

https://dspace.org/

A comparative study of interactive algorithms for multiple objective linear programming

Benvenuti, Fabrizio Roberto

2016-07-18
http://hdl.handle.net/10539/20654

Downloaded from a DSpace repository. https://dspace.org/

A Comparative Study of Interactive Algorithms for Multiple Objective Linear Programming

Fabrizio Roberto Benvenuti

May 1992

A research report submitted to the Faculty of Science, University of the Witwatersrand, in partial fulfilment of the degree of Master of Science.

Abstract

In the last two decades, a vast number of methods for solving decision making problems with multiple objectives have been proposed, leaving little time for algorithm comparison and evaluation. The aim of this research report is to contribute to this neglected area by comparing and evaluating a few interactive algorithms for multiple objective linear programming.

A brief overview of multiple objective linear programming (MOLP) is presented, followed by a literature survey in which the important interactive methods for solving MOLP problems are described. Seven of these algorithms are selected for further investigation. These are described in more detail, broken down to their constituent steps and compared. The performance of the algorithms on a number of test problems is investigated, and the results of an experiment which was conducted in order to ascertain decision makers' impressions of the algorithms are presented. Finally, an attempt is made to draw some conclusions.

Declaration

I declare that this research report is my own unaided work. It is being submitted for the degree of Master of Science in the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination in any other University.

J.R. Beneduki

Thirteenth day of April 1992

Acknowledgements

I would like to express my deep gratitude to my supervisor, Dr Kendal Jordi, for his invaluable assistance and insight during the course of my research.

Thanks are also due to Professors Theo Stewart (UCT), Stanley Zionts (Buffalo), Elmér E Kosinger (UP), and Mr Hans Ittman (CSIR) for providing me with information relevant to my research, as well as Pietre Du Toit (Wits) and Chris Gilfillan (CSIR) for their advice on the statistical aspects of the research.

I would also like to thank the Foundation for Research Development for partially funding this research.

Finally, thanks to Susan Syndercombe and Mari Benvenuti for proofreading and general assistance.

Declaration

The use of the masculine tense throughout this document is simply a matter of literary convenience.

Contents

1 Introduction 1

1.1 Scope of this Study 3

1.2 Preliminaries 7

1.3 The Roots of MOLP 12

2 Literature Survey 16

2.1 The Geoffrion-Dyer-Feinberg Method and its Variants 18

2.2 The Reference Point Method and its Variants 27

2.3 The Visual Interactive Method and its Variants 32

2.4 Steuer's Interval Criterion Weights Method and its Variants . . 35

2.5 The Method of Zionts and Wallenius and its Variants 40

2.6 The Method of Musselman and Talavage and its Variants 45

2.7 The STEM Method and its Variants 51

2.8 The Surrogate Worth Tradeoff Method and its Variants 56

2.9 The Branch-and-Bound Method and its Variants 63

2.10 The Interactive Relaxation Method and its Variants 67

2.11 SEMOPS and its Variants 71

2.12 Other Approaches 76

3 Further Investigation of Selected Algorithms 82

3.1 Sample Algorithms 86

3.2 Tradeoff Algorithms 92

3.3 Projection Algorithms 99

4 Implementation Issues 103

4.1 Linear Programming 105

4.2 Testing for Tradeoff Efficiency 106

4.3 Parametric Programming 108

4.4 Scaling the Objective Functions 111

4.5 Generating Weights from Λ 113

4.6 Filtering 120

4.7 Contracting the Criterion Cone 138

5 Algorithm Breakdown 140

5.1 Generating Weights 142

5.2 Generating Solutions 147

5.3 Imposing Additional Constraints 152

5.4 Stopping Conditions 156

5.5 Method of Interaction with the DM 159

5.6 Conclusion 162

6 Algorithm Comparison and Evaluation 164

6.1 Analytic Comparison 165

6.2 Computational Comparison 168

6.3 An Evaluation of the Algorithms from a Decision Maker's Perspective 178

6.4 Conclusion 193

7 Conclusion 194

A Algorithm Questionnaire 198

B Algorithm Rating Form 201

C Selected Decision Makers' Comments 203

References 208

List of Figures

1.1 Multiple Criteria Decision Making (with only one DM) 5

4.1 A Criterion Cone 113

4.2 Applying Predetermined Convex Combinations 114

4.3 Convex Combinations Drawn from the Uniform Distribution . . 115

4.4 Convex Combinations Drawn from the Weibull Distribution . . 116

4.5 Convex Combinations — 50% from Uniform Distribution and 50% from Weibull Distribution 117

4.6 Convex Combinations — 30% from Uniform Distribution and 70% from Weibull Distribution 117

4.7 Convex Combinations using the Grid Method 118

4.8 Points to be Filtered 122

4.9 The 15 Points Returned by the Threshold Algorithm 129

4.10 The 15 Points Returned by the Closest Point Outside the Neighbourhood Algorithm 129

4.11 The 15 Points Returned by the Furthest Point Outside the Neighbourhood Algorithm 130

4.12 The 15 Points Returned by the Maximin Algorithm 130

4.13 The 15 Points Returned by the K-means Algorithm 132

4.14 The C-cone 134

4.15 Reduced D-cone 135

4.16 Reduced E-cone 136

4.17 Points Generated by Steuer's Weighting Vectors 138

4.18 Reduced E-cone about f^3 138

4.19 Reduced E-cone about f^4 139

4.20 Reduced E-cone about f^7 139

5.1 The Weights which Satisfy the DM's Constraints 144

5.2 Convex Combinations when $\lambda_i \in [0, 1]\ \forall i$ 153

5.3 Convex Combinations when $\lambda_i \in [0.05, 0.95]\ \forall i$ 153

6.1 Constraint Set for Example 6.4 171

6.2 Constraint Set for Example 6.5 173

6.3 Distribution of Algorithm Completion Times 183

6.4 Distribution of Scores 184

6.5 Distribution of Scores with Odd and Even Scores Combined . . 185

List of Tables

5.1 Algorithm Breakdown 163

6.1 How the Algorithms Match Up 167

6.2 Utility Function Value of Each Iterate for Example 6.3 170

6.3 Utility Function Value of Each Iterate for Example 6.4 172

6.4 Utility Function Value of Each Iterate for Example 6.5 174

6.5 Utility Function Value of Each Iterate for Example 6.6 177

6.6 Nutrition and Cost Information on Six Foods 180

Chapter 1

Introduction

Decision making is the process of selecting a possible course of action from all the available alternatives. Methods for making such a decision when only one criterion is used to judge the alternatives have been studied extensively for the last 50 years. In the last two decades however, there has been an increased awareness of the need to evaluate solution alternatives according to multiple criteria. Accordingly, a vast number of methods for solving decision making problems with multiple objectives have been proposed in the last 20 years. This proliferation of methods in a very short period has left little time for the comparison and evaluation of the algorithms. MacCrimmon [34] points out that this comparison is essential so that a combination of methods, which is usually more effective than a single technique, can be used to solve a problem.

The aim of this research report is to contribute to this neglected area by comparing and evaluating a few algorithms. The algorithms investigated all fall into the category of interactive multiple objective linear programming.

The structure of this report is as follows.

Our area of interest is defined and a number of definitions and assumptions used in this study are given. This is followed by a brief historical review of multiple objective linear programming. A literature survey in which the im-

portant interactive methods for solving multiple objective linear programming problems are described is presented next. From these algorithms, seven are selected for further investigation and described in more detail. These seven are

1. Steuer's Contracting Cones
2. The Interactive Weighted Sums/Filtering Approach
3. The Interactive Weighted Tchebbycheff Method
4. The Method of Zionts and Wallenius
5. A Cutting Plane method
6. The Reference Point Method
7. The Visual Interactive Method

The reasons for selecting these specific algorithms are given in at the beginning of chapter 3.

All the algorithms contain a number of steps which are essentially independent of the algorithm, and are common to more than one algorithm. Various ways of implementing these steps are described. The algorithms are then broken down to their constituent parts and compared.

The algorithms were implemented on a computer and their performance was evaluated on a number of test problems. The computer programs were also tested on decision makers in order to evaluate the algorithms from a decision maker's perspective. The results of this computational testing are presented next.

Finally, an attempt is made to draw some conclusions.

1.1 Scope of this Study

As mentioned above, Multiple Criteria Decision Making (MCDM) concerns itself with the making of decisions when more than one criterion is used to judge the various alternatives. The area of MCDM is extremely vast and the problems falling under it are distinguished by various factors, such as whether the decision is being made by one person, or a group of persons, or whether all information is known with certainty, or there is a degree of uncertainty. Clearly, in order to do a comparitive study, algorithms which solve similar problems must be selected for comparison. The class of problems selected for comparison is Multiple Objective Linear Programming. In order to identify where this fits in in MCDM, we take a look at MCDM and classify the various areas which fall under it.

First of all we consider only MCDM where there is only one decision maker.

We now split MCDM problems according to whether the set of alternatives is continuous or discrete. Gershon and Duckstein [16] call this classification criterion "marginal versus non-marginal difference between alternatives" and state that if marginal differences between alternatives are being considered, then the problem is characterized by

1. An infinite, or very large, set of alternatives
2. A set of constraints
3. An objective function vector
4. An algorithm that generates successive preference points until a "satisfactum" is found.

On the other hand if only non-marginal differences between alternatives are possible, the problem possesses

1. A discrete (usually small) set of available alternatives with given attributes

2. An ordinal scaling of the attribute values
3. A set of constraints across attributes
4. A procedure for sequential elimination of alternatives on the basis of attribute values

We now distinguish between deterministic and stochastic problems. A deterministic problem is one which is fully determined and in which probability plays no role. A stochastic problem is one in which we have probabilities of certain outcomes occurring.

A third factor for classification is whether all objective functions and constraints are linear, or one or more objectives or constraints is nonlinear.

In this study, we restrict ourselves to deterministic, continuous, linear multiple objective optimization problems, where there is only one decision maker (hereafter abbreviated as MOLP).

As can be seen from figure 1.1, the area being tackled (shaded area) is a very small subset of the broad area of MCDM.

In addition we restrict ourselves further, to cover only a subset of MOLP.

Quite naturally, there are many different ways of classifying MOLP algorithms. The classification which seems to have been the most widely accepted and used is that proposed by Hwang and Masud [22], in which algorithms are defined according to the time of the DM's articulation of preference information.

The classes of MOLP algorithms are

1. Methods with a priori articulation of preferences, ie. the DM gives preference information before the optimization is done
2. Methods with progressive articulation of preferences, i.e. the DM gives preference information while the optimization is being done

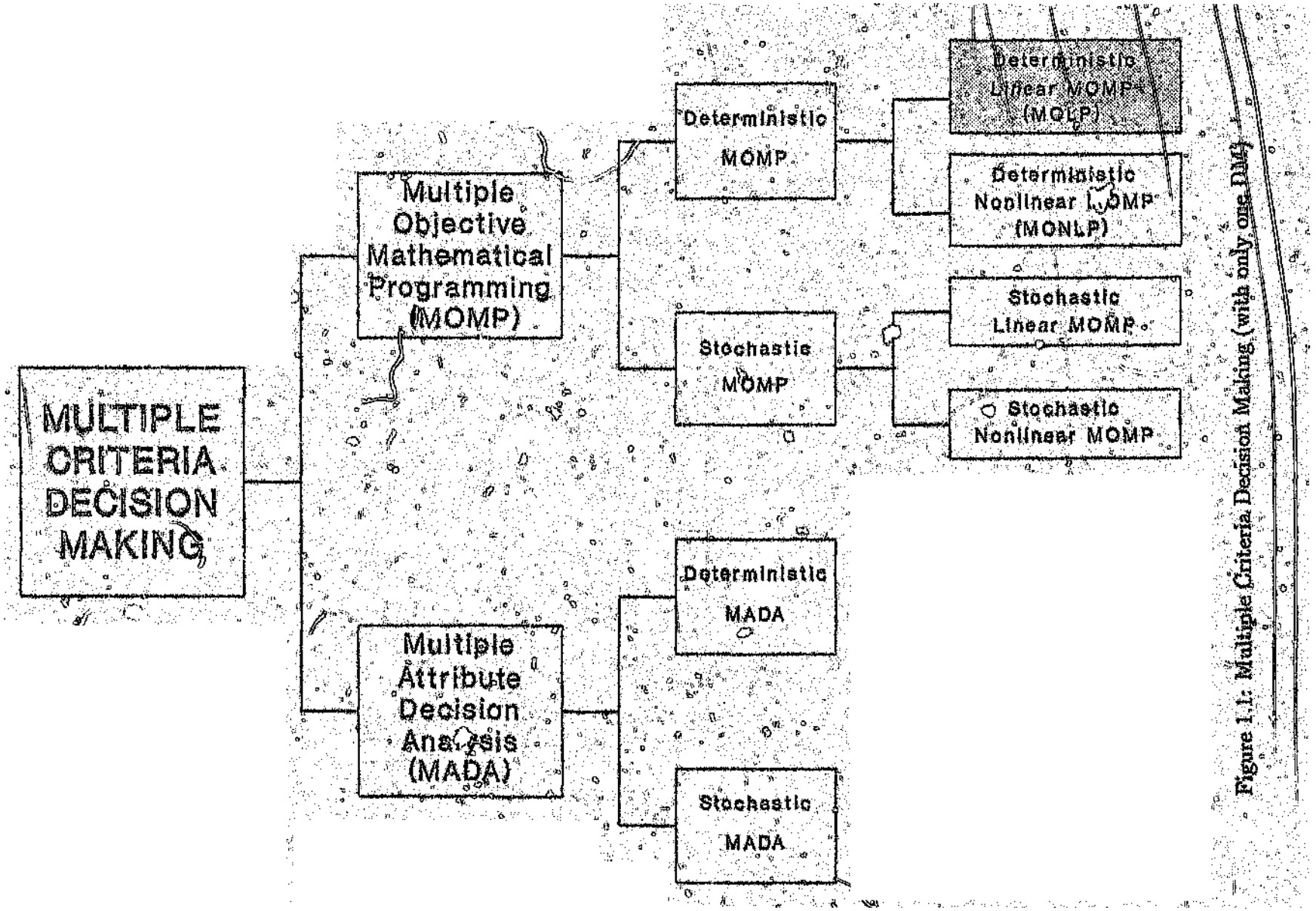


Figure 1.1: Multiple Criteria Decision Making (with only one DM)

Figure 1.1: Multiple Criteria Decision Making (with only one DM)

3. Methods with a posteriori articulation of preferences, i.e. the DM gives preference information after the optimization has been done

The rest of this research report is devoted mainly to algorithms which use a progressive articulation of the DM's preference information. These are also known as *interactive* algorithms.

We now look at the mathematical definition of MOLP, together with some allied definitions, assumptions, and notes which are required in this study.

1.2 Preliminaries

Definition 1.1 A *single objective constrained optimization* problem is of the form

$$\begin{aligned} \max \quad & z = f(\underline{x}) \\ \text{subject to} \quad & \underline{x} \in X \end{aligned}$$

where $\underline{x} \in \Re^n$, $f(\underline{x})$ is the objective function and X is the feasible region.

The purpose of the optimization problem is to find the point in X that has the highest objective function value z.

Definition 1.2 A *multiple objective constrained optimization* problem is of the form

$$\begin{aligned} \max \quad & \underline{z} = \underline{f}(\underline{x}) = [f_1(\underline{x}), f_2(\underline{x}), \cdots, f_p(\underline{x})] \\ \text{subject to} \quad & \underline{x} \in X \end{aligned}$$

This is also sometimes referred to as a vector optimization problem.

Definition 1.3 A *single objective linear programming* (SOLP or LP) problem is a single objective optimization problem where $f(\underline{x})$ is linear and X is defined by linear constraints. i.e. a problem of the form

$$\begin{aligned} \max \quad & z = \underline{c}^T \underline{x} \\ \text{subject to} \quad & A\underline{x} \leq \underline{b} \\ & x_i \geq 0 \quad \forall i = 1 \ldots n \end{aligned}$$

where $\underline{c}$ is $n \times 1$, A is $m \times n$ and $\underline{b}$ is $m \times 1$. A is assumed to be of full row rank.

Definition 1.4 A *multiple objective linear programming* (MOLP) problem is a multiple objective optimization problem where $f(\underline{x})$ is linear and X is defined by linear constraints. i.e. a problem of the form

$$\begin{aligned} \max \quad & \underline{z} = C\underline{x} \\ \text{subject to} \quad & A\underline{x} \leq \underline{b} \\ & x_i \geq 0 \quad \forall i = 1 \ldots n \end{aligned}$$

where C is a $p \times n$ matrix whose rows consist of the coefficients of the p objective functions, A is $m \times n$ and $\underline{b}$ is $m \times 1$. A is assumed to be of full row rank.

Definition 1.5 A *discrete optimization* problem is one in which the set X consists of a finite number of alternatives.

Definition 1.6 A *continuous optimization* problem is one in which the set X consists of an infinite number of alternatives which are inferable from constraints on $\underline{x}$.

Definition 1.7 An *ideal solution* $\underline{z}^*$ is the point in criterion space $(z_1^*, z_2^*, \cdots, z_p^*)$ such that z_i^* is the optimal objective function value for the problem

$$\begin{aligned} &\max && z_i = f_i(\underline{x}) \\ &\text{subject to} && \underline{x} \in X \end{aligned}$$

Definition 1.8 An *anti-ideal solution* $\underline{z}^\circ$ is the point in criterion space $(z_1^\circ, z_2^\circ, \cdots, z_p^\circ)$ such that z_i° is the optimal objective function value for the problem

$$\begin{aligned} &\min && z_i = f_i(\underline{x}) \\ &\text{subject to} && \underline{x} \in X \end{aligned}$$

Definition 1.9 A solution $\underline{x}^{**} \in X$ which simultaneously maximizes each of the p objective functions of a multiple objective optimization problem is called a *superior solution*.

Definition 1.10 A solution $\underline{x} \in X$ is said to be *dominated* if there is a point $\underline{y} \in X$ such that

$$\begin{aligned} &f_i(\underline{x}) \leq f_i(\underline{y}) \quad \forall i = 1 \ldots p \\ &\text{and } f_k(\underline{x}) < f_k(\underline{y}) \quad \text{for at least one } k,\ 1 \leq k \leq p \end{aligned}$$

Definition 1.11 A solution $\underline{x} \in X$ which is not dominated is said to be *non-dominated*. Nondominated solutions are sometimes also referred to as *efficient*, *non-inferior* or *Pareto-optimal*.

Definition 1.12 The person who is responsible for choosing one alternative out of the possible alternatives is called the *decision maker* (DM).

Assumption 1.13 Any two points $\underline{x}$ and $\underline{y}$ in the set X are comparable in the sense that one, and only one, of the following holds:

(i) the DM prefers $f(\underline{x})$ to $f(\underline{y})$ $\quad (f(\underline{x}) \succ f(\underline{y}))$
(ii) the DM prefers $f(\underline{y})$ to $f(\underline{x})$ $\quad (f(\underline{x}) \prec f(\underline{y}))$
(iii) the DM is indifferent between $f(\underline{x})$ and $f(\underline{y})$ $\quad (f(\underline{x}) \sim f(\underline{y}))$

Assumption 1.14 Writing $f(\underline{x}) \succeq f(\underline{y})$ to mean " not $(f(\underline{x}) \prec f(\underline{y}))$ ", we assume that all the relations $\sim, \succ, \succeq$ are transitive.

Note 1.15 A preference structure is defined on the objective space in $\Re^p$ if any two points $f(\underline{x})$ and $f(\underline{y})$ in $\Re^p$ (where $\underline{x}, \underline{y} \in X$) are comparable and there are no intransitivities (Keeney and Raiffa [25] pp. 79-80).

Definition 1.16 Assume that there exists some function $U(\underline{f}(\underline{x})) : \Re^p \to \Re^1$ such that

$U(\underline{f}(\underline{x})) > U(\underline{f}(\underline{y}))$ iff the DM prefers $f(\underline{x})$ to $f(\underline{y})$
$U(\underline{f}(\underline{x})) < U(\underline{f}(\underline{y}))$ iff the DM prefers $f(\underline{y})$ to $f(\underline{x})$
$U(\underline{f}(\underline{x})) = U(\underline{f}(\underline{y}))$ iff the DM is indifferent between $f(\underline{x})$ and $f(\underline{y})$

Then by assumptions 1.13 and 1.14, a preference structure is defined on the objective space. The function $U(\underline{f}(\underline{x}))$ is said to be a *utility function* representing the DM's preference structure.

Note 1.17 If the DM's utility function exists and is explicitly defined, then the multiple objective optimization problem (defn. 1.2) can be expressed as

$$\begin{aligned} \max \quad & z = U(\underline{f}(\underline{x})) \\ \text{subject to} \quad & \underline{x} \in X \end{aligned}$$

U is generally not explicitly defined. Were U explicitly defined, the problem would reduce to a single objective nonlinear programming problem.

U is generally assumed from microeconomic theory to be concave and twice continuously differentiable.

Definition 1.18 $\nabla U(\underline{f}(\underline{x}))$ is called the *marginal utility vector*.

Definition 1.19 The *marginal rate of substitution* (MRS) *of objective i for objective j is* $\dfrac{\nabla U(\underline{f}(\underline{x}))}{\nabla f_j} / \dfrac{\nabla U(\underline{f}(\underline{x}))}{\nabla f_i}$ and is denoted λ_{ij}.

Definition 1.20 A solution (not necessarily optimal) which the DM considers satisfactory is referred to as a *satisficing solution* or *satisfactum*.

