this stage the user has to click "Solve" to get the initial solutions. After the initial solution calculation, the message box as in the figure below appears to guide the user of what to do.

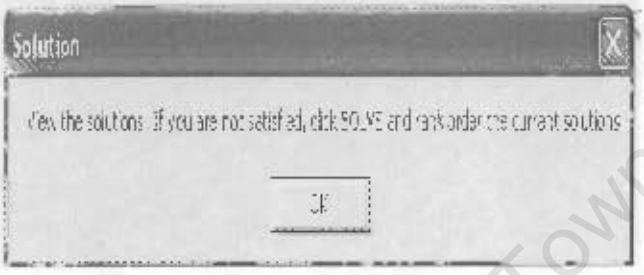


Figure C.6: Value Function Guidance

If the user is not satisfied with the solutions provided then he/she has to click on "Solve" again. There is a message box (as in the figure below) which asks the user whether he/she is satisfied with the initial solutions. This is because it is likely that the initial solutions are similar.

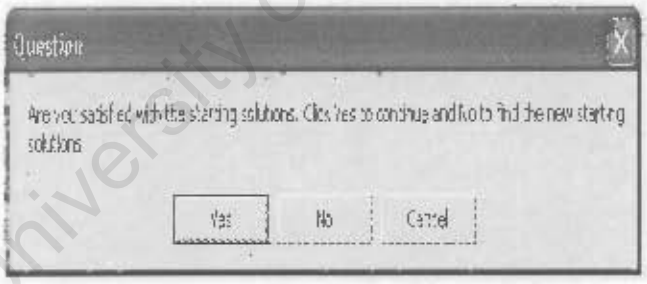


Figure C.7: Value Function Interaction

If the user is not satisfied with the initial solution then he/she has to click on "No" and the new starting solutions will be calculated. The process can go on until the user is satisfied with the starting solution. Otherwise, if the user is satisfied with the initial solution he/she has to click "Yes" and an inputbox as in the figure below appears for the user to rank order the solutions provided.

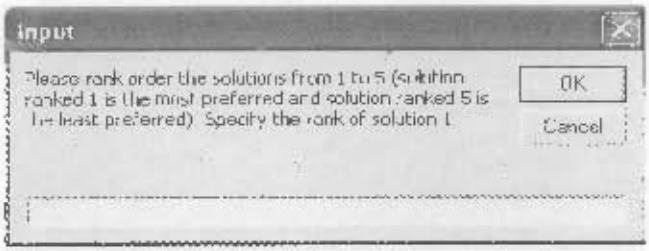


Figure C.8: Value Function Interaction Inputbox

A sequence of inputboxes appear until all the solutions are ranked. The above process occurs until only one solution is displayed. If the user is still not satisfied with any of the solutions provided, he/she has to click on "Solve", but at this moment a different message box appears (as in the figure below) to ask the user if he/she wants to start the process afresh.

Figure C.9: Value Function Interaction Guidance

If the user does not want to start afresh he/she has to click on "No". Otherwise if he/she wants to start afresh he/she has to click on "Yes". Starting afresh gives yet a different starting solution and the process continues likewise.

Bibliography

[A55] Simon H A. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–119, 1955.

[A90] Simon H A. Prediction and prescription in systems modeling. *Operations Research*, 38(1):7–14, 1990.

[A96] Dutta A. Integrating AI and optimization for decision support: A survey. *Decision Support Systems*, 18:217–226, 1996.

[AC02] Hajidimitriou Yannis A and Georgiou Andreas C. A goal programming for partner selection decisions in international joint ventures. *European Journal of Operational Research*, 138:649–662, 2002.

[AJA72] Geoffrion A.M., Dyer J.S., and Feinberg A. An interactive approach for multicriterion optimization, with an application to the operation of an academic department. *Management Science*, 19:357–368, 1972.

[AMR03] Gómez-Limón J A, Arriaza M, and Laura Riesgo. An MCDM analysis of agricultural risk aversion. *European Journal of Operational Research*, 151:569–585, 2003.