Definition 1.21 The *best compromise solution* is the nondominated solution which best satisfies the DM's objectives. Although the term is used interchangeably with the term "optimal solution", it is a more accurate description when dealing with problems with multiple objectives.

Definition 1.22 An *indifference band* is defined to be a subset of the nondominated set where the improvement of one or more objective functions is equivalent (in the mind of the DM) to the degradation of others. In other words $U(\underline{f}(\underline{x}))$ is constant throughout an indifference band.

Notation 1.23 $\underline{0} = (0, 0, 0, \ldots, 0)^T$,
$\underline{1} = (1, 1, 1, \ldots, 1)^T$

Definition 1.24 The *L_ρ norm* of $\underline{x} \in \Re^n$ is given by

$$\|\underline{x}\|_\rho = [\sum_{i=1}^{n} |x_i|^\rho]^{\frac{1}{\rho}}$$

for some ρ. The most common values of ρ used are $\rho = 1, 2$ and ∞.

Definition 1.25 The family of *weighted L_ρ norms* of $\underset{\sim}{x} \in \Re^n$ is given by

$$\|\underset{\sim}{x}\|_\rho^w = \left[\sum_{i=1}^n w_i |x_i|^\rho\right]^{\frac{1}{\rho}}$$

where w_i is the weight associated with dimension i and ρ defines the norm. $w_i \geq 0 \; \forall i = 1, \ldots n, \sum_{i=1}^n w_i = 1$.

Definition 1.26 The family of weighted L_∞ norms for measuring the distance between $\underset{\sim}{z} \in \Re^n$, and the ideal criterion vector $\underset{\sim}{z}^*$ (i.e. $\|\underset{\sim}{z}^* - \underset{\sim}{z}\|_\infty^w$) is also called the *weighted Tchebbycheff metric*.

Definition 1.27 The family of *augmented weighted Tchebbycheff metrics* measuring the distance between $\underset{\sim}{z} \in \Re^n$, and $\underset{\sim}{z}^*$ is given by

$$\|\underset{\sim}{z}^* - \underset{\sim}{z}\|_\infty^w + \epsilon \sum_{i=1}^n |z_i^* - z_i|$$

ϵ positive and small.

1.3 The Roots of MOLP

A comprehensive review of MOLP algorithms is presented in chapter 2. First, however, we present four methods which Zionts [67] refers to as "naïve" methods. These are simple in concept and generally not very good, but most current algorithms for MOLP have evolved directly from these naïve methods. These methods are presented here to establish the roots of MOLP.

1.3.1 Setting Minimum Levels for All Objective Functions

The first of the naïve methods to be considered is that of specifying or setting levels of all objectives, and then solving for a feasible solution. There are three possible outcomes to this problem

(a) No feasible solution

(b) A dominated solution

(c) A nondominated solution

Almost certainly, either (a) or (b) will occur. Only in rare circumstances will (c) occur.

This is quite clearly a method with a priori articulation of preferences.

1.3.2 Setting Minimum Levels for All but One Objective Function

A second naïve approach is to set levels for all but one objective and to maximize the remaining objective. Although this method is an improvement over the previous one since it always leads to a nondominated solution, its downfall is that only one objective function is taken into account. If there are infinitely many solutions which yield the same value for the objective function being maximized, it is unrealistic to expect that the DM will like all such solutions equally; the values of the other objective functions are likely to be crucial in his selection of a solution.

This method is also quite clearly a method with a priori artisulation of preferences.

1.3.3 Estimating the DM's Utility Function

Another naïve method's approach involves constructing a composite objective function as a function of the individual objective functions, and maximizing this composite (utility) function. Jordi [23] lists a number of references which report that the determination of the utility function is difficult for various reasons.

Consider the simplest possible case, viz. approximating the utility function by a linear weighted sum, i.e. $U(\underline{f}(\underline{x})) \approx \sum_{i=1}^{p} \lambda_i f_i(\underline{x})$. In this case, which has been applied extensively in MOLP due to its computational simplicity, the DM is asked to express his preferences in terms of weights $\lambda_i > 0, \sum \lambda_i = 1$, such that the single objective LP problem

$$\begin{aligned} \max \quad & z = \underline{\lambda}^T C \underline{x} \\ \text{subject to} \quad & A\underline{x} \leq b \\ & x_i \geq 0 \quad \forall i = 1 \ldots n \end{aligned} \tag{f.1}$$

will produce the "most desirable" extreme point $\underline{x}^*$.

Since there exists a set of weighting vectors

$$\Lambda^* = \{\lambda \in \mathbb{R}^p | \lambda^T C x^* = \max[\lambda^T C x | Ax \leq b]\},$$

the DM's "most desirable" weighting vector is not unique. Any member of Λ^* will suffice. But because Λ^* is in general very small in relation to the entire weighting space Λ, determining a member of Λ^* can be difficult. Steuer [51] states that "the ability of the DM to ascertain correctly a member of Λ^* is not considered very good even when using a disciplined procedure".

Steuer [56] pp. 193-200 discusses a number of problems associated with the weighted sums approach. These include :- the dependence of the weights on the feasible set, the dependence of the weights on the relative lengths of the objective function gradients, and difficulties caused by the degree to which objectives are correlated. Sensitivity of a solution to the weighted sums problem (1.1) is a fundamental issue in using the weighted sums approach. Hansen *et al.* [20] have derived an approach which allows us to calculate a maximum tolerance percentage by which weights may deviate simultaneously and independently from their estimated values, yielding the same optimal basic solution to (1.1).

Of course a linear approximation to the utility function may be completely inadequate and the weighted sums approach may not be applicable.

The method of estimating the DM's utility function is quite clearly a method with a priori articulation of preferences.

1.3.4 Finding All Nondominated Extreme Points

Generating all the nondominated extreme points and presenting the list to the DM has been widely proposed. In fact, methods in this class were among the first multiobjective solution procedures developed.

The problem of generating all nondominated extreme points, which is also known as the vector maximum problem, is equivalent to solving the family of

weighted sum problems

$$\begin{aligned} \max \quad & z = \underset{\sim}{\lambda}^T C \underset{\sim}{x} \\ \text{subject to} \quad & A\underset{\sim}{x} \leq b \\ & x_i \geq 0 \quad \forall i = 1 \ldots n \end{aligned}$$

for all $\underset{\sim}{\lambda} \in \Lambda$ where $\Lambda = \{\underset{\sim}{\lambda} \in \Re^p | \lambda_i \in [0,1], \sum_{i=1}^{p} \lambda_i = 1\}$.

Once all the nondominated extreme points have been generated, the DM selects one of the solutions from the list, either by just looking at them, or by using some multiple attribute decision making technique. The problem with the approach is that, even for a relatively small problem, the number of nondominated extreme points can be large. The consequences of this are twofold; firstly computation time is long, and secondly, such large lists of alternatives would most likely inhibit the ability of the DM to make a selection. Zionts [67] comments that only in the early 70's was this considered a viable solution procedure.

In contrast to the weighted sums model in which only one weighting vector is specified at a time, the above method uses all possible weighting vectors simultaneously. In this sense the two techniques can be considered polar extremes. Whereas a criticism of the weighted sums technique might be that it generates too few alternatives for the DM, the vector maximum approach suffers from the opposite malady of generating too much information. There is hence a need for techniques which generate a more moderate amount of information. It is towards these goals that the weighted sums and vector maximum approaches have been combined in sample algorithms as discussed in section 3.1.

Steuer [51] lists numerous methods developed for finding all the nondominated extreme points, while a detailed discussion of the topic can be found in Goicoechea *et al* [17] pp. 40-91.

The vector maximum approach utilizes a posteriori articulation of preferences.

Chapter 2

Literature Survey

Since the early seventies, the number of publications in the area of MOLP has proliferated at a rapid rate. Zeleny ([66], pp. 60-61) states that "MCDM was unquestionably the fastest growing and most innovative OR/MS field of the seventies".

A number of reviews and state of the art surveys [7, 6] have been published, each classifying the algorithms in their own way. In an attempt to bring some order to the field, Shin and Ravindran [49] presented a survey of interactive algorithms for MOLP, introducing a new classification scheme. They classify algorithms into the following categories.

1. Feasible Region Reduction Methods
2. Feasible Direction Methods
3. Criterion Weight Space Methods
4. Tradeoff Cutting Plane Methods
5. Lagrange Multiplier Methods
6. Visual Interactive Methods
7. Branch-and-Bound Methods

8. Other Important Methods

(a) Relaxation Methods

(b) Sequential Methods

(c) Scalarizing Function Methods

In general, each class consists of a pioneer method and several variants. One notable exception is the class of criterion weights space methods where two quite different pioneer methods have been developed and variations proposed.

This literature survey is presented according to the above classification; the primary method in each category is presented, together with some of the more important variations.

Sakawa *et al.* [45] comment that three different approaches can be used to solve MOLP problems, viz. goal programming, interactive methods, and fuzzy programming. Although our area of interest is interactive algorithms, a brief description of the goal programming and fuzzy programming approaches is also given in order to complete the picture.

This survey is not in any way intended to be a comprehensive review of all major works and contributions in the area of MOLP, but rather an effort to present the various approaches and directions which algorithms, and MOLP research in general, is taking.

2.1 The Geoffrion-Dyer-Feinberg Method and its Variants

The Geoffrion-Dyer-Feinberg Method [15] is the pioneer method in the class of methods which Shin and Ravindran [49] classify as feasible direction methods. The methods which fall into this class all are direct extensions of the feasible direction methods developed for solving single objective nonlinear programming problems. These algorithms generally start with a feasible solution and iteratively perform two major steps:

1. Find a direction along which the DM's preference appears to increase, called the *direction finding problem*
2. Determine how far to travel in this direction from the current solution, called the *step-size problem*

Recall from note 1.17 that if we assume that the decision maker's utility function exists, the problem we are trying to solve is essentially

$$\begin{aligned} \max \quad & z = U(\underline{f}(\underline{x})) \\ \text{subject to} \quad & \underline{x} \in X \end{aligned}$$

Geoffrion et al. [15] use the Frank-Wolfe method to solve this nonlinear problem. This results in the following algorithm (ignoring for the moment the fact that U is not explicitly known).

1. Find an initial feasible solution $\underline{x}^0$. Set $h = 0$.
2. Solve the direction finding problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} f_i(\underline{x}) \nabla U(\underline{f}(\underline{x}^{(h)})) \\ \text{subject to} \quad & \underline{x} \in X \end{aligned}$$

Call the solution $\underline{y}^{(h)}$. Then the steepest ascent direction from $\underline{x}^{(h)}$ is given by $\underline{y}^{(h)} - \underline{x}^{(h)}$.

3. Solve the step-size problem

$$\max_{\rho \geq 1} U(\underline{f}(\underline{x}^{(k)} + \rho(\underline{y}^{(k)} - \underline{x}^{(k)})))$$

Denote the optimal value of ρ as $\rho^{(k)}$.
Set $\underline{x}^{(k+1)} = \underline{x}^{(k)} + \rho^{(k)}(\underline{y}^{(k)} - \underline{x}^{(k)})$.

4. If $\underline{x}^{(k+1)} = \underline{x}^{(k)}$, stop, $\underline{x}^{(k)}$ is optimal. Otherwise set $k = k+1$ and go to step 2.

The strength of this approach is that the DM is not required to have explicit knowledge of U. Instead only local knowledge of U is required, namely $\nabla U(\underline{f}(\underline{x}^{(k)}))$ and the value of U along a certain direction.

Since U is not explicitly known, we cannot find ∇U. However, it can be shown that

$$\nabla U(\underline{f}(\underline{x}^{(k)})) = \sum_{i=1}^{p} \frac{\partial U(\underline{f}(\underline{x}^{(k)}))}{\partial f_i} \nabla f_i(\underline{x}^{(k)})$$

The solution of the direction finding problem in step 2 above will not be affected if the objective function is multiplied by a scalar. So if the objective function in the direction finding problem is divided through by $\frac{\partial U(\underline{f}(\underline{x}^{(k)}))}{\partial f_1} / \frac{\partial U(\underline{f}(\underline{x}^{(k)}))}{\partial f_1}$, where f_1 was selected arbitrarily, then the direction finding problem becomes

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} f_i(\underline{x}^{(k)}) m_{1i}(\underline{x}^{(k)}) \\ \text{subject to } & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

where $m_{1i}(\underline{x}^{(k)}) = \frac{\partial U(\underline{f}(\underline{x}^{(k)}))}{\partial f_i} / \frac{\partial U(\underline{f}(\underline{x}^{(k)}))}{\partial f_1}$

These "weights" $m_{1i}(\underline{x}^{(k)})$ reflect the DM's tradeoff between f_1 and f_i at the current point $\underline{x}^{(k)}$ and must be obtained before the problem is solved.

The next step in the Geoffrion-Dyer-Feinberg algorithm requires the solution of the line search. Since $U(\underline{f}(\underline{x}^{(k)}))$ is not explicitly known, this cannot be

done by direct solution. The solution can be realised by presenting a number of vectors $\underset{\sim}{f}(\underset{\sim}{x}^{(h)} + \rho \underset{\sim}{q}^{(h)})$ for various values of ρ, $0 \leq \rho \leq 1$. The vector of objectives $\underset{\sim}{f}$ that is preferred by the DM defines the optimal step size $\rho^{(h)}$.

The complete Geoffrion-Dyer-Feinberg algorithm is thus :-

0. Choose a starting set of weights $\underset{\sim}{\lambda}^{(0)}$. Set $h = 0$. ($\underset{\sim}{\lambda}^{(0)}$ is normally selected to be $\lambda_i^{(0)} = \frac{1}{p}$, for $i = 1 \ldots p$). Solve the linear programming problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i^{(0)} f_i(\underset{\sim}{x}) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

1. Obtain the preference weights from the DM either directly or by obtaining the MRS using the following relation

$$\frac{1}{m_{1i}(\underset{\sim}{x}^{(h)})} = \frac{\partial U(\underset{\sim}{f}(\underset{\sim}{x}^{(h)}))}{\partial f_1} \Big/ \frac{\partial U(\underset{\sim}{f}(\underset{\sim}{x}^{(h)}))}{\partial f_i} \approx \frac{\Delta f_i(\underset{\sim}{x}^{(h)})}{\Delta f_1(\underset{\sim}{x}^{(h)})}$$

2. Solve the linear programming problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i^{(h)} f_i(\underset{\sim}{x}) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

where $\lambda_i^{(h)} = m_{1i}(\underset{\sim}{x}^{(h)})$.
Call the solution to this problem $\underset{\sim}{y}^{(h)}$ and let $\underset{\sim}{q}^{(h)} = \underset{\sim}{y}^{(h)} - \underset{\sim}{x}^{(h)}$.

3. If $\underset{\sim}{y}^{(h)} = \underset{\sim}{x}^{(h)}$ then stop, $\underset{\sim}{x}^{(h)}$ is optimal.

4. Otherwise display the q criterion vectors $\underset{\sim}{f}(\underset{\sim}{x}^{(h)} + \frac{k-1}{q-1}\underset{\sim}{q}^{(h)})$, $j = 1 \ldots q$.

5. Obtain the DM's most preferred of the q displayed criterion vectors, criterion vector k say. Then let $\underset{\sim}{x}^{(h+1)} = \underset{\sim}{x}^{(h)} + \frac{k-1}{q-1}\underset{\sim}{q}^{(h)}$. Set $h = h + 1$ and go to 1.

2.1.1 Comments

1. Although this method assumes the existence of an underlying utility function, it does not require this utility function to be identified explicitly.

2. In practice, the DM will emerge from step 1 with only approximations of his true weights, and from step 5 with only approximations of the optimal stepsize. However, Steuer [56] page 379 states that "there are results showing that infinite convergence to the optimum holds, provided the approximations become sufficiently more exact as the iteration count increases".

3. It is quite likely that many of the criterion vectors presented to the DM in step 4 will be dominated.

4. The only quantities that need to be determined are the indifference tradeoffs or MRS. These have to be 'extracted' from the DM, who in principle has to estimate the tradeoffs subjectively. In practice, it is found (Wallenius [60]) that the DM has difficulty accomplishing such a task unaided.

2.1.2 Variations

It order to overcome the problem DMs' have in estimating MRS unaided, Dyer [12] developed an algorithmic procedure for approximating the MRS through a series of ordinal paired comparisons, where the DM is required to indicate a preference for one of two possible choices, or to indicate his indifference between the two.

Dyer's scheme is basically

$$\text{Set } \underset{\sim}{f}^A = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{i-1} \\ f_i \\ f_{i+1} \\ \vdots \\ f_p \end{bmatrix} \quad \text{and} \quad \underset{\sim}{f}^B = \begin{bmatrix} f_1 + \Delta f_1 \\ f_2 \\ \vdots \\ f_{i-1} \\ f_i - \Delta f_i \\ f_{i+1} \\ \vdots \\ f_p \end{bmatrix}$$

Select some small Δf_i. Repeatedly increase or decrease Δf_i depending on whether the DM prefers $\underset{\sim}{f}^B$ to $\underset{\sim}{f}^A$ or vice-versa, respectively. Continue until the DM is indifferent between $\underset{\sim}{f}^A$ and $\underset{\sim}{f}^B$, then $w_{ii} = \dfrac{\Delta f_i}{\Delta f_i}$.

Repeat this process for each $i = 1 \ldots p$.

Rosinger [43] constructs a wide family of possible inquiries on the direction of the p-dimensional vectors to improve the restrictive tradeoffs in the GDF procedure.

His extension allows the DM to

1. choose the way the inquiry is conducted
2. choose the marginal utilities of certain groups of objectives only
3. make statements which, to some extent, contradict each other

Rosinger's scheme is basically as follows.

The DM selects a number $q \geq 1$ of subsets of at least two elements

$$J_1, \ldots, J_q \subset \{1, \ldots, p\}$$

representing groups of objectives whose marginal utilities he is prepared to compare at a certain iteration. These subsets must satisfy certain requirements so as to ensure that adequate information is obtained from the DM. Then, the DM is asked to state the vectors

$$d_i = (d_{ij} \mid j \in J_i), \qquad 1 \leq i \leq q$$

trying to fulfil as much as possible the condition that d_i has the direction of $(m_{ij} \mid j \in J_i)$ where $m_j = \{m_{1j}, m_{2j}, \ldots, m_{nj}\}$ and $m_{ij} = \{\frac{\partial U}{\partial f_j} / \frac{\partial U}{\partial f_i}\}$ evaluated at the current solution.

It is convenient to reformulate the above ideas in terms of matrices.

The *inquiry pattern* P, which represents the groups of objectives compared, is constructed as follows

$$P = \{P_{ij}\}, \quad i = 1 \ldots q, j = 1 \ldots p$$

where

$$P_{ij} = \begin{cases} 1 & \text{if } j \in J_i \\ 0 & \text{if } j \notin J_i \end{cases}$$

Then the *P-answer* matrix D is formulated as follows

$$D = \{D_{ij}\}, \quad i = 1 \ldots q, j = 1 \ldots p$$

where

$$D_{ij} = \begin{cases} d_{ij} & \text{if } j \in J_i \\ 0 & \text{if } j \notin J_i \end{cases}$$

The characteristic property of these matrices D is given by $P_{ij} D_{ij} = D_{ij}$, for all $i = 1 \ldots q, j = 1 \ldots p$.

The nonzero vector of weights $\underline{\lambda}$ is then computed by minimizing, in some sense, the matrix

$$P \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_p \end{bmatrix} = \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_p \end{bmatrix} D$$

for some set of scaling factors $\underline{a} > 0$.

This scheme allows for more flexibility in questioning the DM for his MRS, as well as allowing him to be inconsistent in his estimation of the MRS.

In the GDF method a linear approximation to the DM's overall preference function is maximized over the constraint set at each iteration to determine the next feasible direction. Oppenheimer [41] tried to improve the convergence of the GDF procedure by differentiating between the local and global behaviour of the assumed preference function. He suggested maximising a proxy function instead of a linear approximation to the DM's utility function. This proxy function, denoted $P(\underline{f}(\underline{x}))$, should have the same general characteristics as $U(\underline{f}(\underline{x}))$ (at least locally at $\underline{x}^{(h)}$). The function must be concave on X and $\nabla P(\underline{f}(\underline{x}))$ should be colinear with $\nabla U(\underline{f}(\underline{x}))$ at $\underline{x}^{(h)}$ and $\underline{x}^{(h+1)}$ implying that

$$m_{1i}^{(h)} = \frac{\partial U(\underline{f}(\underline{x}^{(h)}))}{\partial f_i} \Big/ \frac{\partial U(\underline{f}(\underline{x}^{(h)}))}{\partial f_1} = \frac{\partial P(\underline{f}(\underline{x}^{(h)}))}{\partial f_i} \Big/ \frac{\partial P(\underline{f}(\underline{x}^{(h)}))}{\partial f_1}.$$

Oppenheimer proposed a sum of exponential functions, a sum of powers, and a sum of logarithms as suitable proxy functions. Oppenheimer suggested using one of these functions only as local proxy preferences in the GDF method because they are reasonable when assumed locally. In this procedure, at each iteration a new MRS vector is assessed to update the proxy. Then the new proxy is maximized over the constraint set to obtain an improved feasible solution for the next iteration.

Assuming we use the sum of exponentials

$$P(\underline{f}(\underline{x})) = -\sum_{i=1}^{p} a_i e^{-w_i f_i(\underline{x})}$$

as an example of a local proxy preference function. Then Oppenheimer's algorithm is

1. Select arbitrary feasible solutions $\underline{x}^0$ and $\underline{x}^1$. Set $h = 0$.

2. Assess the MRS $m_{1i}(\underline{f}(\underline{x}^0))$ and $m_{1i}(\underline{f}(\underline{x}^1))$ for $i = 2 \ldots p$.

3. Fit proxy $P(\underline{f}(\underline{x})) = -\sum_{i=1}^{p} a_i e^{-\omega_i f_i(\underline{x})}$.

a_i and ω_i are obtained using the following relation

$$m_{ki}(\underline{f}) = \frac{\partial P(\underline{f})}{\partial f_i} \Big/ \frac{\partial P(\underline{f})}{\partial f_k} = \frac{\omega_i a_i e^{-\omega_i f_i}}{\omega_k a_k e^{-\omega_k f_k}}$$

By taking logarithms and solving the set of linear equations, the proxy parameters a_i and ω_i are uniquely determined from the $2p-1$ assessments of $m_{ki}(\underline{f}(\underline{x}))$.

4. Maximize $P(\underline{f}(\underline{x}))$ over the feasible set to obtain a new iterate $\underline{x}^{(h+1)}$.

5. If $U(\underline{f}(\underline{x}^{(h+1)})) \approx U(\underline{f}(\underline{x}^{(h)}))$ then stop; $\underline{x}^{(h)}$ is the DM's most preferred solution. Otherwise, if $\underline{f}(\underline{x}^{(h+1)})$ is preferred to $\underline{f}(\underline{x}^{(h)})$ assess the MRS of the DM at $f(\underline{x}^{(h+1)})$. Set $h = h+1$ and go to 3.
If $\underline{f}(\underline{x}^{(h)})$ is preferred to $\underline{f}(\underline{x}^{(h+1)})$, find a new $\underline{f}(\hat{\underline{x}}^{(h+1)})$ preferred to $\underline{f}(\underline{x}^{(h)})$ such that

$$\underline{f}(\hat{\underline{x}}^{(h+1)}) = \rho \underline{f}(\underline{x}^{(h)}) + (1-\rho)\underline{f}(\underline{x}^{(h+1)}) \qquad 0 < \rho < 1$$

Assess the MRS of the DM at $\underline{f}(\hat{\underline{x}}^{(h+1)})$, set $h = h+1$ and go to 3.

Oppenheimer also examined techniques for checking MRS consistency, employing two types of consistency tests; the first testing MRS consistency of a single point, and the second testing the consistency at successive points.

The following additional variations are reported by Shin and Ravindran [49].

A method developed by Harrison and Rosenthal, which is just an extension of Oppenheimer's method, iteratively formulates an approximate explicit preference function using precise local tradeoff ratios obtained from the DM and then finds the next solution by solving the approximate function. A line search using the new solution is also performed. Wehrung also extended the GDF procedure by requiring only a binary preference relation rather than a real-valued preference function. However, this procedure also requires a line search to be performed by the DM. Hemming modified the direction finding step of the GDF procedure by replacing the precise local tradeoff ratios by vector comparisons. It presents only efficient solutions to the DM in the line search.

Musselman and Talavage [40] improved the GDF procedure by eliminating the line search. Here the indifference tradeoffs obtained from the DM are used to define a cutting plane which removes a portion of the feasible region containing points that have lower values of U than at the current point $x^{(k)}$. This method is described in detail in section 2.6.

2.2 The Reference Point Method and its Variants

The Reference point approach, proposed by Wierzbicki [63], falls into the class of Scalarizing Function Methods. The distinguishing characteristic of this approach is that the DM specifies reference values for the objectives at each iteration. This reference point is then projected onto the set of nondominated feasible solutions and presented to the DM, who then changes his reference levels. The satisficing solution is reached due to learning or improved understanding of the solution process. The Reference Point method is therefore said to support an *aspiration-led* search for a final solution.

Wiezbicki's approach makes use of a scalarizing function $s(\underline{f}(\underline{x}) - \underline{y})$ which is minimised in order to provide a nondominated solution which is "close" in some sense to the reference point $\underline{y}$ (or better if the reference point is feasible). The DM then either chooses the solution to be his satisficing solution, or modifies his reference objective levels in order to generate a new trial solution.

Wierzbicki describes two classes of scalarizing functions. The first is given by the following expression

$$s(\underline{f}(\underline{x}) - \underline{y}) = -U((\underline{f}(\underline{x}) - \underline{y})^+) + \rho\|(\underline{y} - \underline{f}(\underline{x}))^+\| \qquad (2.1)$$

where U is some utility function and $(\underline{f}(\underline{x}) - \underline{y})_i^+ = \max\{(f_i(\underline{x}) - y_i), 0\}$.

Wierzbicki terms this class *extended threshold utility functions* and states that this describes the behaviour of the average DM; if $\underline{f}(\underline{x}) > \underline{y}$, then the average DM will want to maximise his utility U, while if $\underline{f}(\underline{x}) < \underline{y}$, then his disutility corresponds to a distance from satisfying all his needs.

Various norms in $\Re^p$ and various utility functions can be used to define a specific form of (2.1). One of the more useful ones (according to Wierzbicki [63]) is

$$s(\underline{f}(\underline{x}) - \underline{y}) = -\max(\rho \min_{1 \le i \le p}(f_i(\underline{x}) - y_i), \sum_{i=1}^{p}(f_i(\underline{x}) - y_i)), \quad \rho > p$$

The second class of scalarizing functions are *penalty scalarizing functions*, an example of which is

$$s(\underline{f}(\underline{x})-\underline{y}) = \max_i \|(\underline{y}-\underline{f}(\underline{x}))^+\| - \|\underline{f}(\underline{x})-\underline{y}\|, \quad \rho \geq 1 \qquad (2.2)$$

This construction is based on simple reasoning : if $\underline{f}(\underline{x}) > \underline{y}$, we maximise the norm of $\underline{f}(\underline{x})-\underline{y}$, while if $\underline{f}(\underline{x}) < \underline{y}$, then we penalise for the distance between $\underline{f}(\underline{x})$ and $\underline{y}$.

Once again, many specific scalarizing functions of the form (2.2) can be defined using different norms.

Having selected a scalarizing function which is used to generate a nondominated solution, the Reference Point procedure is as follows:

1. Ask the DM to select an initial reference point $\underline{y}^0$. If the DM finds it impossible to specify a reference point, use $\underline{z}^*$. Set $h = 1$.

2. Solve

$$\begin{aligned} \min \quad & s(\underline{f}(\underline{x}) - \underline{y}^{(h)}) \\ \text{s. t.} \quad & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

   to obtain a nondominated solution $\underline{z}^{(h)}$.

3. If the DM is satisfied with the current levels of $f_i(\underline{z}^{(h)})$, $i = 1 \ldots p$, stop, $\underline{z}^{(h)}$ is the best-satisficing solution for the DM. Otherwise, ask the DM to modify $\underline{y}^{(h)}$ to a new reference point $\underline{y}^{(h+1)}$ by considering the current levels of the objective functions. Set $h = h+1$ and go to 2.

The Reference Point method is discussed in more detail in section 3.3.1

2.2.1 Comments

1. The Reference Point Method requires a lot of active involvement on the part of the DM – it does not help the DM to find improved solutions.

2. The DM can experiment with new ideas at any time and abruptly redirect the search effort. The method enables the user to essentially "steer" himself to a final solution.

3. There is no possibility of the optimal solution being "lost" by inaccurate information from the DM as there is in methods which reduce the constraint set.

2.2.2 Variations

A number of methods which use Wierzbicki's approach, defining their own scalarizing functions, are listed in Steuer *et al.* [54].

Zeleny [65] developed the method of the displaced ideal based on his 'Axiom of Choice'

> "Alternatives that are closer to the ideal are preferred to those that are farther away. To be as close as possible to the perceived ideal is the rationale of human choice."

This method can be considered a special case of the scalarizing function technique where the scalarizing function used is the family of L_p metrics, defined in definition 1.24, and the ideal is initially used as the reference point.

If the objective functions are not expressed in commensurable terms, then a scaling function is introduced to ensure the same range for each objective function. Usually this range corresponds to the interval $[0, 1]$. This scaling is accomplished by defining the scaling function

$$d_i(\underline{x}) = \frac{z_i^* - f_i(\underline{x})}{z_i^* - z_i^\circ}$$

Then the distance of a solution from the ideal point is given by weighted L_p

norm

$$s = \| d_i(\underline{x}) \|_p$$

The basic approach of the algorithm can be summarized as follows

1. Calculate $\underline{z}^*$ and $\underline{z}^n$, define d_i's.
2. Minimize s over the constraint set for $p = 1, 2, \infty$.
3. Present the resultant solutions and the ideal point to the DM. If the DM is satisfied with one of these solutions, the algorithm terminates. Otherwise, the DM imposes further restrictions on the constraint set. A new ideal (displaced) is then calculated over the constraint set and the method is restarted.

Seo and Sakawa [48] report the following extensions of the Reference Point Method.

Sakawa and Yano have developed a fuzzy multiobjective programming method which can be viewed as a fuzzy version of the Reference Point method.

Sakawa, Yumine and Yano developed an extended version of the Reference Point method in which the DM is supplied with the tradeoff information.

Instead of solving the penalty scalarizing problem, they solve a minimax problem, viz./

$$\begin{aligned} \min \ & \max_{1 \le i \le p} (y_i - f_i(\underline{x})) \\ \text{s. t. } & A\underline{x} \le \underline{b} \\ & \underline{x} \ge \underline{0} \end{aligned}$$

or equivalently,

$$\begin{aligned} \min \ & \alpha \\ \text{s. t. } & (y_i - f_i(\underline{x})) \le \alpha \quad i = 1 \ldots p \\ & A\underline{x} \le \underline{b} \\ & \underline{x} \ge \underline{0} \end{aligned}$$

This solution is presented to the DM who must either accept it as his satisficing solution or modify his reference point. In order to help the DM express his reference levels, tradeoff information between a reference objective, f_j say, and each of the other objective functions, is displayed. These tradeoffs are easily obtainable from the Lagrange multipliers of the minimax problem according to the relation $\frac{-\partial f_k}{\partial f_i} = \frac{\lambda_i}{\lambda_j}$, where the λ_i are the Lagrange multipliers associated with the constraints of the minimax problem.

The advantage of this variation is that the DM is supplied with tradeoff information as well as objective function levels when trying to express a reference point.

Korhonen and Laakso [28] have presented a Visual Interactive Algorithm which makes use of scalarizing functions. This algorithm is presented in the next section.

2.3 The Visual Interactive Method and its Variants

Korhonen and Laakso [28] presented this graphic-aided interactive approach in an attempt to obtain a method which is easy to use, helps the DM to find improved solutions, and allows for changes to the DM's preference function during the iterative process.

The method is essentially a combination of the Geoffrion-Dyer-Feinberg algorithm and Wierzbicki's scalarizing function approach. Instead of the Frank-Wolfe gradient approach, the search direction is determined using a reference point. The vector from the current solution to the reference point is used as a reference direction.

This reference direction is then projected onto the set of nondominated feasible solutions using a scalarizing function. This results in an efficient curve starting from the current solution and traversing across the efficient frontier.

The step length problem is solved by presenting this whole curve to the DM in the form of a graph of the objective function values along the curve. The DM is then asked to indicate which point of the efficient curve he likes most.

If certain optimality conditions are satisfied, the procedure stops, else, a new search direction is identified and the process is repeated.

More formally, the Visual Interactive Algorithm is

1. Find an initial solution $z^{(0)}$. Set $h = 1$.

2. Obtain a reference point $y^{(h)}$ from the DM. The reference direction $q^{(h)}$ is given by $q^{(h)} = y^{(h)} - z^{(h-1)}$

3. Solve

$$\begin{aligned} \min \ & s(\underline{f}(\underline{x}) - \underline{y}^{(h)}) \\ \text{s.t.} \ & \underline{f}(\underline{x}) = \underline{z}^{(h-1)} + \theta \underline{q}^{(h)} \\ & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

as θ is increased from 0 to ∞, where s is a scalarizing function.

4. Display the objective function values $\underline{f}(\underline{x})$ as θ varies from 0 to ∞, and ask the DM to select his preferred solution. Designate this solution $\underline{z}^{(h)}$.

5. If $\underline{z}^{(h)} \neq \underline{z}^{(h-1)}$, set $h = h + 1$ and go to 2. Otherwise, if the DM is unsatisfied with $\underline{z}^{(h)}$, generate a new search direction $\underline{q}^{(h+1)}$, set $h = h+1$, and go to 3. The way in which new search directions are generated is described in section 3.3.2.

2.3.1 Comments

1. The algorithm assists DM's in finding improved solutions without making any reductions to the feasible region. This means that the DM can alter his preference function without necessitating an algorithm restart.

2. Although the method of interaction with the DM is one of the algorithm's strengths, this may become problematic if there are more than four or five objectives.

3. The method is not based on unduly restrictive assumptions concerning the DM's behaviour or utility function.

4. The DM is free to investigate any part of the nondominated set he pleases; he is not confined to evaluating only extreme point solutions or limited by previous choices.

5. The performance of the method depends on the DM's ability to specify reference directions which lead to improved solutions.

2.3.2 Variations

Korhonen and Wallenius [29] have developed an approach they call a Pareto Race. It combines the above ideas with the ideas of exploring all nondominated solutions. The entire nondominated set can be explored by varying the parameters d (direction) and θ (step-size). However, instead of specifying the parameters explicitly, the DM explores the nondominated set by "driving" around on it by performing the following actions, and the algorithm is almost like a video game.

1. Use the accelerator - move through the nondominated set in the current direction at constant speed
2. Use the gears - increase the speed of travel through the nondominated set
3. Reverse - increase the speed of travel in the backward direction
4. Use the brakes - decrease the speed of travel through the nondominated set
5. Turn - change direction of the path along the nondominated set so that a particular objective is increased.
6. Fix an objective at a minimum acceptable level, or free a previously fixed objective.

The Pareto Race uses bar graphs to represent the values of each objective and as the user moves around on the nondominated frontier, the bar graphs change.

Shin and Ravindran [49] report that with the rapid advance in computer technology, this type of approach is becoming popular.

2.4 Steuer's Interval Criterion Weights Method and its Variants

Steuer's Interval Criterion Weights Method [51, 53] is classified by Shin and Ravindran [49] as a Criterion Weight Space Method. It is in effect an extension of the "naïve" technique of generating all nondominated extreme points (see section 1.3).

Steuer uses the weighted sums approach to solving the vector-maximum problem, viz. solve the following family of weighted sum problems

$$\begin{aligned} &\max \ \underset{\sim}{\lambda}^T C\underset{\sim}{x} \\ &\text{s. t. } A\underset{\sim}{x} \leq \underset{\sim}{b} \\ &\underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned} \tag{2.3}$$

for all $\underset{\sim}{\lambda} \in \Lambda$ where $\Lambda = \{\lambda \in \Re^p | \lambda_i \in (0,1), \sum_{i=1}^p \lambda_i = 1\}$.

Unfortunately, even moderately sized MOLP problems usually have an unworkably large number of nondominated extreme points. Steuer's idea is to generate a small sample of the nondominated extreme points and present these to the DM, who indicates the one he likes most. The area around this solution is then investigated further, while the area "far away" from the selected solution is discarded. This is achieved by reducing the criterion weight interval in the weighted sums problem. In other words

$$\begin{aligned} &\max \ \underset{\sim}{\lambda}^T C\underset{\sim}{x} \\ &\text{s. t. } A\underset{\sim}{x} \leq \underset{\sim}{b} \\ &\underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned} \tag{2.4}$$

where $\underset{\sim}{\lambda} \in \Lambda^{(h)}$ and $\Lambda^{(h)} = \{\underset{\sim}{\lambda} \in \Re^p | \lambda_i \in (l_i, \mu_i), \sum_{i=1}^p \lambda_i = 1\}$. Clearly $\Lambda^{(h)} \subseteq \Lambda$.

Steuer [53] states that problem (2.4) cannot be solved in its present form but can be translated into an equivalent weighted sum problem by examining the effects of the subinterval criterion weights on the gradient cone. In other words,

problem (2.4) is converted to the equivalent problem

$$\begin{aligned} &\max \ \underline{\lambda}^T D \underline{x} \\ &\text{s. t. } A\underline{x} \leq \underline{b} \\ &\qquad \underline{x} \geq \underline{0} \end{aligned} \tag{2.5}$$

where D is a contraction of C, the original criterion cone and $\underline{\lambda} \in \Lambda$. It is for this reason that the algorithm is often referred to as Steuer's Contracting Cones Method.

The algorithm is thus

1. Set $h = 0$. Set $C^{(0)} = C$
2. Generate $2p + 1$ λ-vectors from the set Λ. Various ways in which this can be done are discussed in section 4.5. Steuer [53] generates these by formula.
3. Solve

$$\begin{aligned} &\max \ \underline{\lambda}^T C^{(h)} \underline{x} \\ &\text{s. t. } A\underline{x} \leq \underline{b} \\ &\qquad \underline{x} \geq \underline{0} \end{aligned}$$

for each of the $2p + 1$ λ-vectors generated in step 2 above.

4. Present the solutions to the DM and ask him to select the one he likes most.
5. Using this selected solution, construct a reduced criterion cone $C^{(h+1)}$. Set $h = h + 1$ and go to 2. Practical procedures for reducing the cone are described in section 4.7.

2.4.1 Comments

1. The algorithm only searches among extreme points to locate the efficient extreme point of greatest utility. However there is no guarantee that the algorithm will conclude with the efficient extreme point of greatest utility.

2. The only interaction with the DM is to request him to choose one solution out of a group.

3. Since a number of LP problems must be solved at each iteration, this algorithm requires a considerable amount of CPU time.

2.4.2 Variations

The Interactive Weighted Sums/Filtering Approach proposed by Steuer and Wood [56] is similar in philosophy to the Interval Criterion Weight approach, the main difference being that instead of reducing the criterion cone, the procedure reduces the weighting space Λ directly. This is made possible by generating weights from Λ randomly and filtering to the required sample size, instead of generating the weights by formula. Hence the approach is

1. Set $h = 0$.

2. Generate a large number of λ-vectors from the set Δ and filter down to the required number.

3. Solve

$$\begin{aligned} &\max \ \underset{\sim}{\lambda}^T C^{(h)} \underset{\sim}{x} \\ &\text{s. t. } A\underset{\sim}{x} \leq \underset{\sim}{b} \\ &\underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

for each of the λ-vectors generated in step 2 above.

4. Present the criterion vectors to the DM and ask him to select the one he likes most. Denote his preferred solution $\underset{\sim}{z}^{(h)}$ and the weights which generated this solution $\underset{\sim}{\lambda}^{(h)}$.

5. Using this selected solution, reduce the weight space Λ as follows

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

where r is a reduction factor. Set $h = h + 1$ and go to 2.

Steuer and Choo [55] have developed an Interactive Weighted Tchebbycheff Procedure in which the weighted sums LP problem to generate a nondominated solution is replaced by a scalarizing function problem (see section 5.2). The scalarizing function used is the Tchebbycheff norm. In addition instead of using the λ-vector corresponding to the DM's selected solution to reduce the weight space, the vertex-associated λ-vector corresponding to the solution is used.

Hence the algorithm is

1. Set $h = 0$.

2. Generate a large number of λ-vectors from the set Λ and filter down to the required number.

3. Solve

$$\begin{aligned} &\min \{\alpha + \varepsilon \sum_{i=1}^{p} (z_i^* - z_i)\} \\ &\text{s.t. } \alpha \geq \lambda_i (z_i^* - z_i) \quad i = 1 \ldots p \\ &\qquad f_i(\underset{\sim}{x}) = z_i \qquad i = 1 \ldots p \\ &\qquad A\underset{\sim}{x} \leq \underset{\sim}{b} \\ &\qquad \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

for each of the λ-vectors generated in step 2 above.

4. Present the criterion vectors to the DM and ask him to select the one he likes most. Denote his preferred solution $\underset{\sim}{z}^{(h)}$.

5. Using this selected solution, reduce the weight space Λ as follows

Let $\lambda_i^{(h)} = \begin{cases} \dfrac{1}{z_i^* - z_i^{(h)}} \left[\sum_{i=1}^{p} \dfrac{1}{z_i^* - z_i^{(h)}}\right]^{-1} & \text{if } z_i^{(h)} \neq z_i^* \ \forall i = 1 \ldots p \\ 1 & \text{if } z_i^{(h)} = z_i^* \\ 0 & \text{if } z_i^{(h)} \neq z_i^* \text{ but } \exists j \text{ s.t. } z_j^{(h)} = z_j^* \end{cases}$

Then,

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

where r is a reduction factor. Set $h = h + 1$ and go to 2.

The Interval Criterion Weights Method, the Interactive Weighted Sums/Filtering Method and the Interactive Weighted Tchebbycheff Method are discussed in more detail further in this document

2.5 The Method of Zionts and Wallenius and its Variants

The approach by Zionts and Wallenius [69] is another pioneer method in the class of Criterion Weight Space Methods. It is similar to the Criterion Weight Space Methods given in the previous section but obtains the weights and constrains the weight space in a revolutionary new way.

The method takes an initial set of weights $\underline{\lambda}^{(0)}$ and determines some initial nondominated extreme point by using the Simplex Algorithm to solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i^{(0)} f_i(\underline{x}) \\ \text{subject to} \quad & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

For each nonbasic variable in the simplex tableau which would lead to an efficient solution if introduced into the basis, a vector of tradeoffs is computed and presented to the DM for consideration. The DM is required to decide whether each set of tradeoffs is desirable, undesirable or neither of the two.