[A.P80] Wierzbicki A.P. The use of reference objectives in multiobjective optimisation. In Fandel G. and Gal T., editors, *MCDM theory and Application Proceedings*, number 177 in Lecture notes in economics and mathematical systems, pages 468–486, Hagen, 1980. Springer Verlag.

[AP01] Arbel Ami and Korhonen Pekka. Objective values to start multiobjective linear programming algorithms. *European Journal of Operational Research*, 128:587–596, 2001.

[ARL82] Goicoechea Ambrose, Hansen Don R, and Duckstein Lucien. *Multiobjective Decision Analysis with Engeneering and Business Applications*. John Wiley and Sons, New York, 1982.

[ATTA88] Lewandowski A, Kreglewski T, Rogowski T, and Wierzbicki A. Decision support systems of DIDAS family (Dynanic Interactive Decision Analysis Support). In Grauer M and Wierzbicki A, editors, *Theory,*

Software and Testing Examples in Decision Support Systems, pages 18–41. IISA, WP–88–071, 1988.

[AW61] Charnes A and Cooper W W. *Management Models and Industrial Applications of Linear Programming.* Wiley, New York, 1961.

[AWO55] Charnes A, Cooper W W, and Ferguson R O. Optimal estimation of executive compensation by linear programming. *Management Science*, 1:138–151, 1955.

[BaJE02] Malakooti Behnam and Al alwani Jumah E. Extrimist vs. centrist behavior: Quasi-convex utility functions for interactive multiple objective linear programming problems. *Computers and Operations Research*, 29:2003–2021, 2002.

[BM01] Foued A B and Sameh M. Application of goal programming in a multi–objective reservoir operation model in tunisia. *European Journal of Operational Research*, 133:352–361, 2001.

[BO01] Aouni Beland and Kettani Ossama. Goal programming model: A glorious history and a promising future. *European Journal of Operational Research*, 133:225–231, 2001.

[BR04] Urli B and Nadeau R. PROMISE/senarios: An interactive method for multiobjective stochastic linear programming under partial uncertainty. *European Journal of Operational Research*, 155:361–372, 2004.

[BT01] Aghezzaf B. and Ouaderhman T. An interactive interior point algorithm for multiobjective linear programming problems. *Operational Research Letters*, 29(4):163–170, 2001.

[C91] Romero C. *Handbook of Critical Issues in Goal Programming.* Pergamon Press, Oxford, 1991.

[C01] Romero C. Extended lexicographic goal programming: A unifying approach. *Omega: The International Journal of Management Science*, 29:63–71, 2001.

[C04] Romero C. A general structure of achievement function for a goal programming model. *European Journal of Operational Research*, 153:675–686, 2004.

[CC01] Bradley J C and Millspaugh A C. *Advanced Programming Using Visual Basic 6.* Irwin/McGraw Hill, New York, 2001.

[CMF98] Romero C, Tamiz M, and Jones D F. Goal programming, compromise programming and referece point method formulation : Linkages and utility interpretations. *Journal of the Operational Research Society*, 49:986–991, 1998.

[CR93] Reeves C and Hedin S R. A generalized interactive goal programming procedure. *Computers and Operations Research*, 20:747–753, 1993.

[CT03] Novak D C and Ragsdale C T. A decision support methodology for stochastic multi–criteria linear programming using spreadsheets. *Decision Support Systems*, 36:99–116, 2003.

[D98] Borenstein D. Towards a practical method to validate decision support systems. *Decision Support Systems*, 23:227–239, 1998.

[E76] Steuer R E. Multiple objective linear programming with interval criterion weights. *Management Science*, 23(3):305–316, 1976.

[E86] Steuer Ralph E. *Multiple Criteria Optimization: Theory, Computation and Application.* John Wiley and Sons, New York, 1986.

[E88] Steuer R E. Computational experience concerning payoff tables and minimum criterion values over the efficient set. *European Journal of Operational Research*, 33(1):91–97, 1988.

[E89] Markland Robert E. *Topics in Management Science.* New York, John Wiley and Sons, 1989.