Now if the DM likes a tradeoff $\underline{t}$, then

$$\sum_{i=1}^{p} t_i \lambda_i > 0$$

Similarly if the DM dislikes a tradeoff $\underline{t}$, then

$$\sum_{i=1}^{p} t_i \lambda_i < 0$$

and

$$\sum_{i=1}^{p} t_i \lambda_i = 0$$

if the DM is indifferent to the tradeoff.

So having presented the efficient tradeoffs to the DM, the weights space is constrained as above, and a set of weights which satisfy these constraints such

that $\sum_{i=1}^{p} \lambda_i = 1$ is found. This set of weights is used to solve another weighted sums LP problem and the algorithm is restarted.

2.5.1 Comments

1. The implicit assumption that the utility function is linear means that only extreme point solutions will be generated.
2. Even if the most preferred solution does occur at an extreme point, there is no guarantee that the weights which generate this point will satisfy the constraints on the weights for a more general class of utility functions than the linear form. (Jordi [23] p 36)
3. The major advantage of this method is that it is less demanding than most other methods.

2.5.2 Variations

Zionts and Wallenius [71] refined the above method to allow for an underlying pseudoconcave utility function. The method is essentially the same but takes into consideration special factors relating to the pseudoconcavity of U. This algorithm is presented in further detail in section 3.2.1.

Stewart [58] incorporated aspects of the UTA method into the method of Zionts and Wallenius. By using piecewise-linear approximations to an assumed underlying concave additive utility function, many more vertices are created than exist in the original LP problem. Investigating these vertices will therefore bring us closer to the DM's most-preferred solution than is the case if only the vertices of the original LP are investigated.

Assuming the attributes to be mutually preferential-independent, Stewart expresses his utility function $U(\underset{\sim}{f}(\underset{\sim}{x}))$ as $U = \sum_{i=1}^{p} U_i(f_i(\underset{\sim}{x}))$.

Without loss of generality, we assume that $U_i(z_i^*) = 0, \forall i = 1 \ldots p$ and $U(z^*) = W$ where W is some suitable positive constant.

Stewart's argument is as follows.

For each attribute i, partition the interval $[z_i^*, z_i^0]$ into v_i segments of equal length Δ_i.

Make the approximation $U_i(z_i) \approx \sum_{k=1}^{v_i} \theta_{ik} y_{ik}$ where the y_{ik} satisfy

$$
\begin{aligned}
&0 \leq y_{ik} \leq \Delta_i \\
&y_{i,k+1} > 0 \Rightarrow y_{ik} = \Delta_i \\
&\textstyle\sum_{k=1}^{v_i} y_{ik} - c_i^T \underline{x} = -z_i^*
\end{aligned}
$$

where $\Delta_i = \dfrac{z_i^* - z_i^0}{v_i}$.

The problem

$$
\begin{aligned}
\max \quad & z = U(f(\underline{x})) \\
\text{subject to} \quad & A\underline{x} \leq \underline{b} \\
& \underline{x} \geq \underline{0}
\end{aligned}
$$

can therefore be approximated by

$$
\begin{aligned}
\max \quad & z = \sum_{i=1}^{p} \sum_{k=1}^{v_i} \theta_{ik} y_{ik} \\
\text{subject to} \quad & 0 \leq y_{ik} \leq \Delta_i && \forall i = 1 \ldots p, k = 1 \ldots v_i \\
& \textstyle\sum_{k=1}^{v_i} y_{ik} - c_i^T \underline{x} = -z_i^* && \forall i = 1 \ldots p \qquad (2.6) \\
& A\underline{x} \leq \underline{b} \\
& \underline{x} \geq \underline{0}
\end{aligned}
$$

Note that if the DM prefers $\underline{x}^A$ to $\underline{x}^B$, then

$$
\begin{aligned}
& U(\underline{x}^A) > U(\underline{x}^B) \\
\Rightarrow\ & U(\underline{x}^A) - U(\underline{x}^B) > 0 \\
\Rightarrow\ & \sum_{i=1}^{p} [U_i(z_i^A) - U_i(z_i^B)] > 0 \\
\Rightarrow\ & \sum_{i=1}^{p} \sum_{k=1}^{v_i} \theta_{ik} (y_{ik}^A - y_{ik}^B) > 0
\end{aligned}
$$

So by making pairwise comparisons between solutions, restrictions can be imposed on the set of θs which can feasibly represent the DM's preferences.

Stewart's scheme is therefore

1. Define
$$\Theta = \{\theta \in \Re^N | \theta_{i1} > \theta_{i2} > \cdots \theta_{iv_i} > 0, \quad \forall i = 1 \ldots p$$
$$\sum_{i=1}^{p} \sum_{k=1}^{v_i} \Delta_i \theta_{ik} = W\}$$

2. Select an arbitrary $\theta \in \Theta$. Obtain the corresponding solution to (2.6) and denote the vector of objective function values at this solution $\underline{z}^A$.

3. Obtain a feasible solution to (2.6), $(\underline{x}^B, \underline{y}^B)$ say, which is optimal for some $\theta \in \Theta$ such that $\underline{z}^A \neq \underline{z}^B$. If no such solution exists, then $\underline{z}^A$ is optimal for all $\theta \in \Theta$.

4. Present $\underline{z}^A$ and $\underline{z}^B$ to the DM and set
$$\Theta = \Theta \cap \{\theta \in \Re^N | \sum_{i=1}^{p} \sum_{k=1}^{v_i} \theta_{ik}(y_{ik}^A - y_{ik}^B) > 0\}$$
if the DM prefers $\underline{z}^A$ to $\underline{z}^B$ and
$$\Theta = \Theta \cap \{\theta \in \Re^N | \sum_{i=1}^{p} \sum_{k=1}^{v_i} \theta_{ik}(y_{ik}^A - y_{ik}^B) < 0\}$$
if the DM prefers $\underline{z}^B$ to $\underline{z}^A$.

5. Go to step 2.

Practical procedures for steps 2 and 3 are given in Stewart [58].

This method will generate an approximation to the DM's most preferred solution even when this is not a basic solution to the original problem, provided that the DM's preferences can be represented by an additive value function, with concave terms. This latter condition can, in principle, be verified a priori.

White [62] derives his method in part from Zionts and Wallenius. This method allows for an unspecified form of underlying utility. An extreme point of X

is found. A set of weights is then used to generate further extreme points which are then compared to each other by the DM in an attempt to find the best solution so far. Responses from the DM are incorporated into constraints which are used to shrink the convex cone containing the optimal weights. A new set of weights is computed from which further solutions are found, and presented to the DM.

Shin and Ravindran [49] describe the following variations of the method of Zionts and Wallenius.

Malakooti and Ravindran improve the method by emploting the paired comparisons of alternatives and strength of preference. They also propose an interesting approach to handle the problem of the DM's inconsistency. The method is extended to problems with quasi-concave preference functions by Malakooti.

In the Zionts and Wallenius method, when inconsistency arises in the weight space, some of the constraints on weights are dropped and this results in a loss of information on the preference structure. Ramesh, Karwan and Zionts proposed a hybrid scheme to avoid this problem.

Roy and Wallenius extended the method of Zionts and Wallenius to a unified approach to solve various classes of multiple objective problems by using the generalised reduced gradient method.

2.6 The Method of Musselman and Talavage and its Variants

The class of methods defined by Shin and Ravindran [49] as tradeoff cutting plane methods is essentially another variation of the feasible direction method. The methods of this class are unique in that they isolate the best compromise solution by iteratively reducing the objective function space by cutting planes. The method of Musselman and Talavage [40] is such a tradeoff cutting plane method. It uses local tradeoffs to progressively eliminate a portion of the objective space. It uses the method of centres to locate the new solution point towards the centre of the remaining objective space.

From the concavity assumption of U,

$$U(\underline{f}(\underline{x})) \leq U(\underline{f}(\underline{x}^{(k)})) + \nabla U(\underline{f}(\underline{x}^{(k)}))^T(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(k)})) \qquad (2.7)$$

Assuming $\partial U(\underline{f}(\underline{x}^{(k)}))/\partial f_1 > 0$, we divide (2.7) by $\partial U(\underline{f}(\underline{x}^{(k)}))/\partial f_1$ to get

$$\sum_{j=1}^{n} m_{1j}^{(k)}(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(k)})) \geq \frac{U(\underline{f}(\underline{x})) - U(\underline{f}(\underline{x}^{(k)}))}{\partial U(\underline{f}(\underline{x}^{(k)}))/\partial f_1} \qquad (2.8)$$

where $m_{1i}^{(k)}$ is the MRS between objectives 1 and i at the point $\underline{x}^{(k)}$.

Since we wish to generate solutions which are better that the current one, we henceforth limit the solutions generated to those that have the property that $U(\underline{f}(\underline{x})) \geq U(\underline{f}(\underline{x}^{(k)}))$ where $\underline{x}^{(k)}$ is the current solution.

From (2.8), this implies $\sum_{j=1}^{n} m_{1j}^{(k)}(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(k)})) \geq 0$ which defines a cutting plane called the tradeoff cut by Musselman and Talavage.

At each iteration, one more constraint of the above form is generated in order to reduce the constraint set further. Denote the constraint set at the kth iteration as $X^{(k)}$.

In order to find the next point $\underline{x}^{(k+1)}$ from $\underline{x}^{(k)}$, the method uses the two usual steps; finding the direction of search, and the step size in that direction. The

direction of search used here, however, is not the direction of steepest ascent but rather a feasible direction of improving U that attempts to move to the centre of the feasible region in order to make the next tradeoff cut as large as possible.

The usable direction $\underline{g}^{(h)}$ at the hth iteration can be obtained by solving the following LP problem

$$\begin{aligned} &\min \alpha \\ &\text{s.t. } g_i(\underline{x}^{(h)}) + \nabla_{\underline{x}} g_i(\underline{x}^{(h)})^T \underline{g} \leq \alpha \qquad i = 1 \ldots m+h \\ &\qquad -1 \leq d_j \leq 1 \qquad\qquad\qquad\quad j = 1 \ldots n \end{aligned}$$

where

$$g_i = \sum_{j=1}^{n} a_{ij} x_j \qquad \text{for } i = 1 \ldots m$$

$$g_i = \sum_{j=1}^{n} m_{ij}^{(h)} (f_j(\underline{x}) - f_j(\underline{x}^{(h)})) \text{ for } i = m+1 \ldots m+h$$

If the optimal $z^* \geq 0$, then the best compromise solution is found since there is no feasible direction in $X^{(h)}$ where U can be further improved. If on the other hand $z^* < 0$, then the optimal $\underline{g}^{(h)}$ is a feasible direction in $X^{(h)}$; hence a direction of improving $U(\underline{f}(\underline{x}))$ is ensured by the tradeoff cut.

Having found the direction of search $\underline{g}^{(h)}$, it is necessary to determine the step size $\rho^{(h)}$ which maximizes $U(\underline{f}(\underline{x}^{(h)} + \rho^{(h)} \underline{g}^{(h)}))$. Musselman and Talavage suggest a trial-and-error procedure for this. This trial-and-error procedure requires interaction with the DM in the form of tradeoff estimation.

The complete algorithm of Musselman and Talavage [40] is therefore

1. Select an initial point $\underline{x}^0 \in X$. Set $h = 1$.

2. Determine the tradeoffs $m_{ij}^{(h)}$.

3. Determine the direction vector by solving the LP

$$\begin{aligned} &\min \alpha \\ &\text{s.t. } g_i(\underline{x}^{(h)}) + \nabla_{\underline{x}} g_i(\underline{x}^{(h)})\underline{d} \leq \alpha \quad i = 1\ldots,m+h \\ &\qquad -1 \leq d_j \leq 1 \qquad\qquad\qquad\quad j = 1\ldots n \end{aligned}$$

where

$$g_i = \sum_{j=1}^{n} a_{ij}x_j \qquad \text{for } i = 1\ldots m$$

$$g_i = \sum_{j=1}^{n} m_{ij}^{(h)}(f_j(\underline{x}) - f_j(\underline{x}^{(h)})) \text{ for } i = m+1\ldots m+h$$

to obtain $(\alpha^{(h)}, \underline{d}^{(h)})$. If $\alpha^{(h)} \geq 0$, stop, $\underline{x}^{(h)}$ is the preferred solution of the DM. If $\alpha^{(h)} < 0$, then $\underline{d}^{(h)}$ is the usable direction vector

4. Determine the stepsize $\rho^{(h)}$ which maximises $U(f(\underline{x}^{(h)} + \rho^{(h)}\underline{d}^{(h)}))$. Musselman and Talavage [40], page 1429, describe a trial-and-error procedure for doing this. Set $\underline{x}^{(h+1)} = \underline{x}^{(h)} + \alpha^{(h)}\underline{d}^{(h)}$. Set $h = h+1$ and go to step 2.

2.6.1 Comments

1. Loganathan and Sherali [33] comment that this method represents an improvement over the Geoffrion-Dyer-Feinberg method in that it directs the search towards improving solutions.

2. In executing the algorithm, a sequence is constructed which is either finite (in which case the last point generated is the best-compromise solution) or else it is infinite and has the property that any limit point of this sequence is a best-compromise solution.

3. The possibility exists that the most preferred solution could be eliminated by the tradeoff cut if the tradeoffs provided by the DM are inaccurate.

4. The method requires a large amount of user interaction, particularly in the line search procedure.

5. The method relies heavily on a subjective determination of indifference tradeoffs, which has often been questioned from the practical viewpoint (Chankong and Haimes [6], p. 345).

2.6.2 Variations

Loganathan and Sherali [33] replaced the direction and step size finding problem (steps 3 and 4) of the Musselman and Talavage algorithm by the following

$$\begin{aligned} &\max \alpha \\ &\text{s.t. } \alpha \leq \sum_{i=1}^{p} m_{1i}^{(j)}(f_i(\underline{x}) - f_i(\underline{x}^{(j)})) \quad j = 1 \ldots h \\ &\underline{x} \in X \end{aligned}$$

where $m_{1i}^{(j)}$ is the MRS between objectives 1 and i at the point $\underline{x}^{(j)}$.

Denote the solution $(\underline{x}^{(h+1)}, \underline{z}^{(h+1)})$. If $\underline{z}^{(h+1)} = 0$, stop, $\underline{x}^{(h+1)}$ is optimal, else set $h = h + 1$ and go to step 2.

In other words, the Loganathan and Sherali algorithm is

1. Select an initial (preferably efficient) point $\underline{x}^0 \in X$. Set $h = 1$.
2. Determine the direction vector by solving the LP

$$\begin{aligned} &\max \alpha \\ &\text{s.t. } \alpha \leq \sum_{i=1}^{p} m_{1i}^{(j)}(f_i(\underline{x}) - f_i(\underline{x}^{(j)})) \quad j = 1 \ldots h \\ &\underline{x} \in X \end{aligned}$$

   Call the solution $(\underline{x}^{(h+1)}, \alpha^{(h+1)})$.
3. If $\alpha^{(h)} = 0$, stop, $\underline{x}^{(h+1)}$ is the preferred solution of the DM. Otherwise, set $h = h + 1$ and go to step 2.

The advantages of this algorithm over the one of Musselman and Talavage are

1. Line searches are eliminated; the solution to the subproblem is the next iterate. This reduces the amount of interaction with the DM.
2. Each iterate generated by the algorithm is a nondominated solution.
3. X need not be explicitly mathematically defined.
4. The algorithm maintains its important features, including convergence properties, even if the objective functions are not concave and the feasible region is not convex.

Shin and Ravindran [49] report two additional variations; Shin and Ravindran combined the direction finding step of Sandagopan and Ravindran and the cutting plane concept, and Malakooti and Deviprasad developed a heuristic cut approach using the method of pairwise comparisons.

A method developed by Benvenuti and Jordi [2, 4] is similar to the one of Loganathan and Sherali, the primary difference being in the weights specifying the tradeoff cut.

Loganathan and Sherali use the following tradeoff cut

$$\alpha \leq \sum_{i=1}^{n} \lambda_i^{(j)} (f_i(\underline{x}) - f_i(\underline{x}^{(j)}))$$

where $\lambda_i^{(j)}$ is the value of the MRS between objectives 1 and i at $\underline{x}^{(j)}$. Benvenuti and Jordi use the same cutting plane but evaluate $\lambda_i^{(j)}$ using the Zionts and Wallenius technique for determining weights. In other words, having generated an iterate $\underline{x}^{(h)}$, the DM is presented with the tradeoffs representing efficient edges from $\underline{x}^{(h)}$.

For each tradeoff presented to the DM a constraint is set up. If the DM likes tradeoff t_j, then the constraint $\sum_{i=1}^{n} t_{ij}\lambda_i > 0$ is set up.

Similarly, if the DM dislikes tradeoff t_j, then the constraint $\sum_{i=1}^{n} t_{ij}\lambda_i < 0$ is set up.

Then a set of weights $\underline{\lambda}$ is selected such that $\sum_{i=1}^{n} \lambda_i = 1$ and the above

constraints are satisfied. Benvenuti and Jordi [2, 4] suggest that weights be selected which are in the centre of the set of allowable weights.

The advantage of this method over the one of Loganathan and Sherali is that the DM does not need to estimate tradeoffs at a solution. Rather, he must simply state whether he likes or dislikes a given set of tradeoffs for the current solution.

The method of Benvenuti and Jordi will be investigated further later in this document and is described in detail in section 3.2.2.

2.7 The STEM Method and its Variants

The STEP Method (STEM), proposed by Benayoun *et al.* [1], is classified by Shin *et al.* [49] as a feasible region reduction method. Feasible region reduction methods are classified as those having three phases to each iteration; viz. a calculation phase, a decision phase, and a feasible reduction phase. In the calculation phase, an efficient solution which is nearest, in the minimax sense for given weights, to the ideal solution is obtained. In the decision phase, the DM interacts with the method, and his responses are used to construct additional constraints in the feasible region reduction phase. The method continues to perform the three phase iterations until the DM considers the current solution to be the best current solution.

The STEM process follows the following steps.

0. Initialization

Construct a pay-off table as follows

	1	2	...	p
1	$f_1(\underset{\sim}{x}_1^*)$	$f_2(\underset{\sim}{x}_1^*)$	$\cdots$	$f_p(\underset{\sim}{x}_1^*)$
2	$f_1(\underset{\sim}{x}_2^*)$	$f_2(\underset{\sim}{x}_2^*)$	$\cdots$	$f_p(\underset{\sim}{x}_2^*)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
p	$f_1(\underset{\sim}{x}_p^*)$	$f_2(\underset{\sim}{x}_p^*)$	$\cdots$	$f_p(\underset{\sim}{x}_p^*)$

where $\underset{\sim}{x}_i^*$ is the solution to

$$\begin{aligned} \max \quad & z = f_i(\underset{\sim}{x}) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

for $i = 1 \ldots p$.

In other words, the ith row in the payoff table consists of the objective function values at the solution $\underset{\sim}{x}_i^*$ which maximizes $f_i(\underset{\sim}{x})$ over the constraint set. Note that the elements found on the diagonal of the payoff matrix make up the elements of the ideal solution. (defn 1.7).

1. Calculation Phase

For each iteration h, a feasible solution $\underline{x}^{(h)}$ whose criterion vector is closest to $\underline{z}^*$ in the minimax sense, is sought. This is achieved by solving the following problem

$$\begin{aligned} &\max && \alpha \\ &\text{subject to} && \alpha \geq \lambda_i^{(h)}(z_i^* - f_i(\underline{x})) \quad \forall i = 1 \ldots p \\ &&& \underline{x} \in X^{(h)} \\ &&& \alpha \geq 0 \end{aligned}$$

where $X^{(0)}$ is $\{\underline{x} | A\underline{x} \leq \underline{b}, \underline{x} \geq \underline{0}\}$.

Call the solution $\underline{x}^{(h)}$.

The coefficients $\lambda^{(h)}$ give the relative importance to the distances to the optima, and are calculated as follows.

Letting m_i be the minimum value in the ith column of the payoff table, let

$$\pi_i = \begin{cases} \dfrac{z_i^* - m_i}{z_i^*} \left[\sum_{i=1}^{p} (c_{ij})^2\right]^{-\frac{1}{2}} & \text{when } z_i^* > 0 \\ \dfrac{m_i - z_i^*}{m_i} \left[\sum_{i=1}^{p} (c_{ij})^2\right]^{-\frac{1}{2}} & \text{when } z_i^* \leq 0 \end{cases}$$

Then $\lambda_i^{(h)} = \dfrac{\pi_i}{\sum_{j=1}^{p} \pi_j}$.

Each weight π_i is made up of a product of two terms. From the first term the following assertion can be made : if the value of f_i does not vary very much from the optimal solution by varying $\underline{x}$, the corresponding objective is not sensitive to a variation in the weights, therefore a small weight is assigned to this optimal solution. As the variation gets larger, the corresponding weight gets bigger. The second term normalizes the values taken by the objective functions according to the L_2 norm.

The λ's are just a normalized version of the π's such that $\sum \lambda_i = 1$.

2. Decision Phase

The DM now compares $\underline{f}(\underline{x}^{(h)})$ with $\underline{z}^*$. If some of the components of $\underline{f}(\underline{x}^{(h)})$ are satisfactory and others not, the DM must accept a drop in a satisficing objective in order to allow an improvement of the unsatisfactory ones in the next iteration. He is therefore asked which objective he wishes to relax and by how much. Let the objective to be relaxed be objective j and the amount by which it is relaxed be δ_j.