[E90] Steuer Ralph E. Interactive multiple objective programming: Concepts, current status, and future directions. In Carlos A and Bana e Costa, editors, *Readings in Multiple Criteria Decision Aid*, pages 413–444. Spinger-Verlag, Berlin, 1990.

[E97] Ballestero E. Utility functions: a compromise programming approach to specification and optimization. *Journal of Multi–criteria Decision Analysis*, 6:11–16, 1997.

[EJ82] Jacquet-Lagrèze E and Siskos J. Assessing a set of additive utility functions for multicriteria decision making, the UTA method. *European Journal of Operational Research*, 10(2):151–164, 1982.

[EJW93] Steuer R E, Silverman J, and Whisman A W. A combined Tchebycheff/Aspiration criterion vector interactive multiobjective programming procedure. *Management Science*, 39:1255–1260, 1993.

[ERR87] Jacquet-Lagrèze E, Meziani R, and Slowinski R. MOLP interactive assessment of a piecewise utility function. *European Journal of Operational Research*, 31(3):95–110, 1987.

[FE93] Tamiz M Jones D F and El-Darzi E. A review of goal programming and its applications. *Annals of Operational Research*, 58:59–63, 1993.

Appendix D

Questionnaire

This questionnaire aims to gather information on your experience using a multi-criteria decision support system (MC–DSS). Your information helps to evaluate the DSS before it is implemented. In completing this questionnaire, we are interested in your opinion on various aspects of the DSS.

Note:

1. We need answers to all questions (i.e. please do not skip any question)
2. Move rapidly through the questionnaire. We are interested in your first impression. Do not spend an excessive amount of time on each question. The questionnaire should take no more than 10 minutes of your time to complete.
3. For each question please circle a number from 1 to 5, where 1 represents the least satisfactory outcome of the corresponding question and 5 represents the most satisfactory outcome of the corresponding question.
4. For questions which require a written statement, provide for your response in the space provided.

D.1 Personal Information

Sex:

1 Female 2 Male

Educational Background: _____

D.2 Background

(i) **How familiar are you with multicriteria decision making (MCDM)?**

1 2 3 4 5

(ii) **How do you rank your level of computer literacy?**

1 2 3 4 5

D.3 Ease of Use

(i) **How user–friendly did you find the DSS?**

1 2 3 4 5

(ii) **How error–tolerant did you find the DSS?**

1 2 3 4 5

(ii) **How easy did you get to the solution?**

1 2 3 4 5

(iv) **How flexible did you find the DSS in changing your mind during the solution process?**

1 2 3 4 5

(v) **How easy did you interact with the DSS?**

1 2 3 4 5

(vi) **How often did you get lost while using the DSS?**

1 2 3 4 5

(vii) **Is the flow of steps logical?**

1 2 3 4 5

(viii) **Did you feel comfortable in using the DSS?**

1 2 3 4 5

(ix) **Is the guidance enough through the solution process?**

1 2 3 4 5

(x) **Were you comfortable with the terminology you came across?**

1 2 3 4 5

(xi) **Was it easy for you to find your way throughout the use of the DSS?**

1 2 3 4 5

D.4 Graphical User Interface

(i) **How do rate the screen appearance (look and feel)?**

1 2 3 4 5

(ii) **Are the menus logically organized (i.e. are similar and related actions grouped together)?**

1 2 3 4 5

(iii) **Are items such as buttons placed in a sensible way?**

1 2 3 4 5

D.5 Method Evaluation

(i) **Which method do you prefer (easy to use) in a DSS?**

1 Value Function 2 Reference Point 3 Interactive Weighted Tchebycheff

(ii) **Which method gave the most satisfactory solution?**

1 Value Function 2 Reference Point 3 Interactive Weighted Tchebycheff

(iii) **Did you perceive substantial differences in the solutions obtained by the three methods?**

1 Yes 2 No

(iv) **How confident were you that you had achieved as good a solution as possible?**

1 2 3 4 5

(v) **Were the methods fail–safe (i.e. there was no need to reinitialize the method during the solution process)?**

1 Yes 2 No

If *No*, which method(s) was(were) not fail–safe? _____

_____

D.6 Time

(i) **How quickly did you get to the solution using Value Function Method?**

1 2 3 4 5
(very slowly) (very quickly)

(ii) **How quickly did you get to the solution using Interactive Weighted Tchebycheff Method?**

1 2 3 4 5
(very slowly) (very quickly)

(iii) **How quickly did you get to the solution using Reference Point Method?**

1 2 3 4 5
(very slowly) (very quickly)

(iv) **How do you rate the general speed of the DSS?**

1 2 3 4 5
(very slow) (very good)

D.7 Comments

(i) What did you like most about the DSS? _____

_____

(ii) What did you dislike most about the DSS? _____

(ii) Other suggestions or comments: _____

Thank you for your cooperation.

Makaya L Makaya

Appendix E

VBA Code

The VBA code can be accessed from the compact disk attached to the write–up. To view the code, the file DSS in the compact disk should be opened. Then a pop-up for 'security warning' appears. The macros should be enabled by clicking on "Enable Macros". The VBA code can only be seen in the Visual Basic Editor window. The Visual Basic Editor window can be accessed under *Tools|Macro|Visual Basic Editor* from the Excel window.

[FEL86] Szidarovszky Ferenc, Gershon Mark E, and Duckstein Lucien. *Techniques for Multiobjective Decision Making in Systems Management.* Elsevier Science Publishers, Amsterdam, 1986.

[H52] Markowitz H. Portfolio selection. *Journal of Finance*, 7:77–91, 1952.

[H56] Schein E H. The chinese indoctrination program for prisoners of war. *Psychiatry*, 19:149–72, 1956.

[H59] Markowitz H. *Portfolio selection: Efficient diversification of investments.* John Wiley and Sons, New York, 1959.

[HY84] Nakayam H and Sawaragi Y. Satisficint trade–off method for multiobjective programming. In Grauer M and Wierzbicki A, editors, *Interactive Decision Analysis*, pages 113–122. Springe Verlag, Berlin, 1984.

[I83] Gass S I. Decision–aiding models: Validation, assessment and related issues for policy analysis. *Operations Research*, 31(4):603–631, 1983.

[IBG01] Nhantumbo I, Dent J B, and Kowero G. Goal programming: Application in the management of miambo woodland in mozambique. *European Journal of Operational Research*, 133:310–322, 2001.

[IG03] Gass Saul I. and Roy Pallabi Guha. The compromise hypersphere for multiobjective linear programming. *European Journal of Operational Research*, 144(3):459–479, 2003.

[J72] Rawls J. *A Theory of Justice.* Oxford University Press, Oxford, 1972.

[J81] Spronk J. *Interactive Multiple Goal Programming.* Martinus Nijhoff Publishing, 1981.

[J85] Siskos J. *Analyses de règression et programmation lenèaire.* Revue de Statistique Appliquèe 33(1), Cahier du ALAMSADE no. 54, Universitè de Paris–Dauphine, May 1984, 1985.

[J87] Emery J. *The Critical Strategic Resource.* Oxford University Press, New York, 1987.

[J93] Stewart T J. Use of piecewise linear value functions in interactive multicriteria decision support: A Monte Carlo study. *Management Science*, 39(11):1369–1381, 1993.

[J95] Schniederjans M J. *Goal Programmming: Methodology and Applications.* Kluwer Academic Publishers, Boston, 1995.

[J96] Stewart T J. Robustness of additive value function methods in MCDM. *Journal of Multi–criteria Decision Analysis*, 5:301–309, 1996.

[J99a] Stewart T J. Evaluation and refinement of aspiration–based methods in mcdm. *European Journal of Operational Research*, 113:643–652, 1999.

[J99b] Stewart Theodor J. Concepts of interactive programming. In Gal Thomas, Stewart Theodor J, and Hanne Thomas, editors, *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications*, pages 10–1 – 10–28. Kluwer Acacemic Publishers, Massachusetts, 1999.

[J04] Walkenbach J. *Excel 2003 Power Programming with VBA*. Wiley Publishing Inc., 2004.