3. Feasible Region Reduction Phase

Reduce the feasible region as follows

$$X^{(h+1)} = \{\underline{x} \in X^{(h)} \text{ such that } \begin{cases} f_i(\underline{x}) \geq z_i(\underline{x}^{(h)}) & \forall i = 1 \ldots p, (\neq j) \\ f_j(\underline{x}) \geq z_j(\underline{x}^{(h)}) - \delta_j \end{cases}\}$$

Set $h = h + 1$ and go to 1 but set $\pi_i^{(h)} = 0$ if objective i was relaxed.

The algorithm terminates when the DM has found a solution which has acceptable values for all the objectives.

The complete STEM algorithm is therefore

1. Set $h = 0$. Let $J = 1 \ldots p, R = \emptyset$. Construct a payoff table. The ith row of the payoff table consists of the solution to the problem

$$\begin{aligned} &\max && z = f_i(\underline{x}) \\ &\text{subject to} && A\underline{x} \leq \underline{b} \\ & && \underline{x} \geq \underline{0} \end{aligned}$$

for $i = 1 \ldots p$.

2. Letting m_i be the minimum value in the ith column of the payoff table, calculate

$$\pi_i = \begin{cases} 0 & \text{if } i \in R \\ \dfrac{z_i^* - m_i}{z_i^*}\left[\sum_{i=1}^{p}(c_{ij})^2\right]^{-\frac{1}{2}} & \text{if } i \notin R \text{ and } z_i^* > 0 \\ \dfrac{m_i - z_i^*}{m_i}\left[\sum_{i=1}^{p}(c_{ij})^2\right]^{-\frac{1}{2}} & \text{if } i \notin R \text{ and } z_i^* \leq 0 \end{cases}$$

3. Using weights $\lambda_i^{(h)} = \dfrac{\pi_i}{\sum_{j=1}^{p}\pi_j}$, solve the weighted minimax problem

$$\begin{aligned} &\max && \alpha \\ &\text{subject to} && \alpha \geq \lambda_i^{(h)}(z_i^* - f_i(\underline{x})) \quad \forall i = 1 \ldots p \\ & && \underline{x} \in X^{(h)} \\ & && \alpha \geq 0 \end{aligned}$$

where $X^{(0)}$ is $\{\underline{x} | A\underline{x} \leq \underline{b}, \underline{x} \geq \underline{0}\}$. Call the solution $\underline{x}^{(h)}$,

4. The DM now compares $f(x^{(h)})$ with z^*. If the DM is satisfied with solution $x^{(h)}$, stop, $x^{(h)}$ is the best compromise solution. Otherwise the DM must relax one objective in order to allow an improvement in unsatisfactory objectives. Let objective j be the objective relaxed, and let δ_j be the amount by which it is relaxed.

5. Reduce the feasible region as follows

$$X^{(h+1)} = \{x \in X^{(h)} \text{ such that } \begin{cases} f_i(x) \geq f_i(x^{(h)}) & \forall i = 1 \ldots p, i \neq j \\ f_j(x) \geq f_j(x^{(h)}) - \delta_j \end{cases}\}$$

6. Set $h = h + 1, R = R \cup \{j\}, J = J - \{j\}$, and go to 3.

2.7.1 Comments

1. The algorithm produces a solution in, at most, p iterations.

2. STEM is basically a structured approach for generating a sequence of improved solutions. There is no guarantee that we will converge to an optimal solution.

3. It is possible that an objective function vector presented to the DM for comparison is dominated, and hence STEM can converge to nonextreme final solutions.

4. The weights give the relative importance of the distances to the optima, but are only locally effective.

5. The technique is based on the premise that the best compromise solution has the minimum combined deviation from the ideal point. However the method assumes that the DM has a pessimistic view or strong dislike of the "worst" component of the objectives. As such, it chooses the worst component of all individual deviations from the ideals of component objectives as the measure of combined deviation from the ideal point (Chankong and Haimes [6], p 325).

2.7.2 Variations

Hwang and Masud [22] report that the Progressive Orientation Procedure (POP) is similar to STEM. In this method the DM is presented with a subset of nondominated extreme points. If the subset contains an acceptable solution, the procedure is terminated; otherwise the DM chooses a best subset of solutions, which is used to determine a new set of non-dominated extreme points, and the process is repeated.

Jordi [23] and others report that Fichefet developed an algorithm which uses goal programming with STEM, called GPSTEM. The advantage is that a satisficing solution can be obtained in fewer iterations.

Shin and Ravindran [49] reports that Slowinsky and Warcynski developed another variation of STEM by using the ellipsoid method for the single objective optimization in STEM.

2.8 The Surrogate Worth Tradeoff Method and its Variants

The Surrogate Worth Tradeoff Method (SWTM), as proposed by Haimes [18], is classified by Chin and Ravindran [49] as a Lagrange multiplier method. It maximizes one objective function subject to a varying set of bounded objectives and uses the resulting Lagrange multipliers of the constraints in an interaction with the DM. The distinguishing feature of the method is the generation of tradeoff functions which show the relationship between the objective function values and the weights on the objectives.

The SWTM consists of four steps:

1. Generate a representative subset of nondominated solutions.
2. Obtain relevant tradeoff information for each generated solution.
3. Interact with the DM to obtain information about his preferences and express these preferences in terms of surrogate worth functions.
4. Retrieve the best compromise solution from these worth functions.

These steps are now described in more detail.

The first step consists of generating a subset of nondominated solutions. While there are many methods which can be used to do this, the ε-constraint method is adopted in view of what is required in step 2.

A reference objective f_k is selected. The ε-constraint problem $P_k(\varepsilon)$ is then formulated as

$$\begin{aligned} \max \quad & z = f_k(\underline{x}) \\ \text{subject to} \quad & f_i(\underline{x}) \geq \varepsilon_i \quad i = 1, \ldots, p, i \neq k \\ & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \end{aligned} \tag{2.9}$$

where ε_i is the minimum acceptable value of the the ith objective function.

To generate a representative subset of nondominated solutions, a reasonable number of values for each ε_i are selected and $P_k(\underline{\varepsilon})$ is solved for each combination of these values ($i \neq k$). As a guideline to ensure that $P_k(\underline{\varepsilon})$ is feasible, each ε_i should be selected such that $z_i^* \leq \varepsilon_i \leq z_i^0$, where $\underline{z}^*$ and $\underline{z}^0$ are the ideal and anti-ideal solutions respectively.

Having generated a number of nondominated solutions, the tradeoff information can be easily obtained merely by observing the optimal Lagrange multipliers corresponding to the ε constraints.

Consider problem (2.9). The Lagrangian is

$$L = f_k(\underline{x}) + \sum_{\substack{i=1 \\ i \neq k}}^{p} \lambda_{ki}(f_i(\underline{x}) - \varepsilon_i) + \sum_{j=1}^{m} \mu_{kj}\left(\sum_{i=1}^{n} a_{ij}x_j - b_j\right) \qquad (2.10)$$

where λ_{ki} and μ_{kj} are the generalised Lagrange multipliers. The subscript ki in λ_{ki} denotes that λ is a Lagrange multiplier associated with the ith constraint, where the objective function is $f_k(\underline{x})$.

From equation (2.10) we have $\dfrac{\partial L}{\partial \varepsilon_i} = -\lambda_{ki}$ or $\lambda_{ki} = -\dfrac{\partial L}{\partial \varepsilon_i}$.

Recalling the Kuhn-Tucker conditions,

$$\lambda_{ki}(f_i(\underline{x}) - \varepsilon_i) = 0, \quad \lambda_{ki} \geq 0$$

$$\mu_{kj}\left(\sum_{i=1}^{n} a_{ij}x_j - b_j\right) = 0, \quad \mu_{kj} \geq 0$$

It can be shown that

$$\frac{\partial f_k(\underline{x})}{\partial \varepsilon_i} = \frac{\partial L}{\partial \varepsilon_{ki}} = -\lambda_{ki} \quad \text{for } i, k = 1 \cdots p, i \neq k$$

Note that only those λ_{ki} which are positive are of interest. If $\lambda_{ki} = 0$, then $\frac{\partial f_k(\underline{x})}{\partial \varepsilon_i} = 0$ which means that decreasing the level of the ith objective does not

result in any improvement in the kth objective, which implies that this solution is dominated. On the other hand, if $\lambda_{ki} > 0$, then $\frac{\partial f_k(\underline{x})}{\partial \varepsilon_i} < 0$ which means that decreasing the level of the ith objective results in an improvement in the kth objective, which implies that this solution belongs to the nondominated set.

Hence only the cases where $\lambda_{ki} > 0$ are of interest. Now since $\lambda_{ki} > 0$ if and only if $f_i(\underline{x}) = \varepsilon_i$, then

$$\frac{\partial f_k(\underline{x})}{\partial \varepsilon_i} = \frac{\partial f_k(\underline{x})}{\partial f_i} = -\lambda_{ki}$$

Hence, by considering one objective function as primary and all others at minimum satisfying levels, the set of nonzero Lagrange multipliers represents the set of tradeoff ratios between the principal objective and each of the constraining objectives respectively at the solution which solves problem $P_k(\underline{\varepsilon})$.

The following relationships apply to the tradeoff ratios

$$\lambda_{ij} = \lambda_{ik}\lambda_{kj}$$

$$\lambda_{ij} = \frac{1}{\lambda_{ji}}$$

Hence not all the λ_{ki} need be generated computationally, but can be derived from the above relationships. Alternatively, one can generate all λ_{ki} and use the above relationships to check the consistency of the DM.

The Surrogate Worth Functions (SWF) provide the interface between the DM and the method, and are the means by which the tradeoffs are evaluated. For each tradeoff function, there exists a corresponding SWF. The SWF's are designated by W_{ij} and are ordinal in nature.

The value of W_{ij} is the DM's assessment of how much (on an ordinal scale, say from -10 to 10 with 0 signifying indifference) he prefers trading λ_{ij} marginal units of f_i for one marginal unit of f_j, given the values of all the objectives $f_1 \cdots f_n$ corresponding to λ_{ij}.

The indifference band is then constructed as follows : the DM is asked whether he likes, is indifferent to, or dislikes trading λ_{ki} units of f_i for one unit of f_k for various distinct values of λ_{ki}. This is assessed on an ordinal scale $W_{ki}(\lambda_{ki})$. λ_{ki}^* is then approximated by finding $W_{ki}(\lambda_{ki}^*) = 0$ using regression or some other curve fitting technique. If only two values of λ_{ki} are used, then linear interpolation can be used. At the solutions λ_{ki}^*, the SWFs are simultaneously equal to zero, i.e. marginal gain=marginal loss for all objectives taken two at a time.

The final step in the SWTM is to determine an $\underset{\sim}{x}^*$ that corresponds to all λ_{ki}^*.

For each λ_{ki}^* determined, there corresponds $f_i^*(\underset{\sim}{x})$. These $f_i^*(\underset{\sim}{x})$ are the values of $f_i(\underset{\sim}{x})$ when the ϵ_i constraints are active, so $\lambda_{ki}[f_i^*(\underset{\sim}{x}) - \epsilon_i] = 0$.

Accordingly, $\underset{\sim}{x}^*$ can be obtained by solving

$$\begin{aligned} \max \quad & z = f_k(\underset{\sim}{x}) \\ \text{subject to} \quad & f_i(\underset{\sim}{x}) \geq f_i^*(\underset{\sim}{x}) \quad \forall i = 1 \ldots p, i \neq k \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

Hence, the complete SWTM algorithm is as follows

1. Transform the problem to

$$\begin{aligned} \max \quad & z = f_k(\underset{\sim}{x}) \\ \text{subject to} \quad & f_i(\underset{\sim}{x}) \geq \epsilon_i \quad \forall i = 1 \ldots p, i \neq k \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned} \tag{2.11}$$

   where ϵ is a lower bound on the ith objective function.

2. One of the objectives expressed as a constraint, say objective i, is varied over q values of ϵ_i, and the other $p-2$ objectives are fixed at ϵ_j. $(j = 1 \ldots p, j \neq k, j \neq i)$.

3. (2.11) is now solved for each value of ϵ_i, producing at most q nondominated solutions. The dual variable associated with the constraint for the

ith objective, when the kth objective is the objective function, is λ_{ki}. As shown, λ_{ki} is the tradeoff between objectives k and i.

4. Now another of the objectives expressed as a constraint is selected as objective i and step 3 is repeated until λ_{ki} is generated for all $i = 1, 2, \ldots, k-1, k+1, \ldots, p$.

5. Now a different objective is selected for the objective function. Steps 3 and 4 are repeated until λ_{ki} is generated for all $k = 1 \ldots p$ and $i = 1, 2, \ldots, k-1, k+1, \ldots, p$. The result is a set of objective functions which relate the weights to the levels of objectives.

6. The λ_{ki} are presented to the DM. The DM must give an assessment of how much he likes trading λ_{ki} units of f_k for 1 unit of f_i. This assessment is called the surrogate worth W_{ki}. The W_{ki} are ordinal varying between -10 and 10, 0 representing indifference.

7. Regression analysis (or some other curve-fitting procedure) is used to evaluate the optimal tradeoffs $\underset{\sim}{\lambda}^*$. $\underset{\sim}{\lambda}^*$ is an estimate of the tradeoffs which the DM considers to have surrogate worth 0.

8. $\underset{\sim}{\lambda}^*$ is used to identify the best compromise solution by solving the following LP problem.

$$\begin{aligned} \max \quad & z = f_k(\underset{\sim}{x}) \\ \text{subject to} \quad & f_i(\underset{\sim}{x}) \geq f_i^*(\underset{\sim}{x}) \quad \forall i = 1, \ldots p, i \neq k \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

2.8.1 Comments.

1. The consistency of the DM can be crosschecked by using the relation $\lambda_{ij} = \lambda_{ik}\lambda_{kj}$.

2. Hwang and Masud [22] comment that as the number of objective functions increase, the number of tradeoffs which are required from the DM becomes excessive.

3. The approximations of the worth function may not be good enough to generate indifference solutions.

4. It is possible to exclude the most preferred solution while varying the ε's parametrically.

5. The DM may not perceive that the λ_{ij}'s indicate marginal tradeoffs in a very small region around the current solution, when he is providing the surrogate worth functions.

6. The method requires the DM to consider only two objectives at a time while assessing the worth functions, even if there are many objectives in the problem.

7. There is only one major interaction with the DM to obtain the worth functions.

2.8.2 Variations

Chankong and Haimes [6] proposed an interactive version of the SWTM. This algorithm proceeds in the same way as the SWTM until the the worth functions have been assessed. These worth function values are then used to estimate a new $\underline{\varepsilon}^{(h+1)}$ so that $\underline{x}^{(h+1)}$ (which is a solution to $P(\underline{\varepsilon}^{(h+1)})$) will be an improved solution.

Assuming W_{ki} is positive, then the DM likes the tradeoff λ_{ki}, where f_k is the maximizing objective. He therefore would like to decrease f_k by λ_{ki} units per one unit increase in f_i. Hence, let $\varepsilon_i^{(h+1)} = \varepsilon_i^{(h)} + \delta f_i \quad (\delta f_i \geq 0)$.

Similarly, if W_{ki} is negative, then the DM does not like the tradeoff λ_{ki} and therefore let $\varepsilon_i^{(h+1)} = \varepsilon_i^{(h)} - \delta f_i \quad (\delta f_i > 0)$.

Chankong and Haimes [6] also established the relationship between the surrogate worth functions and the DM's utility function U and utilised this to develop schemes for determining the search direction, step size and some stopping criteria, thereby imbedding some features of the Geoffrion-Dyer-Feinberg method.

Sakawa [44] combined the SWTM with Oppenheimer's proxy approach (see section 2.1.2) to form the Sequential Proxy Optimization Technique (SPOT). In SPOT, the SWTM is used to find a nondominated solution and to compute the generalised Lagrange multipliers (tradeoff functions). A proxy function is then fitted to the MRS at this solution, which are provided by the DM. The function is used to determine a direction of improvement and the step size is then computed numerically to find a new nondominated solution. The algorithm is continued until the Lagrange multipliers and the MRS at the current solution are sufficiently close.

Haimes [18] describes the Multiobjective Statistical Method (MSM) in which the SWTM is extended to incorporate statistical data.

Chankong *et al.* [7] (p. 63) list a number of other developments and applications of the SWTM.

2.9 The Branch-and-Bound Method and its Variants

Marcotte and Soland [35] apply the Branch and Bound concept to MOLP, as follows.

1. Solve
$$\begin{aligned} \max \quad & U(f(x)) \\ \text{s.t.} \quad & x \in X_i \end{aligned}$$
where $X_i \subseteq X$. Initially $X_i = X$.

2. Using the solution obtained in 1 above, partition X_i into a finite number of subsets of X_i whose union is X_i. This is the branching (or separating) step.

3. Calculate for each subset created, an upper bound on the value of each objective function in that subset. This is the bounding step.

4. Select one of the subsets created in step 2 for further investigation and go to 1.

Consider first the method of branching. Designate N_j the node currently being examined, X_j the subset of X corresponding to N_j. Denote z_j^* as the ideal of X_j. The first step of the branching consists of finding an efficient solution in X_j. The simplest way to do this is to solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i f_i(x) \\ \text{s. t.} \quad & x \in X_j \end{aligned}$$

for some $\lambda_i > 0$, $i = 1 \ldots p$. Call the solution to this problem $x^{(h)}$.

Define the set $I_j = \{k \mid 1 \leq k \leq p,\ z_{jk}^* - f_k(x) > \delta_k\}$. $\delta_k > 0$ represents some band of indifference for objective k.

The case where I_j is empty can be ignored since in this case $x^{(h)}$ coincides with the ideal of node N_j so $x^{(h)}$ is the solution preferred by the DM. When

I_j is not empty, every efficient point $\underline{x}$ of X_j is such that for at least one index $h \in I_j$, $f_h(\underline{x}) \geq f_h(\underline{x}^{(h)})$.

Consequently, if the DM's preferred decision belongs to X_j, it will necessarily belong to the set

$$\{\underline{x}^{(h)}\} \cup [\bigcup_{h \in I_j} (X_j \cap \{\underline{x} \mid f_h(\underline{x}) > f_h(\underline{x}^{(h)})\})]$$

This method of branching results in each node of the branch-and-bound tree representing a set of the form $X \cap \{\underline{x} \mid \underline{f}(\underline{x}) \geq \underline{b}\}$ for some vector $\underline{b}$.

Consider now the question of the order in which the nodes will be examined and under what conditions a node will be rejected.

At each separation, the ideals of the newly created nodes are calculated. The DM is asked to arrange the ideals of all nodes in the node list in preferential order. Marcotte and Soland [35] suggest that in order to make this ordering easier for the DM, a master list and several partial lists can be kept.

The algorithm can terminate in one of the following three ways :

1. the ideal of the first node of the master list is feasible
2. the ideal of the first node of the master list is not preferred to the incumbent solution
3. the master list is empty.

Marcotte and Soland [35] also include in the algorithm a test which permits the elimination of superfluous nodes.

The branch-and-bound algorithm is therefore as follows.

1. Set $h = 0, j = 0, l = 0, X^{(0)} = X_0 = X$. Calculate the ideal of $X^{(0)}$ and denote it z_0^*.

2. Solve the LP problem

$$\max \sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x})$$
$$\text{s. t.} \quad \underset{\sim}{x} \in X^{(h)}$$

for some positive λ_i's. Call the solution $\underset{\sim}{x}^{(h)}$.

3. Define $I_j = \{k \mid 1 \leq k \leq p, z_{jk}^* - f_k(\underset{\sim}{x}^{(h)}) > \delta_k\}$. For each $k \in I_j$, set $l := l + 1$, define a node N_l with corresponding $X_l = X_j \cap \{\underset{\sim}{x} \mid f_k(\underset{\sim}{x}) \geq f_k(\underset{\sim}{x}^{(h)})\}$ and add it to the node list.

4. Calculate ideal solutions for all newly created X_j's and denote them z_j^*. Ask the DM to order all the nodes in the node list in order of preference of their ideals.

5. If the ideal of the first node of the list is feasible, or if the ideal of the first node of the list is not preferred to the incumbent solution, or if the list is empty, stop - the incumbent solution is optimal.

6. Otherwise, set j to the subscript of the node at the top of the list. Make N_j the incumbent solution and remove it from the node list. Let $X^{(h+1)} = X_j$. Set $h = h + 1$ and go to 2.

2.9.1 Comments

1. The only hypotheses made about the DM's preferences are those that ensure the existence of indifference curves.

2. The branch-and-bound method provides the DM with an efficient solution even if the DM decides to stop before the algorithm terminates.

3. Marcotte and Soland [35] claim that the method "is not an adaptation of some existing algorithm to an interactive mode, but rather the logic seems to follow from the very nature of MCDM".

2.9.2 Variations.

None encountered.

2.10 The Interactive Relaxation Method and its Variants

The Interactive Relaxation Method of Lazimy [31] is the only one classified by Chin and Ravindran [49] as a relaxation algorithm. It solves the MOLP problem iteratively, with one problem, a relaxed master problem (RMP), being solved at each iteration. The RMP is a relatively simple problem since its constraints are linear in x.