[JB90] Martel J and Aouni B. Incorporating the decision–maker's preferences in goal programming model. *Journal of the Operational Research Society*, 41:1121–1132, 1990.

[JB98] Martel J and Aouni B. Diverse imprecise goal programming model formulations. *Journal of Global Optimization*, 1:127–138, 1998.

[JC82] Sprague R H Jr. and Carlson E C. *Building Effective Decision Support Systems*. Prentice Hall, Englewood Cliffs, NJ, 1982.

[JC97] Ringuest J.L. and Downing C.E. Multiobjective linear programming with context-dependent preferences. *Journal of the Operational Research Society*, 48(7):714–725, 1997.

[JC00] Alves M J and Clímaco J C. An interactive reference point approach for multiobjective mixed–integer programming using branch–and–bound. *European Journal of Operational Research*, 124:478–494, 2000.

[JHG90] Powers M J, Cheney P H, and Grow G. *Structured Systems Development: Analysis, Design, Implementation*. Boyd and Fraser Publishing, Boston, 1990.

[JJ86] Korhonen Pekka J and Laakso Jukka. Visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research*, 24:277–287, 1986.

[JMEE90] O'leary T J, Goul M, Moffitt K E, and Radwan A E. Validating expert systems. *IEEE Expert*, 5(3):51–58, 1990.

[JS88] Miser H J and Quade E S. Validation. In Miser H J and Quade E S, editors, *Handbook of System Analysis–Craft Issues and Procedural Choices*. Wiley, UK, 1988.

[JS03] Chen Jian and Lin Song. An interactive neural network-based approach for solving multiple criteria decision-making problems. *Decision Support Systems*, 36(2):137–146, 2003.

[KH01] Kim Jae Kyeong and Choi Sang Hyun. A utility range-based interactive group support system for multiattribute decision making. *Computers and Operations Research*, 28(5):485–503, 2001.

[L92] Ringuest J L. *Multiobjective Optimization: Behavioral and Computational Considerations.* Kluwer Academic Publishers, Boston, 1992.

[LC97] Winston W L and Albright S C. *Practical Management Science Spreadsheet Modeling and Applications.* International Thompsom Publishing Inc., Mexico, 1997.

[LDM94] Whitten J L, Bently L D, and Barlow V M. *Systems Analysis and Design Methods.* Richard D. Irwin Inc., Sydney, 1994.

[LH76] Keeney R L and Raiffa H. *Decisions with Multiple Objectives: Preferences and Value Trade–offs.* John Wiley and Sons, New York, 1976.

[LRKM80] Hwang C L, Paidy S R, Youn K, and Muad A S M. Mathematical programming with multiple objectives: A tutorial. *Computers and Operations Research*, 7:5–31, 1980.

[M72] Lee S M. *Goal Programming for Decision Analysis.* Auerbach, Philadelphia, 1972.

[M81] Zeleny M. The pros and cons of goal programming. *Computers and Operations Research*, 8:357–359, 1981.

[M82] Sakawa M. Interactive multiobjective decision making by the sequential proxy aptimization technique: SPOT. *European Journal of Operational Research*, 9:386–396, 1982.

[M91] Makowski M. Guidelines for software development for decision support systems. Working Paper WP–91–15, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1991.

[M94a] Makowski M. Design and implementation of model–based decision support systems. Working Paper WP–94–86, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[M94b] Makowski M. Methodology and a modular tool for multiple criteria analysis of lp models. Working Paper WP–94–102, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[M99] Marakas G M. *Decision Support Systems in the 21st Century.* Prentice–Hall Inc., New Jersey, 1999.

[M00] Sun M. A multiple objective programming approach for determining faculty salary equity adjustments. *European Journal of Operational Research*, 138:302–319, 2000.

[MA95] Kompan M and Bera Achintya. Application of goal programming in project selection decision: A case study form the indian coal mining industry. *European Journal of Operational Research*, 84:134–149, 1995.

[MAE96] Sun M., Stam A., and Steuer R. E. Solving multiple objective programming problems using feed-forward artificial neural networks: The interactive FFANN procedure. *Management Science*, 42(6):835–849, 1996.

[MAE00] Sun Minghe, Stam Antonie, and Steuer Ralph E. Interactive multiple objective programming using tchebycheff programs and artificial neural networks. *Computers and Operations Research*, 27(7–8):601–620, 2000.

[MD03] Ehrgott M and Tenfelde-Podehl D. Computation of ideal and nadir values and implications for their use in mcdm methods. *European Journal of Operational Research*, 151:119–139, 2003.

[MDC98] Tamiz Mehrdad, Jones Dylan, and Romero Carlos. Goal programming for decision making: An overview of the current state-of-the-art. *European Journal of Operational Research*, 111:569–581, 1998.

[MF97] Tamiz M and Jones D F. Interactive frameworks for investigation of goal programming models: Theory and practice. *Journal of Multi–criteria Decision Analysis*, 6:52–60, 1997.

[MFC93] Sumpi J M, Amador F, and Romero C, editors. *A research on the Andalusian farmers' objectives: methodological aspects and policy implications*, VIIth EAAE congress, Stresa, Italy, 1993.

[MFC97] Sumpi J M, Amador F, and Romero C. On farmers' objectives: A multi–criteria approach. *European Journal of Operational Research*, 96:64–71, 1997.

[MKC04] Ehrgott M, Klamroth K, and Schwehn C. An MCDM approach to portfolio optimization. *European Journal of Operational Research*, 155:752–770, 2004.

[ML81] Masud A S M and Hwang C L. Interactive sequential goal programming. *Journal of the Operational Research Society*, 32:391–400, 1981.

[ML83] Gershon M and Duckstein L. Multi–objective approaches to river basin planning. *Journal of Water Resources Planning and Management*, 109:13–28, 1983.

[ML84] Gershon M and Duckstein L. A procedure for the selection of a multi–objective technique with application to water and mineral resources. *Applied Mathematics and Computation*, 14:245–271, 1984.

[ML99] Lee Sang M and Oslon David L. Goal programming. In Gal Thomas, Stewart Theodor J, and Hanne Thomas, editors, *Multicriteria Decision Making: Advances in MCDM Models, Algorithms, Theory, and Applications*, pages 8–1 – 8–33. Kluwer Acacemic Publishers, Massachusetts, 1999.

[MOP87] O'Keefe R M, Balci O, and Smith E P. Validating expert system performance. *IEEE Expert*, 2(4):81–90, 1987.

[MP56] Frank M and Wolfe P. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1):95–110, 1956.

[MP03] Makowski M and Wierzbicki A P. Modeling knowledge: Model–based decision support and soft computations. In Yu X and Kacprzyk J, editors, *Applied Decision Support with Soft Computing*, pages 3–60. Springe Verlag, Berlin, 2003.

[OA81] Balci O and Sargent A. A methodology for cost–risk analysis in the statistical validation simulation models. *Common ACM*, 24(11):190–197, 1981.

[P76] Vincke P. Une méthod interactive en programmation lenéaire à plusieurs fonctions ćonomiques. *Revue Française d'Informatique et de Recherche Opérationnelle*, 2:187–192, 1976.

[P82a] Ignizio J P, editor. *Linear programming in single and multiple objective systems*, Prentice–Hall, Englewood Cliffs, NJ, 1982.

[P82b] Wierzbicki A P. A mathematical basis for satisficing decision making. *Mathematical Modelling*, 3:391–405, 1982.

[P83] Wierzbicki P. A mathematical basis for satisficing decion making. *Mathematical Modelling*, 3:391–405, 1983.

[P89] Zaraté P. The process of designing a dss: A case study in planning management. *European Journal of Operational Research*, 1989.

[P92] Vincke P. *Multicriteria decision–aid.* John Wiley and Sons, Chichester, 1992.

[P95] Finlay P. *Introducing Decision Support Systems.* Basil Publishers, Cambridge, 1995.

[P99] Wierzbicki A P. Reference point approaches. In Gal T, Stewart T J, and Hanne T, editors, *Multicriteria Decision Making: Advances in MCDM Models algorithms theory and applications*, pages 9–1 – 9–39. Kluwer Academic Publishers, Boston, 1999.