Lazimy [31] shows that the problem

$$\begin{aligned} \max \quad & z = U(\underline{f}(\underline{x})) \\ \text{subject to} \quad & \underline{x} \in X \end{aligned} \tag{2.12}$$

is equivalent to the problem

$$\begin{aligned} \max \quad & \theta \\ \text{subject to} \quad & \theta \leq U(\underline{f}(\underline{y})) - (\underline{y} - \underline{x})^T \nabla U(\underline{f}(\underline{y})) \qquad \forall \underline{y} \geq 0 \\ & \underline{\xi}(A\underline{x} - \underline{b}) - (\underline{y} - \underline{x})^T \nabla \underline{\xi}(A\underline{y} - \underline{b}) \leq 0 \quad \forall \underline{y}, \underline{\xi} \geq 0 \\ & \underline{x} \in X \end{aligned} \tag{2.13}$$

in the sense that the optimal solution to the two problems is the same.

Call problem (2.13) the master problem (MP) and refer to the constraints

$$\theta \leq U(\underline{f}(\underline{y})) - (\underline{y} - \underline{x})^T \nabla U(\underline{f}(\underline{y})) \quad \forall \underline{y} \geq 0$$

of MP as constraints of type I and constraints

$$\underline{\xi}(A\underline{x} - \underline{b}) - (\underline{y} - \underline{x})^T \nabla \underline{\xi}(A\underline{y} - \underline{b}) \leq 0 \quad \forall \underline{y}, \underline{\xi} \geq 0$$

as constraints of type II.

Since MP includes an infinite number of constraints, it will be solved by a relaxation process.

Suppose that some of the constraints of MP are relaxed (i.e. temporarily ignored). Define RMP to be the relaxed resulting version MP where RMP includes only a few constraints of type I and II. RMP is then solved.

The following observations can be made about the relationship between MP and RMP.

If RMP is not feasible at some iteration, neither is MP.

If RMP is feasible, then each iteration of the relaxation produces a new trial solution $(\theta, \underline{x})$ for MP. If an optimal solution $(\theta, \underline{x})$ of RMP satisfies all of the relaxed constraints of MP, then $(\theta, \underline{x})$ is optimal for MP. If $(\theta, \underline{x})$ is not feasible for MP, then one of the relaxed constraints of MP is forced into the relaxed program RMP and the process is repeated.

Relaxation is an effective strategy for solving the master problem since typically only a few of the constraints of MP will be binding at the optimal solution.

Consider the reasons for adding new constraints.

If $\underline{x}^{(k)}$ is feasible for problem (2.12), (i.e. $\sum_{j=1}^{n} a_{ij}x_j^{(k)} > 0$ for some i) but $\theta^{(k)} > U(\underline{f}(\underline{x}^{(k)}))$ then $\theta^{(k)} > U(\underline{f}(\underline{y}^{(k)})) - (\underline{y} - \underline{x})^T \nabla_{\underline{y}} U(\underline{f}(\underline{y}))$ for some $\underline{y}$. Consequently, $(\theta^{(k)}, \underline{x}^{(k)})$ violates some of the ignored constraints of type I of the master problem (MP). In order to satisfy the most violated constraints of this type, set $\underline{y}^{(k)} = \underline{x}^{(k)}$ and add the following constraint to problem RMP

$$\theta - \underline{x}\nabla U(\underline{f}(\underline{x}^{(k)})) \leq U(\underline{f}(\underline{x}^{(k)})) - \underline{x}^{(k)}\nabla U(\underline{f}(\underline{x}^{(k)}))$$

If $\underline{x}^{(k)}$ is not feasible, then $\underline{x}^{(k)}$ violates some of the constraints of type II of problem MP and there exists $\underline{y}, \xi \geq 0$ such that $\xi(A\underline{y} - \underline{b}) - (\underline{y} - \underline{x}^{(k)})\nabla\xi(A\underline{x} - \underline{b}) > 0$. Set $\underline{y}^{(k)} = \underline{x}^{(k)}$ and $\xi^{(k)}$ such that $\xi_i^{(k)} = 1$ for each i such that $\sum_{j=1}^{n} a_{ij}x_j - b_j > 0$ and $\xi_i^{(k)} = 0$ otherwise. Then the admissible point $\underline{x}^{(k)}$ is eliminated by adding the following constraint to RMP

$$\underline{x}\nabla\xi^{(k)}[A\underline{x}^{(k)} - \underline{b}] \leq \underline{x}^{(k)}\nabla\xi^{(k)}[A\underline{x}^{(k)} - \underline{b}] - \xi^{(k)}[A\underline{x}^{(k)} - \underline{b}]$$

The algorithm is therefore

1. If $A\underline{x}^{(h)} \leq \underline{b}$ (i.e. $\underline{x}^{(h)}$ is feasible) go to 2, else go to 4.

2. If $\underline{\theta}^{(h)} \leq U(\underline{f}(\underline{x}^{(h)}))$ then stop, $\underline{x}^{(h)}$ is an optimal solution of problem (2.12). Otherwise, go to 3.

3. If $\underline{\theta}^{(h)} > U(\underline{f}(\underline{x}^{(h)}))$ then add the constraint

$$\underline{\theta} - \underline{x}\nabla U(\underline{f}(\underline{x}^{(h)})) \leq U(\underline{f}(\underline{x}^{(h)})) - \underline{x}^{(h)}\nabla U(\underline{f}(\underline{x}^{(h)}))$$

to problem RMP and go to 5.

4. Set $\varepsilon_i^{(h)} = 1$ for all i such that $\sum_{j=1}^{n} a_{ij}x_j^{(h)} > 0$ and $\varepsilon_i^{(h)} = 0$ otherwise. Add the constraint

$$\underline{x}\nabla\underline{\xi}^{(h)}[A\underline{x}^{(h)} - \underline{b}] \leq \underline{x}^{(h)}\nabla\underline{\xi}^{(h)}[A\underline{x}^{(h)} - \underline{b}] - \underline{\xi}^{(h)}[A\underline{x}^{(h)} - \underline{b}]$$

to RMP and go to step 5.

5. Solve the revised relaxed master problem RMP and let $(\underline{\theta}^{(h+1)}, \underline{x}^{(h+1)})$ be its optimal solution. Set $h = h + 1$ and go to step 1.

2.10.1 Comments

1. Reference assessments are only required when constraints of type I are added — i.e. they are not required at each iteration.

2. The algorithm is capable of predicting upper and lower bounds on the optimal objective value of the problem at each iteration. These bounds can be used to terminate the search prematurely at a satisficing solution, and consequently, to reduce the computational time as well as the number of preference assessments the DM must make.

3. All tradeoffs the DM must make are in the form of ordinal paired-comparisons, most of which are simple since they involve simple changes in only 2 criteria at a time. Also the preference assessments are made in the contex of feasible solutions only.

4. Constraints of type II do not eliminate any feasible solutions of problem (2.12).

5. The procedure for obtaining the necessary information from the DM is the Geoffrion-Dyer-Feinberg marginal rates of substitution method.

2.10.2 Variations

None encountered.

2.11 SEMOPS and its Variants

The Sequential Multiobjective Problem Solving (SEMOPS) Method, developed by Monarchi *et al.* [37], sequentially uses a surrogate objective function based on the goals and their aspiration levels elicited from the DM. It is a nonlinear interactive method that relies on minimization of deviation from the given goal levels. It differs however from approaches such as goal programming in that the goal levels elicited from the DM may be intervals rather than fixed points.

Instead of p objective functions, there are p goals. Associated with each of these goals, is a measure of the deviation from the goal which is used to judge the attainment or non attainment of goals. Since aspiration levels may be intervals, the norm is not used as measure of deviation.

The range of f_i is denoted by $\Gamma_i = [f_{iL}, f_{iU}]$ which is not necessarily defined by the maximum and minimum values of the ith objective function.

Nondimensionality is achieved by transforming $f_i(\underline{x})$ into $y_i(\underline{x})$ with a range of values in the interval $[0, 1]$ such that

$$y_i(\underline{x}) = \frac{f_i(\underline{x}) - f_{iL}}{f_{iU} - f_{iL}} + \varepsilon$$

where ε is small and positive and added to circumvent division by 0 later.

Similarly, if $AL = (AL_1, \ldots, AL_p)$ denotes the DM's aspiration levels, the transformation

$$A_i = \frac{AL_i - f_{iL}}{f_{iU} - f_{iL}} + \varepsilon$$

can be used to define A_i with values in the range $[0, 1]$.

The method permits five types of goals:

1. at most $(f_i(\underline{x}) \leq AL_i)$
2. at least $(f_i(\underline{x}) \geq AL_i)$

3. equals $(f_i(\underline{x}) = AL_i)$

4. within an interval $(AL_{iL} \leq f_i(\underline{x}) \leq AL_{iU})$

5. outside an interval $(f_i(\underline{x}) \leq AL_{iL}$ or $f_i(\underline{x}) \geq AL_{iU})$

Types 1, 2 and 4 are the most common.

By an appropriate transformation, the above goal types are transformed into a function $\underline{d}$ with real values in the set of real positive numbers. In each case, a value of $d_i \leq 1$ implies that goal i is satisfied. Specifically $\underline{d}$ is defined as a "dimensionless indicator of attainment" for the p goals.

There are five types of transformations corresponding to the five types of goals mentioned above.

These are :-

1. $d_i = \dfrac{y_i(\underline{x})}{A_i}$

2. $d_i = \dfrac{A_i}{y_i(\underline{x})}$

3. $d_i = \dfrac{1}{2}\left[\dfrac{A_i}{y_i(\underline{x})} + \dfrac{y_i(\underline{x})}{A_i}\right]$

4. $d_i = \left[\dfrac{A_{iU}}{A_{iL} + A_{iU}}\right]\left[\dfrac{A_{iL}}{y_i(\underline{x})} + \dfrac{y_i(\underline{x})}{A_{iL}}\right]$

5. $d_i = \left[\dfrac{A_{iL} + A_{iU}}{A_{iU}}\right]\left[\dfrac{A_{iU}}{y_i(\underline{x})} + \dfrac{y_i(\underline{x})}{A_{iU}}\right]^{-1}$

Except for the first transformation, these transformations are all nonlinear functions of an objective function. The critical fact in this construction is that the direction of change in $\underline{d}$ has a consistent interpretation for all p goals.

Without loss of generality, assume that all goals are of the "at least" type.

At each iteration, the DM is asked not only to provide the interval goal levels, but also to specify which objectives are to be kept strictly within stated goals. Let $J = 1, \ldots, p$. If the DM specifies the index set $I^{(h)}$ such that all goals in $I^{(h)}$ are to be kept strictly within the specified goals, then the surrogate problem to be solved at the hth iteration is

$$\begin{aligned} \min \ & S^{(h)} = \sum_{i \in J^{(h)}} d_i \\ \text{s. t.} \ & x \in X^{(h)} \end{aligned}$$

where $J^{(h)} = J - I^{(h)}$ and $X^{(h)} = \{x \in X | f_j(x) \geq AL_j, \forall j \in I^{(h)}\}$

This problem is solved for the hth iterate $x^{(h)}$ and $f_1(x^{(h)}), \ldots, f_p(x^{(h)})$ are calculated. These objective function values are presented to the DM and new goal level specifications and $I^{(h+1)}$ are then elicited for the next iteration. The process is terminated whenever the DM is satisfied with the current solution.

Monarchi *et al.* [37] also suggested that additional information may also be presented to the DM to assist in finding the new goal levels. This information is obtained by solving as many auxiliary subproblems as the number of elements in $J^{(h)}$. These auxiliary problems are defined as follows.

$$\begin{aligned} \min \ & S_l^{(h)} = \sum_{i \in J^{(h)}, i \neq l} d_i^{(h)} \\ \text{s. t.} \ & x \in X^{(h)} \\ & f_l(x) \geq AL_l \end{aligned}$$

for each l in $J^{(h)}$.

In the first iteration, $I^{(h)}$ is set to $\emptyset$, i.e. no goals are forced to be satisfied.

The complete algorithm is therefore

1. Transform $f_i(x)$ to $y_i(x)$. Set $I^{(0)} = \emptyset$ and $h = 0$.
2. Obtain the aspiration levels AL_i from the DM and transform them to A_i.

8. Solve the principal problem

$$\min \; S^{(h)} = \sum_{i \in J^{(h)}} d_i$$
$$\text{s. t. } x \in X^{(h)}$$

where $J^{(h)} = J - I^{(h)}$ and $X^{(h)} = \{x \in X | f_j(x) \geq AL_j, \; \forall j \in I^{(h)}\}$ and auxiliary subproblems

$$\min \; S_l^{(h)} = \sum_{i \in J^{(h)}, i \neq l} d_i^{(h)}$$
$$\text{s. t. } x \in X^{(h)}$$
$$f_l(x) \geq AL_l$$

for each $l <> J^{(h)}$.

4. Present the resulting solutions to the DM. If the DM is satisfied with one of the solutions presented to him, stop with that solution as best compromise solution. Otherwise, the DM specifies new goal level specifications and $I^{(h+1)}$ – the set of goals which are to be kept strictly within the specified goals. Set $h = h + 1$ and go to 2.

2.11.1 Comments

1. SEMOPS does not solve problems; it generates information so that the DM can select an acceptable alternative.
2. As in goal programming, the best compromise solution obtained by this method is not necessarily efficient.
3. Unless mathematical simplifications are made, this method may convert a linear problem into a nonlinear problem.
4. An advantage of the method is that the DM can re-evaluate his aspiration levels at each iteration.
5. If the relative range $[f_{iL}, f_{iU}]$ is too narrow, then $y_i(x)$ can become negative or greater than one, and hence d_i becomes negative or greater than one. On the other hand, if the relative range is much wider than the values actually achieved by the criterion function, then d_i will be relatively insensitive to alternative solutions.

6. Hwang and Masud [22] comment that a very undesirable aspect of this approach is the possibility of finding inconsistent constraint sets when solving auxiliary problems. In such a situation, the determination of a consistent set of aspiration levels becomes random and unsystematic.

2.11.2 Variations

The Sequential Information Generator for Multiobjective Problems (SIGMOP) proposed by Monarchi, Weber and Duckstein [38] replaces the surrogate objective function in SEMOPS and imbeds a nonlinear goal programming approach within the principal problem and the set of auxiliary problems.

2.12 Other Approaches

Sakawa *et al.* [45] comment that there are three major approaches in finding the best-compromise solution to MOLF problems:

1. the goal programming approach
2. the interactive approach
3. the fuzzy programming approach

While the emphasis of this document is on interactive methods, it was decided to include a brief overview on the goal programming and fuzzy programming approaches, in the interests of completeness. In any case, goal programming and fuzzy programming approaches have, of late, become increasingly interactive.

2.12.1 Goal Programming

The original concept of goal programming, as introduced by Charnes and Cooper, was effectively just a method for setting the values of all objectives (cf. section 1.3.1). Goal programming differs from linear programming in that instead of maximizing objectives, the difference between the objective function values and some target objective values are minimised.

i.e. the problem

$$\begin{aligned} &\max && f_1(\underline{x}) \\ &\max && f_2(\underline{x}) \\ &&& \vdots \\ &\max && f_p(\underline{x}) \\ &\text{subject to,} && A\underline{x} \leq \underline{b} \\ &&& \underline{x} \geq \underline{0} \end{aligned}$$

becomes

$$\begin{aligned}
\text{min} \quad & \sum_{i=1}^{p} w_i d_i^- \\
\text{subject to} \quad & f_1(\underline{x}) + d_1^- - d_1^+ = q_1 \\
& f_2(\underline{x}) + d_2^- - d_2^+ = q_2 \\
& \vdots \\
& f_p(\underline{x}) + d_p^- - d_p^+ = q_p \\
& A\underline{x} \leq \underline{b} \\
& d_i^- \cdot d_i^+ = 0 \\
& \underline{x} \geq \underline{0} \\
& d_i^+, d_i^- \geq 0 \forall i = 1 \ldots p
\end{aligned}$$

where the w_i are weights to represent the importance of the objectives, the q_i are the minimum levels the DM would like the objectives to achieve, and the d_i^- and d_i^+ are the negative and positive deviations from these goal levels respectively.

The above goal programming problem can be solved using the simplex algorithm for linear programming.

A variation of the above goal programming approach employs preemptive priorities instead of numerical weights when certain of the objectives are far more important than others. The objectives are partitioned into groups, where the objectives in group 1 are far more important than those in group 2, those in group 2 are far more important than those in group 3, and so on.

A weighted sum of the deviation variables in group 1 is minimised. This weighted sum is constrained to be equal to its minimum value, and the weighted sum of the deviational variables in group 2 is minimised, and so on, for as many preemptive priority groups as there may be.

These versions of goal programming, are methods of a priori articulation of preferences.

Interactive Goal Programming (IGP) was developed with the idea that a priori articulation of preferences is insufficient. On seeing the solution resulting from some goal levels, the DM may be willing to alter some goal levels in order to

bring other objective functions closer to their target levels.

Various interactive goal programming methods have been proposed, one of the more well-known being the following algorithm proposed by Masud and Hwang [36] in 1981.

1. Obtain goals q_i from the DM where $z_i^* \leq q_i \leq z_i^{**}$.

2. Solve the goal programming problem

$$\begin{aligned} \min \quad & \{\sum_{i=1}^{p} d_i^-, \sum_{i=1}^{p} d_i^+\} \\ \text{subject to} \quad & f_i(\underline{x}) + w_i d_i^- - w_i d_i^+ = q_i \qquad \forall i = 1 \ldots p \\ & d_i^- \leq 1 \qquad \forall i = 1 \ldots p \\ & d_i^- \cdot d_i^+ = 0 \qquad \forall i = 1 \ldots p \\ & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \\ & d_i^+, d_i^- \geq 0 \qquad \forall i = 1 \ldots p \end{aligned}$$

   The first objective is to achieve the goal levels and the second is to achieve a nondominated solution.

3. Solve the above problem a further p times with d_i^- set to zero in turn for $i = 1 \ldots p$ so as to generate solutions which satisfy individual goals.

4. Create a tradeoff table comprising $\underline{z}^*, \underline{z}^{**}$, the DM-specified goal levels, and the solutions to the $p - 1$ problems solved above. If the DM does not like any of the $p + 1$ solutions, he is asked to modify $\underline{q}$ from the information in the table.

5. Solve a new goal programming problem where the first priority is to attain those goals which were reduced by the DM, the second priority is to satisfy the other goals, and the third priority is to ensure nondominance of the solution.

6. Compute a further p solutions by setting d_i^- to zero in turn for $i = 1 \ldots p$.

7. A new tradeoff table is constructed.

The procedure continues until the DM finds a nondominated solution from the table which he considers to be satisfactory.

2.12.2 Fuzzy Programming

The fuzzy programming approach is based on the premise that the decision maker's objectives or goals are often vague. For example the DM may wish to achieve a profit 'somewhat larger' than R 100 000. What constitutes 'somewhat larger' is vague, and this type of statement is therefore quantified by a membership function.

Associated with each objective is a membership function which has value 0 when that objective is at a totally unsatisfactory level, 1 when the DM is completely satisfied with the level of that objective, and somewhere in between otherwise. The membership function is, in effect, a measure of the DM's satisfaction with the objective function level.

In other words, if z_i^0 is an unacceptable level for $f_i(\underline{x})$ and z_i^1 is a totally desirable level for $f_i(\underline{x})$, then $\mu_{f_i}(\underline{x})$ is defined by

$$\mu_{f_i}(\underline{x}) = \begin{cases} 1 \text{ or } \to 1, & \text{if } f_i(\underline{x}) > z_i^1 \\ d_i(f_i(\underline{x})), & \text{if } z_i^0 \leq f_i(\underline{x}) \leq z_i^1 \\ 0 \text{ or } \to 0, & \text{if } f_i(\underline{x}) < z_i^0 \end{cases}$$

where $d_i(f_i(\underline{x}))$ is a strictly monotone decreasing and continuous function with respect to $f_i(\underline{x})$ and may be linear or nonlinear.

Sakawa *et al.* [45] cite five examples of membership functions :

1. linear

$$d_i(f_i(\underline{x})) = \frac{f_i(\underline{x}) - z_i^1}{z_i^1 - z_i^0}$$

The linear membership function can be determined by asking the DM to specify two points z_i^0 and z_i^1 within z_i^* and z_i^*.

2. exponential

$$d_i(f_i(\underline{x})) = a_i\left(1 - e^{-\alpha_i \frac{f_i(\underline{x}) - z_i^0}{z_i^* - z_i^0}}\right)$$

The exponential membership function can be determined by asking the DM to specify the three points $z_i^0, z_i^{0.5}$ and z_i^1 within z_i^0 and z_i^*, where α_i is a shape parameter and z_i^* represents the value of $f_i(\underline{x})$ such that the degree of membership function $\mu_{f_i}(\underline{x})$ is a.

3. hyperbolic

$$d_i(f_i(\underline{x})) = \frac{1}{2}\tanh(\alpha_i(f_i(\underline{x}) - b_i)) + \frac{1}{2}$$

The hyperbolic membership function can be determined by asking the DM to specify the two points $z_i^{0.25}$ and $z_i^{0.5}$ within z_i^0 and z_i^*, where α_i is a shape parameter and b_i is associated with the point of inflection.

4. hyperbolic-inverse

$$d_i(f_i(\underline{x})) = a_i \tanh^{-1}(\alpha_i(f_i(\underline{x}) - b_i)) + \frac{1}{2}$$

The hyperbolic-inverse membership function can be determined by asking the DM to specify the three points z_i^0, $z_i^{0.25}$ and $z_i^{0.5}$ within z_i^0 and z_i^*, where α_i is a shape parameter and b_i is associated with the point of inflection.