[P6a] Ignizio J P. *Goal Programming and extensions.* Lexington Books, Lexington, 1976a.

[P6b] Ignizio J P. *Inroduction to Linear Goal Programming.* Sage, 1976b.

[PC92] Lara P and Romero C. An interactive multigoal programming model for determining livestock rations: An application to dairy cows in Andalusia, Spain. *Journal of the Operational Research Society*, 43:945–953, 1992.

[PC94] Costa J P and Clímaco J C. A multiple reference point parallel approach in MCDM. In Tzeng G H, Wang H F, Wen B P and Yu P L, editors, *Multiple Criteria Decision Making: Expand and Enrich the Domain of Thinking and Application*, pages 255–263. Springe Verlag, Berlin, 1994.

[PJ85] Korhonen P and Laakso J. A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research*, 22:277–287, 1985.

[PJ92] Korhonen P and Wallenius J. A computer graphics–based decision support system for multiple objective linear programming. *European Journal of Operational Research*, 60(3):280–286, 1992.

[PJ98] Sen P and Yang J. *Multiple Criteria Decision Support in Engineering Design.* Springler, London, 1998.

[PJ02] Hamalainen Raimo P and Mantysaari Juha. Dynamic multiobjective heating optimization. *European Journal of Operational Research*, 142:1–15, 2002.

[PM92] Wierzbicki A P and Makowski M. Multi–objective optimization in negotiation support. Working Paper WP–92–07, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1992.

[PP91] Bogetoft Peter and Pruzan Peter. *Planning with Multiple Criteria.* Elsevier Science Publishers, Armsterdam, 1991.

[PSE97] Korhonen P, Salo S, and Steuer R E. A heuristic for estimating nadir criterion values in multiple objective linear programming. *Operations Research*, 45(5):751–757, 1997.

[R85] Weistroffer H R. Careful usage of pessimistic values is needed in multiple objective optimization. *Operational Research Letters*, 1:23–25, 1985.

[R90] Schach S R. *Software Engineering.* Asken and Associates Inc., Boston, 1990.

[RE83] Steuer R and Choo E. An interactive weighted tchebycheff procedure for multiple objective progamming. *Mathematical Programming*, 26:326–344, 1983.

[RE94] Gardiner Lorraine R and Steuer Ralph E. Unified interactive multiobjective programming. *European Journal of Operational Research*, 74:391–425, 1994.

[RH03] Borges Ana Rosa and Antunes Carlos Henggeler. A weight space-based approach to fuzzy multiple-objective linear programming. *Decision Support Systems*, 34(4):427–443, 2003.

[RJJO71] Benayoun R., DeMontgolfier J., Tergny J., and Larichev O. Linear programming with multiple objective functions: Step-method (stem). *Mathematical Programming*, 1:366–375, 1971.

[RL85] Reeves G R and Franz L. A simlified interactive multiple objective linear programming procedure. *Computers and Operations Research*, 12:658–601, 1985.

[S72] Dyer James S. Interactive goal programming. *Management Science*, 19:62–70, 1972.

[S77a] Alter S. A taxonomy of decision support systems. *Sloan Management Review*, 12:39–56, 1977.

[S77b] Dyer J S. On the relationship between goal programming and multi–attribute decision theory. Discussion Paper 69, Management Science Study Center, University of California, Los Angeles, 1977.

[S77c] Zionts S. Integer linear programming with multiple objectives. *Annals of Discrete Mathematics*, 1:551–562, 1977.

[S89] Zionts S. Multiple criteria mathematical programming: an updated overview and several approaches. In Karpak B and Zionts S, editors, *MCDM and Risk Analysis*, page 6 §5. Springe Verlag, Berlin, 1989.

[S97] Licker P S. *Management Information Systems: A Strategic Leadership Approach.* Harcourt Brace College Publishers, United States, 1997.

[SA91] Shin W S and Ravandran A. Interactive Multiple Objective Optimization: Survey I – Continuous Case. *Computers and Operations Research*, 18(1):97–114, 1991.