5. piecewise-linear

$$d_i(f_i(\underline{x})) = t_{ir} f_i(\underline{x}) + s_{ir} \quad \text{for } g_{ir-1} \le f_i(\underline{x}) \le g_{ir},\ r = 1, \cdots, N_{i+1}$$

That is, it is assumed that $d_i(f_i(\underline{x})) = t_{ir} f_i(\underline{x}) + s_{ir}$ for each segment $g_{ir-1} \le f_i(\underline{x}) \le g_{ir}$, where t_{ir} is the slope and s_{ir} is the y intercept for the section of the curve between g_{ir-1} and g_{ir}. The piecewise-linear membership function can be determined by asking the DM to specify the degree of membership in each of several values of objective function within z_i^0 and z_i^*.

Having elicited the membership functions $\mu_{f_i(\underline{x})}$ from the DM for each of the p objective functions $z_i(\underline{x})$, the MOLP problem

$$\begin{aligned} \max \quad & \underline{z} = \underline{f}(\underline{x}) = [f_1(\underline{x}), f_2(\underline{x}), \cdots, f_p(\underline{x})] \\ \text{subject to} \quad & A\underline{x} \le \underline{b} \\ & \underline{x} \ge \underline{0} \end{aligned} \tag{2.14}$$

is converted into the fuzzy MOLP (FMOLP) problem defined by

$$\begin{aligned} \max \quad & [\mu_{f_1}(\underset{\sim}{x}), \mu_{f_2}(\underset{\sim}{x}), \cdots, \mu_{f_p}(\underset{\sim}{x})] \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned} \tag{2.15}$$

By introducing a general aggregation function $\mu_D(\mu_{f_i})$, problem (2.15) is converted to

$$\begin{aligned} \max \quad & \mu_D(\mu_{f_1}(\underset{\sim}{x}), \mu_{f_2}(\underset{\sim}{x}), \cdots, \mu_{f_p}(\underset{\sim}{x})) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned} \tag{2.16}$$

The value of $\mu_D(\mu_{f_i}(\underset{\sim}{x}))$ can be interpreted as the overall degree of satisfaction of the DM's fuzzy goals, and as such, is analogous to the utility function for objectives.

Sakawa *et al.* [45] comment that in the conventional fuzzy approaches, it has been implicitly assumed that the minimum operator

$$\min_{1 \leq i \leq p} [\mu_{f_1}(\underset{\sim}{x}), \mu_{f_2}(\underset{\sim}{x}), \ldots, \mu_{f_p}(\underset{\sim}{x})]$$

is the proper representation of human decision makers' fuzzy preferences, and hence (2.16) becomes

$$\begin{aligned} \max \min_{1 \leq i \leq p} \quad & [\mu_{f_1}(\underset{\sim}{x}), \mu_{f_2}(\underset{\sim}{x}), \cdots, \mu_{f_p}(\underset{\sim}{x})] \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned} \tag{2.17}$$

or equivalently

$$\begin{aligned} \max \quad & \alpha \\ \text{subject to} \quad & \alpha \leq \mu_{f_i}(\underset{\sim}{x}) \quad i = 1 \ldots p \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned} \tag{2.18}$$

To summarize, the fuzzy programming technique is

1. Calculate $\underset{\sim}{z}^*$ and $\underset{\sim}{z}^0$.

2. Elicit a membership function $\mu_{f_i}(\underset{\sim}{x})$ for each objective-function from the DM.

3. Solve the (possibly) nonlinear programming problem

$$\begin{aligned} \max \quad & \alpha \\ \text{subject to} \quad & \alpha \leq \mu_{f_i}(\underset{\sim}{x}) \quad i = 1 \ldots p \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

Sakawa *et al.* [45] propose a method to solve problem (2.18), by combined use of linear programming and the bisection method.

Chapter 3

Further Investigation of Selected Algorithms

It was decided to select a number of algorithms for further investigation, specifically a computational comparison. Originally it was decided to only consider three algorithms, viz. the Interactive Weighted Tchebbycheff algorithm, the Visual Interactive method, and the method of Zionts and Wallenius.

These three in particular were selected because they seemed to be (in the author's opinion) among the more promising methods. In addition, the three algorithms approach the problem in very different ways and have completely different methods of user interaction.

However having programmed the Zionts and Wallenius algorithm, it became apparent that it would be possible to program the Cutting Plane method with a minimum amount of effort. The advantage of this would be that the scope of the study would be extended to cover another method and that a fourth class of algorithms would be represented. In addition, the author has an obvious interest in comparing the performance of the algorithm to other algorithms.

Similarly, having programmed the other methods, it was simple enough to also implement Steuer's Contracting Cones method, the Interactive Weighted

Sums method and the Reference Point method. There would be a number of advantages in doing this.

Firstly, the study would again be extended and another of Shin and Ravindran's [49] classes of methods would be considered.

Secondly, no method in the study would have a unique method of interaction with the DM. This was considered a desirable feature as it would reduce the chance of users evaluating the algorithms purely on the basis of user interface.

A final reason for including Steuer's Contracting Cones and the Interactive Weighted Sums/Filtering approach is that these two methods are known to produce worse best compromise solutions than the other algorithms. Having these two algorithms included in the study could serve as a type of experimental control.

The algorithms which are therefore selected for further investigation are :

1. Steuer's Contracting Cones method
2. The Interactive Weighted Sums/Filtering method
3. The Interactive Weighted Tchebbycheff algorithm
4. The method of Zionts and Wallenius
5. The Cutting Plane algorithm
6. The Reference Point method
7. The Visual Interactive method

These algorithms are now presented more fully. Since a brief intuitive motivation for each algorithm was given in chapter 2, and the algorithms will be broken down into their constituent steps in chapter 5, not much explanation is given here.

The algorithms are not presented in the categories into which they were classified by Shin and Ravindran [49]. Instead, they are presented in three different categories, viz. sample algorithms, tradeoff algorithms and projection algorithms.

3.1 Sample Algorithms

In this section we consider Steuer's Contracting Cones Algorithm, the Interactive Weighted Sums/Filtering Algorithm and the Interactive Weighted Tchebycheff Algorithm. These algorithms are considered sample algorithms because in each case a widely dispersed group of weighting vectors λ is formed, and these weighting vectors are used to form a sample of the nondominated set of solutions.

This sample of nondominated solutions is presented to the DM who selects his most preferred solution, say (x^k, z^k). We then generate another group of weighting vectors which sample the nondominated set in a neighborhood around (x^k, z^k). As the algorithm proceeds, the weighting vectors formed sample a smaller and smaller neighbourhood of the nondominated set, until the interactive procedure "focuses in" on the DM's final solution.

The three methods are all quite similar (not surprising since they were all developed, entirely or in part, by Steuer) and are basically modifications to the vector maximum algorithm (cf. section 1.3.4).

3.1.1 Steuer's Method of Contracting Cones

The motivation for this algorithm was given briefly in section 2.4. Steuer [53] generates the dispersed set of weights using the following formula.

$$
\begin{aligned}
\underset{\sim}{\lambda}^{1} &= (1, 0, 0, \ldots, 0) \\
\underset{\sim}{\lambda}^{2} &= (0, 1, 0, \ldots, 0) \\
&\vdots \\
\underset{\sim}{\lambda}^{p} &= (0, 0, \ldots, 0, 1) \\
\underset{\sim}{\lambda}^{p+1} &= \left(\tfrac{1}{p^2}, \tfrac{p+1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2}\right) \\
\underset{\sim}{\lambda}^{p+2} &= \left(\tfrac{p+1}{p^2}, \tfrac{1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2}\right) \\
&\vdots \\
\underset{\sim}{\lambda}^{2p} &= \left(\tfrac{p+1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2}, \tfrac{1}{p^2}\right) \\
\underset{\sim}{\lambda}^{2p+1} &= \left(\tfrac{1}{p}, \tfrac{1}{p}, \tfrac{1}{p}, \ldots, \tfrac{1}{p}\right)
\end{aligned}
$$

Refer to section 4.5.1 for the motivation behind this particular choice.

An additional feature not discussed in section 2.4 is that the DM is given the choice as to how many criterion vectors he wishes to compare at a time, P say. This means that the $2p + 1$ nondominated solutions are first filtered to obtain the P most different which are then presented to the DM.

The complete algorithm is therefore

1. Specify P, the number of solutions the DM is prepared to compare at a time. Set $h = 1$. Let $C^{(1)} = C$.

2. Form the $2p+1$ weighting vectors according to Steuer's formula.

$$\begin{aligned}
\underset{\sim}{\lambda}^1 &= (1,0,0,\ldots,0)\\
\underset{\sim}{\lambda}^2 &= (0,1,0,\ldots,0)\\
&\vdots\\
\underset{\sim}{\lambda}^p &= (0,0,\ldots,0,1)\\
\underset{\sim}{\lambda}^{p+1} &= (\tfrac{1}{p^2}, \tfrac{p+1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2})\\
\underset{\sim}{\lambda}^{p+2} &= (\tfrac{p+1}{p^2}, \tfrac{1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2})\\
&\vdots\\
\underset{\sim}{\lambda}^{2p} &= (\tfrac{p+1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2}, \tfrac{1}{p^2})\\
\underset{\sim}{\lambda}^{2p+1} &= (\tfrac{1}{p}, \tfrac{1}{p}, \tfrac{1}{p}, \ldots, \tfrac{1}{p})
\end{aligned}$$

3. Solve the $2p+1$ LP problems

$$\begin{aligned}
&\max \sum_{i=1}^{p} \lambda_i^k f_i\\
&\text{s.t. } A\underset{\sim}{x} \leq \underset{\sim}{b}\\
&\underset{\sim}{x} \geq \underset{\sim}{0}
\end{aligned}$$

for $k = 1 \ldots 2p+1$. Call the solutions $\underset{\sim}{x}^k$, $k = 1 \ldots 2p+1$. Let $\underset{\sim}{z}^k = C\underset{\sim}{x}^k$. If all $\underset{\sim}{x}^k$ are identical, stop, $\underset{\sim}{x}^k$ is the optimal solution.

4. Filter the $2p+1$ $\underset{\sim}{z}^k$-vectors to obtain the P most different ones.

5. Present these P criterion vectors to the DM and denote the most preferred one $\underset{\sim}{z}^{(h)}$.

6. Contract the reduced criterion cone $C^{(h)}$ about the convex combination coefficients that produced $\underset{\sim}{z}^{(h)}$ to form $C^{(h+1)}$.

7. Go to 2.

3.1.2 Interactive Weighted Sums/Filtering Method

As described in section 2.4.2, this approach differs from the previous one in two ways; instead of reducing the criterion cone, the set of weights which are generated are constrained to fall within a certain interval. This is made possible by altering the way in which the weights are generated. Instead of generating

these by formula, an overabundance of weights are generated randomly and then filtered to obtain the required amount.

Thus the complete Interactive Weighted Sums/Filtering algorithm is

1. Specify P, the number of solutions the DM is prepared to compare at a time, and r, the Λ-reduction factor. Set $h = 0$. Let $[l_i^{(h)}, r_i^{(h)}] = [0,1]$ for all i.

2. Let $h = h + 1$. Form $\Lambda^{(h)} = \{\lambda \in \mathbb{R}^p \mid \lambda_i \in [l_i^{(h)}, r_i^{(h)}], \sum_{i=1}^{p} \lambda_i = 1\}$.

3. Randomly generate $50p$ λ-vectors from $\Lambda^{(h)}$.

4. Filter these to obtain the $3P$ most different ones. Call these filtered vectors, $\lambda^1, \lambda^2, \ldots, \lambda^{3P}$.

5. Solve the $3P$ LP problems

$$\begin{aligned} \max & \sum_{i=1}^{p} \lambda_i^k f_i \\ \text{s.t.} & \; Ax \leq b \\ & \; x \geq 0 \end{aligned}$$

for $k = 1 \ldots 3P$. Call the solutions $x^k, k = 1 \ldots 3P$. Let $z^k = Cx^k$. If all x^k are identical, stop, x^k is the optimal solution.

6. Filter the $3P$ z^k-vectors to obtain the P most different ones.

7. Present these P criterion vectors to the DM and denote the most preferred one $z^{(h)}$.

8. Let $\lambda^{(h)}$ be the λ-vector which generated $z^{(h)}$.

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

9. Go to 2.

3.1.3 Interactive Weighted Tchebbycheff Method

As described in section 2.4.2, the Interactive Weighted Tchebbycheff Method generates a sample of the nondominated set by minimising a scalarizing function instead of maximizing a weighted sum of the objectives.

The scalarizing function used is the weighted Tchebbycheff metric, i.e.

$$s(\underline{f}(\underline{x}) - \underline{z}^*) = \max_{i=1\ldots p} \lambda_i \mid f_i(\underline{x}) - z_i^* \mid$$

In addition, the weights which provide a definition point for the Tchebbycheff norm are used. (See section 5.1.3).

This results in the following algorithm :

1. Specify P, the number of solutions the DM is prepared to compare at a time, and r, the Λ-reduction factor. Obtain $\underline{z}^*$, the ideal criterion vector. Set $h = 0$. Let $[l_i^{(h)}, r_i^{(h)}] = [0, 1]$ for all i.

2. Let $h = h + 1$. Form $\Lambda^{(h)} = \{\lambda \in \Re^p \mid \lambda_i \in [l_i^{(h)}, r_i^{(h)}], \sum_{i=1}^{p} \lambda_i = 1\}$.

3. Randomly generate $50p$ λ-vectors from $\Lambda^{(h)}$.

4. Filter these to obtain the $2P$ most different. Call these filtered vectors, $\underline{\lambda}^1, \underline{\lambda}^2, \ldots, \underline{\lambda}^{2P}$.

5. Solve the $2P$ Tchebbycheff Programming problems

$$\begin{aligned} & \min \{\alpha + \varepsilon \sum_{i=1}^{p} (z_i^* - z_i)\} \\ \text{s.t. } & \alpha \geq \lambda_i^k (z_i^* - z_i) \quad i = 1 \ldots p \\ & f_i(\underline{x}) = z_i \quad i = 1 \ldots p \\ & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

for $k = 1 \ldots 2P$. Call the solutions $\underline{x}^k, k = 1 \ldots 2P$. Let $\underline{z}^k = C\underline{x}^k$. If all $\underline{x}^k$ are identical, stop, $\underline{x}^k$ is the optimal solution.

6. Filter the $2P$ z^k-vectors to obtain the P most different ones.

7. Present these P criterion vectors to the DM and denote the most preferred one $z^{(k)}$.

8. Let $\lambda_i^{(k)} = \begin{cases} \frac{1}{z_i^* - z_i^{(k)}} \left[\sum_{i=1}^{p} \frac{1}{z_i^* - z_i^{(k)}}\right]^{-1} & \text{if } z_i^{(k)} \neq z_i^* \ \forall i = 1 \ldots p \\ 1 & \text{if } z_i^{(k)} = z_i^* \\ 0 & \text{if } z_i^{(k)} \neq z_i^* \text{ but } \exists j \text{ s.t. } z_j^{(k)} = z_j^* \end{cases}$

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

9. Go to 2.

3.1.4 Comments

1. Steuer's Contracting Cones and the Method of Interactive Weighted Sums only search among extreme points to locate the most efficient extreme point. The Tchebbycheff method, on the other hand, can converge to non-extreme solutions.

2. Steuer [56] pp. 399 claims that because of the way in which the weighting space is reduced in the Interactive Weighted Sums and Tchebbycheff methods, "the algorithm has an error correcting facility provided the error is not too severe".

3.2 Tradeoff Algorithms

In this section we consider the method of Zionts and Wallenius and the Cutting Plane Algorithm. These algorithms are called tradeoff algorithms because they use the tradeoff information found in the simplex tableau to obtain information from the DM.

The algorithms follow the following basic scheme

1. Solve the problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i f_i, \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

   for some weights $\underset{\sim}{\lambda}$. Denote the solution $\underset{\sim}{x}^{(h)}$.

2. Ask the DM if he likes the tradeoffs leading from $\underset{\sim}{x}^{(h)}$.

3. Obtain a new set of weights $\underset{\sim}{\lambda}$ consistent with the DM's responses.

4. Reduce the solution space in some way.

5. Go to 1.

3.2.1 The Method of Zionts and Wallenius

The method of Zionts and Wallenius in which the utility function is assumed to be pseudoconcave is similar to the one presented in section 2.5, which assumes a linear utility function. The following additions are made to handle the pseudoconcavity of U.

Since the utility function is assumed to be pseudoconcave, the constraints on the weight space created on the basis of the DM's responses may be valid when they are constructed but invalid later. These constraints are then dropped.

This means that the Zionts and Wallenius scheme must also consider efficient edges which are not efficient with respect to the set of previous responses of the DM, because some of these responses may be dropped.

In addition, since U is pseudoconcave, the fact that the DM prefers $\underset{\sim}{x}^A$ to $\underset{\sim}{x}^B$ does not necessarily mean that he does not like the tradeoffs leading from $\underset{\sim}{x}^A$ to $\underset{\sim}{x}^B$. However since DMs prefer comparing alternatives to evaluating tradeoffs (Zionts and Wallenius [71]), the DM is first asked to compare alternatives. Should he not prefer an adjacent extreme point to the current solution, the tradeoff between the two is presented to the DM for evaluation.

Having generated a new iterate, the DM is asked to compare the new iterate to the current solution and also to solutions he preferred to the current solution. Constraints based on these responses are added and the procedure is restarted.

Thus the Zionts and Wallenius algorithm in which the utility function is assumed to be pseudoconcave is :

1. Choose a starting set of weights $\underset{\sim}{\lambda}^{(0)}$. Set $h = 0$. ($\underset{\sim}{\lambda}^{(0)}$ is normally selected to be $\lambda_i^{(0)} = \frac{1}{p}$, for $i = 1 \ldots p$)

2. Solve the linear programming problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i^{(h)} f_i(\underset{\sim}{x}) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

   Call the solution to this problem $\underset{\sim}{x}^{(h)}$.

3. Identify all efficient edges from $\underset{\sim}{x}^{(h)}$ and the corresponding adjacent efficient extreme point solutions.

4. Determine which of the efficient edges are also efficient with respect to the set of previous responses of the DM. Call the set of such edges A. Call its complement with respect to the set of efficient edges B. Let $I = A$.

5. With respect to set I, ask the DM to choose between $\underset{\sim}{x}^{(h)}$ and a distinctly different adjacent extreme point solution. He may express a preference for one of the two points, or the inability to express a preference. If he prefers at least one alternative to $\underset{\sim}{x}^{(h)}$, save one such alternative, denote it as $\underset{\sim}{x}^{a}$, and go to step 9.

6. With respect to set I, ask the DM whether, beginning at point $\underset{\sim}{x}^{(h)}$, he likes any efficient tradeoff not leading to a point asked about in step 5. He may respond that he likes the tradeoff, that he does not like the tradeoff, or that he is unable to decide. If he likes at least one such tradeoff, then go to step 9.

7. With respect to set I, ask the DM whether, beginning at point $\underset{\sim}{x}^{(h)}$, he likes any tradeoff leading to adjacent efficient points which he did not prefer to $\underset{\sim}{x}^{(h)}$ in the most recent step 5. If he likes at least one such tradeoff, then go to step 9.

8. If I is equal to A, let I equal B and go to step 5. Otherwise stop, solution $\underset{\sim}{x}^{(h)}$ is globally optimal.

9. Write constraints on the weights based on the DM's responses and add them to the set.
i.e. if he likes the tradeoff t_{ij}, then construct a constraint of the form

$$\sum_{i=1}^{p} t_{ij}\lambda_i \leq -\varepsilon$$

if he does not like the tradeoff t_{ij}, then construct a constraint of the form

$$\sum_{i=1}^{p} t_{ij}\lambda_i \geq \varepsilon$$

if he is indifferent to the tradeoff t_{ij}, then construct a constraint of the form

$$\sum_{i=1}^{p} t_{ij}\lambda_i = 0$$

The equality constraint is normally left-out since it is unlikely that the DM will have a sufficiently accurate estimation of tradeoffs to be able to declare indifference.

10. Find a set of positive weights $\underset{\sim}{\lambda}^{(h+1)}$ consistent with all previous responses. If no such set of consistent weights exists, drop the oldest active constraint and repeat step 10.

11. Using the new weighting vector $\underline{\lambda}^{(h+1)}$, solve the linear programming problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i^{(h+1)} f_i(\underline{x}) \\ \text{subject to} \quad & A\underline{x} \leq \underline{b} \\ & \underline{x} \geq \underline{0} \end{aligned}$$

Call this solution $\underline{x}^b$.

12. If solution $\underline{x}^b$ is not null, go to step 14. Otherwise have the DM choose between solutions $\underline{x}^a$ and $\underline{x}^{(h)}$. If the DM chooses $\underline{x}^a$, add a constraint based on the choice of $\underline{x}^a$ over $\underline{x}^{(h)}$, designate the solution $\underline{x}^a$ as $\underline{x}^{(h)}$ and go to step 3.

13. Solution $\underline{x}^{(h)}$ is a local optimum, but there exist better solutions, some of which are not extreme points. A search procedure (distinct from the method) should be used to find that optimum. The method terminates at local optimum $\underline{x}^{(h)}$.

14. Ask the DM to choose between $\underline{x}^a$ and $\underline{x}^b$. If the DM chooses $\underline{x}^a$, add a constraint based on the choice of $\underline{x}^a$ over $\underline{x}^b$, denote the preferred solution as $\underline{x}^{(h)}$, make $\underline{x}^b$ null and go to step 3. If he is unable to choose, denote the $\underline{x}^a$ as $\underline{x}^{(h)}$, make $\underline{x}^b$ null and go to step 3.