[SCF04] Maturana S, Feffer J C, and Baranao F. Design and implementation of an optimization–based decision support system generator. *European Journal of Operational Research*, 154:170–183, 2004.

[SJ76] Zionts Stanley and Wallenius Jyrki. An interactive programming method for solving the multiple criteria problem. *Management Science*, 22:652–663, 1976.

[SJ83] Zionts Stanley and Wallenius Jyrki. An interactive multiobjective linear programming for a class of underlying utility nonlinear functions. *Management Science*, 29:519–529, 1983.

[SLM96] Chaudhry S S, Salchenberger L, and Beheshtian M. A small business inventory dSS:design, development and implementation issues. *Computers and Operations Research*, 23(1):63–72, 1996.

[SM81] Franz L S and Lee S M. A goal programming based interactive support system. In *Organizations: Multiple Agents with Multiple Criteria, Springer Lecture Notes on Economic Mathematical Systems*, pages 110–115. Springe Verlag, Berlin, 1981.

[Sta79] Zionts Stanley. Methods for solving management problems involving multiple objectives. In Fandel G and Gal T, editors, *Multiple Criteria Decision Making Theory and Application*, pages 540–558. Spinger-Verlag, Berlin, 1979.

[T97] Buchanan J T. A naive approach for solving MCDM: The GUESS method. *Journal of the Operational Research Society*, 48:202–206, 1997.

[TJT99] Gal Thomas, Stewart Theodor J, and Hanne Thomas. *Multicriteria Decision Making: Advances in MCDM Models Algorithms Theory and Applications.* Kluwer Acacemic Publishers, Massachusetts, 1999.

[TL95] Buchanan J T and Corner J L. Experimental consideration of preference in decision making process. *Journal of Multi–criteria Decision Analysis*, 4:107–121, 1995.

[TTS90] Trafalis T., Morin T.L., and Abhyankar S.S. Efficient faces of polytopes : Interior point algorithms, parameterization of algebraic varieties, and multiple objective optimization. *Contemporary Mathematics*, 114:319–341, 1990.

[TW02] Blake John T and Carter Mechael W. A goal programming approach to strategic resource allocation in acute care hospitals. *European Journal of Operational Research*, 140:541–561, 2002.

[VJ02] Belton V and Stewart T J. *Multiple Criteria Decision Analysis: An Integrated Approach.* Kluwer Academic Publishers, Boston, 2002.

[VTS92] Lotfi V, Stewart T.J., and Zionts S. An aspiration–level interactive model for multiple criteria decision making. *Computers and Operations Research*, 19(7):671–681, 1992.

[vWDE86] von Winterfeldt D and Edwards E. *Decision Analysis and Behavioral Research.* Cambridge University Press, New York, 1986.

[VYS97] Lotfi V, Yoon YS, and Zionts S. Aspiration–based search algorithm for multiple objective linear programming problems: Theory and comparative tests. *Management Science*, 43(8):1047–1059, 1997.

[W77] Edwards W. How to use multiattribute utility measurement for social decision making. *Man and Cybernetics*, 7:326–340, 1977.

[W81] Keen P G W. *Decision Support Systems: A Researh Perspective in Decision Support Systems: Issues and Challenges.* Pergamon Press, 1981.

[W84] Evans G W. An overview of techniques for solving multiojective mathematical programs. *Management Science*, 30(11):1268–1282, 1984.

[W92] Ozernoy W. Choosing the "best" multiple criteria decision–making method. *INFOR: Canadian Journal of Operational Research*, 30:159–171, 1992.

[WSB93] Holsapple C W, Park S, and Whinston A B. Framework for dss interface development. In Holsapple C and Whinston C, editors, *Recent Developments in Decision Support Systems.* Springer Verlag, Berlin, 1993.

[Y65] Ijiri Y. *Management goals and Accounting for Control.* Rand–McNally, Chicago, 1965.

[YTE96] Aksoy Y, Butler T, and Minor E. Comparative studies in interactive multiple objective mathematical programming. *European Journal of Operational Research*, 89:408–422, 1996.