3.2.2 The Cutting Plane Method

As described in section 2.6.2, the Cutting Plane method combines aspects of the method of Zionts and Wallenius and the cutting plane method proposed by Loganathan and Sherali [33].

The algorithm is similar to the method of Zionts and Wallenius with three major exceptions :-

1. Having set up constraints on $\underline{\lambda}$, the Zionts and Wallenius technique selects *any* weights which satisfy the constraints, while the Cutting Plane method selects weights which are in the centre of the feasible weight space.

2. Once this set of weights has been generated, all the constraints based on the DM's preferences are dropped in the Cutting Plane method. Instead a cutting plane is introduced to reduce the variable space.

3. Iterates are generated by using Loganathan and Sherali's technique of searching in the direction of maximum increase, as opposed to the weighted sums approach of the method of Zionts and Wallenius.

The algorithm is therefore

1. Choose a starting set of weights $\underset{\sim}{\lambda}^{(0)}$. Set $h = 0$. ($\underset{\sim}{\lambda}^{(0)}$ is normally selected to be $\lambda_i^{(0)} = \frac{1}{p}$, for $i = 1 \ldots p$)

2. Solve the linear programming problem

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} \lambda_i^{(h)} f_i(\underset{\sim}{x}) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

Call the solution to this problem $\underset{\sim}{x}^{(h)}$.

3. Identify all efficient edges from $\underset{\sim}{x}^{(h)}$ and the corresponding adjacent efficient extreme point solutions. Let A be the set of indices of efficient tradeoffs, i.e. $i \in A$ iff $\underset{\sim}{t}^i$ is an efficient tradeoff. Let $\underset{\sim}{x}^i$ be the extreme point corresponding to tradeoff $\underset{\sim}{t}^i$. Let q be the first element in A. Let $B = C = \emptyset$.

4. If point $\underset{\sim}{x}^{(h)}$ and $\underset{\sim}{x}^q$ are 'too close' go to step 6. What is meant by too close is that the solutions are such that the DM has difficulty in making a choice between them. Zionts and Wallenius [71] suggest that $\underset{\sim}{x}^{(h)}$ and $\underset{\sim}{x}^q$ can be considered sufficiently different if there is at least ten percent difference in one objective function value.

5. Ask the DM if he prefers $\underset{\sim}{x}^q$ to $\underset{\sim}{x}^{(h)}$. If he does then add q to set B and go to step 7. Otherwise go to step 6.

6. Ask the DM whether, beginning at point $\underset{\sim}{x}^{(h)}$, he likes the efficient tradeoff leading to $\underset{\sim}{x}^q$. If he does, then add q to the set B, otherwise add q

to the set C. In practice, instead of asking the DM whether he likes the tradeoff leading from $\underset{\sim}{x}^{(h)}$ to $\underset{\sim}{x}^{q}$ we ask him if he prefers the point $\underset{\sim}{x}^{q''}$ to $\underset{\sim}{x}^{(h)}$, where $\underset{\sim}{x}^{q''} = \underset{\sim}{x}^{(h)} + 0.1(\underset{\sim}{x}^{q} - \underset{\sim}{x}^{(h)})$. This is done so that the DM only has to do pairwise comparisons and not tradeoff evaluation.

7. If q is the last element of set A then go to step 8, otherwise let q be the next element of set A and go to step 4.

8. If $B = \emptyset$, then stop. $\underset{\sim}{x}^{(h)}$ is optimal.

9. Solve the linear programming problem

$$\begin{aligned} \max \quad & \alpha \\ \text{subject to} \quad & \sum_{i=1}^{p} t_i^j \lambda_i + s_j = 0 \quad \forall j \in B \\ & \sum_{i=1}^{p} t_i^j \lambda_i - s_j = 0 \quad \forall j \in C \\ & s_j \geq \alpha \quad \forall j \in B \cup C \\ & \sum_{i=1}^{p} \lambda_i = 1 \\ & \lambda_i \geq \alpha \quad \forall i = 1 \ldots p \\ & \alpha \geq 0 \end{aligned}$$

Call the solution $(\underset{\sim}{\lambda}^{(h+1)}, \underset{\sim}{s}^{(h+1)})$.

10. Solve the linear programming problem

$$\begin{aligned} \max \quad & \alpha \\ \text{subject to} \quad & \sum_{i=1}^{p} (f_i(\underset{\sim}{x}) - f_i(\underset{\sim}{x}^{(j)}))\lambda_i^{(j)} \geq \alpha \quad \forall j = 1 \ldots h \\ & \underset{\sim}{A}\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \\ & \alpha \geq 0 \end{aligned}$$

Call the solution $(\underset{\sim}{x}^{(h+1)}, \alpha^{(h+1)})$.

11. If $\alpha^{(h+1)} = 0$ then stop, $\underset{\sim}{x}^{(h+1)}$ is optimal. Otherwise set $k = k + 1$ and go to step 3.

3.2.3 Comments

1. Constraints added in the method of Zionts and Wallenius are not always valid and may have to be dropped later (Step 9). The constraints added in the Cutting Plane are valid throughout.

2. The fact that some constraints are dropped in the method of Zionts and Wallenius means that the algorithm has an error correcting capability.

3. The method of Zionts and Wallenius cannot converge to nonextrema solutions whereas the Cutting Plane method can.

4. The Cutting Plane method only ever asks the DM to choose between two points. He may be expected to express an opinion on the desirability of a tradeoff in the Method of Zionts and Wallenius.

3.3 Projection Algorithms

In this section we consider the Reference Point Method and the Visual Interactive Method of Korhonen and Laakso. These algorithms are termed projection algorithms because they project some (possibly infeasible) criterion vector point(s) onto the set of nondominated criterion vectors. The projection is done by means of a scalarizing function. The Reference Point method projects a single refrence point onto the nondominated set, while the Visual Interactive Method projects an entire unbounded line segment onto the nondominated set.

3.3.1 The Reference Point Method

The motivation for the Reference Point algorithm was given in section 2.2. The particular scalarizing function used here is the weighted Tchebbycheff norm. i.e.

$$s(f(x) - z^*) = \max_{i=1\ldots p} \lambda_i \mid f(x) - z^* \mid$$

where λ is selected such that the reference point y is a definition point of the Tchebbycheff norm. This is explained in section 5.1.3.

This results in the following algorithm

1. Obtain z^*, the ideal criterion vector.
2. Ask the DM to express his aspirations as $y^{(k)}$ where $y_i^{(k)} < z_i^*$ for all i.
3. Let $\lambda_i^{(k)} = \begin{cases} \dfrac{1}{z_i^* - y_i^{(k)}} \left[\sum_{i=1}^{p} \dfrac{1}{z_i^* - y_i^{(k)}}\right]^{-1} & \text{if } y_i^{(k)} \neq z_i \ \forall i = 1 \ldots p \\ 1 & \text{if } y_i^{(k)} = z_i^* \\ 0 & \text{if } y_i^{(k)} \neq z_i^* \text{ but } \exists j \text{ s.t. } y_j^{(k)} = z_j^* \end{cases}$
4. Solve the problem

$$\min \{\alpha + \varepsilon \sum_{i=1}^{p} (y_i^{(h)} - z_i)\}$$
$$\text{s.t. } \alpha \geq \lambda_i^{(h)} (y_i^{(h)} - z_i) \quad \forall i = 1 \ldots p$$
$$f_i(\underline{x}) = z_i \quad \forall i = 1 \ldots p$$
$$A\underline{x} \leq \underline{b}$$
$$\underline{x} \geq \underline{0}$$

in order to obtain $\underline{z}^{(h)}$ which is a projection of $\underline{y}^{(h)}$ onto the nondominated set.

5. If the DM is satisfied with $\underline{z}^{(h)}$ then stop, else go to 2.

3.3.3 The Visual Interactive Method of Korhonen and Laakso

This algorithm was described briefly in section 2.3. Once again, the scalarizing function used is

$$s(\underline{f}(\underline{x}) - \underline{z}^*)) = \max_{i=1 \ldots p} \lambda_i | \underline{f}(\underline{x}) - \underline{z}^* |$$

Thus the Visual Interactive method is

1. Select an achievement scalarizing function $s(\underline{q}, \underline{z}, \underline{\lambda})$ and an initial point $\underline{z}^0$ in criterion space (not necessarily feasible). An achievement scalarizing function is one which projects a reference point q onto the set of nondominated criterion vectors N. Set $h = 0$.

2. Set $h = h + 1$. Select a reference point $\underline{q}^{(h)}$. Set $\underline{d}^{(h)} = \underline{q}^{(h)} - \underline{z}^{(h-1)}$. Obtain weights $\underline{\lambda}$ from the DM.

3. Solve the achievement scalarizing program to project $\underline{z}^{(h-1)}$ onto N. If we use the achievement scalarizing function

$$s(\underline{q}, \underline{z}, \underline{\lambda}) = \max_i \{\lambda_i (q_i - z_i)\} - \varepsilon \sum_{i=1}^{p} z_i$$

then to project the unbounded line segment that emanates from $\underline{z}^{(h-1)}$ in the direction $\underline{d}$, we solve the achievement scalarizing parametric programming problem

$$\min \{\alpha - \varepsilon \sum_{i=1}^{p} z_i\}$$
$$\text{s.t. } z_i + \alpha \leq z_i^{(h-1)} + \beta d_i \quad (h)$$
$$f_i(\underline{x}) = z_i$$
$$A\underline{x} \leq \underline{b}$$
$$\underline{x} \geq \underline{0}$$

4. Display the criterion value trajectories along the projection on N. Designate the DM's most preferred criterion vector $\underline{z}^{(h)}$.

5. If the DM is satisfied with $\underline{z}^{(h)}$ then stop, else go to step 2.

Korhonen and Laakso [28] replace step 5 by the following steps

5. If $\underline{z}^{(h)} \neq \underline{z}^{(h-1)}$ then go to step 2.

6. Obtain a set of generators for the cone of feasible criterion vector directions at $\underline{z}^{(h)}$.

7. If there are any generators that have not been tested as a direction for improvement go to step 8, else go to step 9.

8. Let $h = h + 1$. Select an untested generator and designate it $\underline{g}^{(h)}$. Go to step 3.

9. Let $\underline{x}^{(h)}$ be the inverse image of $\underline{z}^{(h)}$. Terminate with $\underline{x}^{(h)}, \underline{z}^{(h)}$ as final solution.

Steuer [56] reports that Korhonen and Wallenius have been working on a method to yield the generators of the cone of feasible directions at $\underline{z}^{(h)}$. The hypothesis is that the achievement scalarizing program with $\underline{\lambda}$-weights such that $\underline{z}^{(h)}$ is a solution will provide the generators. In the meanwhile, there appears to be no simple way in which to obtain the generators.

3.3.3 Comments

1. Both methods can converge to nonextreme optimal solutions.

2. The Reference Point Method requires the solving of linear programming problems only, while the Visual Interactive Method requires the solving of parametric programming problems.

Chapter 4

Implementation Issues

All the algorithms under discussion include steps (or subproblems) which are not unique to the algorithm. These subproblems tend to be standard problems and the method of solution of these subproblems is not inherently part of the algorithm. Examples are the scaling of objective functions and the solving of linear programming problems.

However, in order to implement the algorithms in practice, a method of solution for each subproblem must be selected and implemented. The various subproblems which occur in the algorithms under scrutiny in this study are now discussed, together with possible methods of solution.

The particular subproblems discussed are

1. Solving LP problems
2. Testing whether a tradeoff leads to an efficient solution or not
3. Solving parametric programming problems
4. Scaling the objective functions
5. Generating a set of weights λ from Λ

6. Filtering

7. Contracting the criterion cone

Definition 4.1 A subproblem is termed an *independent subproblem* if the solution to the subproblem is independent of the solution method.

For example, the subproblem of testing whether a tradeoff leads to an efficient solution or not is an independent subproblem. If the tradeoff leads to an efficient solution then every method for testing this will tell us that it leads to an efficient solution.

Definition 4.2 A subproblem is termed a *dependent subproblem* if the solution to the subproblem is dependent on the method of solving the subproblem.

For example, the subproblem of scaling the objective functions is a dependent subproblem since each method used results in a different set of scaled objective functions.

Clearly the first three subproblems are independent subproblems, and the rest are dependent subproblems.

The reason we make this distinction is that the method of solving independent subproblems does not affect the performance of the algorithms, and hence the method of solving the subproblem is relatively unimportant. Dependent subproblems may affect the performance of the algorithm and hence solution methods are important, and are therefore discussed in more detail.

We first present the independent subproblems.

4.1 Linear Programming

The Cutting Plane algorithm and the method of Zionts and Wallenius, are both completely dependent on the Simplex algorithm, since they require the $z_j - c_j$ values from the Simplex tableau. The other algorithms are fairly independent of the linear programming method. A detailed discussion into the Simplex algorithm is not warranted here. Refer to Murty [39], Zionts [68] or one of the other numerous books which cover the subject of linear programming in detail.

4.2 Testing for Tradeoff Efficiency

Four methods are discussed, but since this is an independent subproblem, no justification is given. A more detailed discussion of the methods can be found in Steuer [56] pp. 233-242.

4.2.1 The Evans-Steuer Test

Solve

$$\begin{array}{lll} \max & \underset{\sim}{1}^T \underset{\sim}{y} & \\ \text{subject to} & \underset{\sim}{w}^j \delta - W\underset{\sim}{y} + I\underset{\sim}{v} = \underset{\sim}{0} & \\ & \underset{\sim}{0} \leq \underset{\sim}{y} \in \Re^{n-m} & \\ & 0 \leq \delta \in \Re^1 & \\ & \underset{\sim}{0} \leq \underset{\sim}{v} \in \Re^p & \end{array}$$

Where W is the $c_j - z_j$ reduced cost matrix of an efficient basis and $\underset{\sim}{w}^j$ is the jth column of W. If the optimal objective function value is zero, the jth nonbasic variable is efficient. If the optimal objective function value is positive-unbounded, the jth nonbasic variable is inefficient.

4.2.2 Isermann's Test

Solve

$$\begin{array}{lll} \max & \underset{\sim}{1}^T \underset{\sim}{v} & \\ \text{subject to} & W^N \underset{\sim}{\delta} - W\underset{\sim}{y} + I\underset{\sim}{v} = \underset{\sim}{1} & \\ & \underset{\sim}{0} \leq \underset{\sim}{y} \in \Re^{n-m} & \\ & \underset{\sim}{0} \leq \underset{\sim}{\delta} \in \Re^{|N|} & \\ & \underset{\sim}{0} \leq \underset{\sim}{v} \in \Re^p & \end{array}$$

where W^N is the matrix of columns of W corresponding to the nonbasic indices in an index set N and $|N|$ is the number of elements in N.

In Isermann's test, each nonbasic variable x_j is efficient iff the above problem

has an optimal solution.

Isermann's test is very similar to the Evans-Steuer test, but Steuer [56] claims that it offers opportunities for being more economical when we wish to determine all nonbasic variables which lead to efficient solutions.

4.2.3 Ecker's Test

Consider the set of constraints

$$\begin{aligned} W^T(\underline{\lambda} + \underline{\Pi}) + \underline{\gamma} &= \underline{0} \\ \underline{0} \leq \underline{\lambda} &\in \Re^n \\ \underline{0} \leq \underline{\gamma} &\in \Re^{n-m} \end{aligned}$$

where W is the $c_j - z_j$ reduced cost matrix of an efficient basis. Then x_j is an efficient nonbasic variable iff there exists a solution $(\overline{\lambda}, \overline{\gamma}) \in \Re^{p+n-m}$ such that $\overline{\gamma}_j = 0$.

4.2.4 The Zionts and Wallenius Test

For each $k \in N$ solve

$$\begin{aligned} \max \quad & \sum_{i=1}^{p} w_{ik} y_i \\ \text{subject to} \quad & \sum_{i=1}^{p} w_{ij} y_i \leq 0 \quad \forall j \in N,\ j \neq k \\ & \text{all } y_i \geq 0 \end{aligned}$$

where N is the set of nonbasic variables.

If the maximum objective function value of the problem is positive, then x_k is efficient. If the maximum objective function value of the problem is nonpositive, then x_k is not efficient.

4.3 Parametric Programming

Suppose we have found an optimal solution to the LP problem

$$\begin{aligned} \text{min} \quad & z = \underset{\sim}{c}^T \underset{\sim}{x} \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

and then decide that we wish to modify the requirement vector $\underset{\sim}{b}$, i.e. we want to solve the problem

$$\begin{aligned} \text{min} \quad & z = \underset{\sim}{c}^T \underset{\sim}{x} \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \hat{\underset{\sim}{b}} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

where $\hat{\underset{\sim}{b}} = \underset{\sim}{b} + \underset{\sim}{f}$.

We need to recompute the values of the basic variables corresponding to the vectors in the current basic feasible solution. If we denote the current basis matrix by B then

$$\begin{aligned} \hat{\underset{\sim}{x}}_B &= B^{-1}\hat{\underset{\sim}{b}} \\ &= B^{-1}(\underset{\sim}{b} + \underset{\sim}{f}) \\ &= B^{-1}\underset{\sim}{b} + B^{-1}\underset{\sim}{f} \\ &= \underset{\sim}{x}_B + B^{-1}\underset{\sim}{f} \end{aligned}$$

Note that the $z_j - c_j$'s are unaffected by a change in the right-hand-side vector and therefore if this $\hat{\underset{\sim}{x}}_B$ is feasible, it is also optimal. If $\hat{\underset{\sim}{x}}_B$ is infeasible, (i.e. one or more $\hat{x}_{B_i} < 0$) we must use the dual simplex algorithm to find the new optimal solution.

The problem which we need to solve using parametric programming is

$$\begin{aligned} \text{min} \quad & z = \underset{\sim}{c}^T \underset{\sim}{x} \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} + \underset{\sim}{f}\theta \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

for $\theta = 0 \rightarrow \infty$.

The way in which we achieve this is as follows

1.
$$\begin{aligned} \min \quad & z = \underset{\sim}{c}^T \underset{\sim}{x} \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

2. Let $\theta^c = 0$

3. We wish to find the value of θ at which $\hat{\underset{\sim}{b}}_B = 0$.
Now $\hat{\underset{\sim}{b}}_B = 0 \Rightarrow \underset{\sim}{x}_B = -B^{-1}\underset{\sim}{f}\theta$.
Let $\theta_i = \dfrac{\underset{\sim}{x}_{B_i}}{(-B^{-1}\underset{\sim}{f})_i}$, i.e. θ_i is the value of θ at which the ith constraint becomes infeasible.
If all $\theta_i < 0$ then go to 8, else let $\theta^* = \min_i\{\theta_i \mid \theta_i > 0\}$.
Hence $\hat{\underset{\sim}{x}}_B = \underset{\sim}{x}_B + B^{-1}\underset{\sim}{f}\theta^*$ for $0 \leq \theta \leq \theta^*$.

4. Let $\theta^c = \theta^*$

5. Now solve
$$\begin{aligned} \min \quad & z = \underset{\sim}{c}^T \underset{\sim}{x} \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} + \underset{\sim}{f}(\theta^c + \varepsilon) \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$
ε small.

6. Again solve for θ^* as in step 3. $\hat{\underset{\sim}{x}}_B = \underset{\sim}{x}_B + B^{-1}\underset{\sim}{f}\theta^*$ for $\theta^c \leq \theta \leq \theta^c + \theta^*$.

7. Go to 5

8. There is no feasible solution for $\theta > \theta^c$.

Step 5 is commonly solved using the dual simplex algorithm.

Note that B^{-1} is immediately available from the final Simplex tableau.

4.4 Scaling the Objective Functions

Consider the two objective functions

$$\begin{aligned} f_1 &= 27312x_1 + 91318x_2 - 31791x_3 \\ f_2 &= 0.83x_1 - 0.932x_2 + 0.103x_3 \end{aligned} \tag{4.1}$$

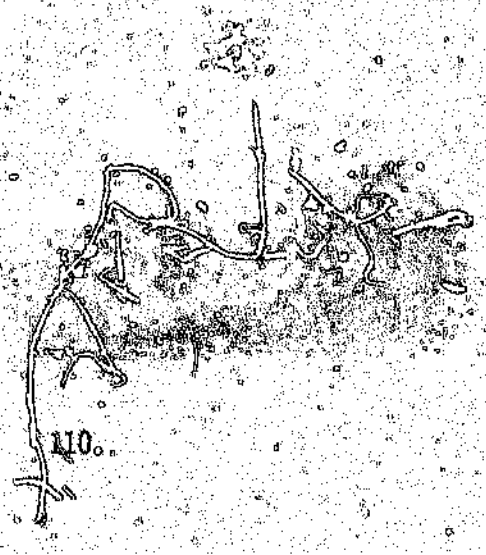
Assume the DM decides that f_2 is four times more important than f_1. It is logical to expect that his weights are therefore $\underline{\lambda} = (0.2, 0.8)$.

The surrogate objective function $\sum_{i=1}^{p} \lambda_i f_i$ is going to be dominated by f_1 simply because of its large coefficients, even though the DM felt that f_2 was far more important. It is therefore necessary to scale the objective functions to equal orders of magnitude.

We discuss three ways of scaling the objective functions.

4.4.1 Normalization

Dividing each component of $\underline{c}_i$ where $f_i = \underline{c}_i^T \underline{x}$ by $\|\underline{c}_i\|$ will normalize $\underline{c}_i$. Doing this to all objective functions will bring all objective functions into the same order of magnitude. One can use any of the L_p norms as defined in definition 1.24.



Example 4.3 Normalizing the objective functions 4.1 given above using the three most common L_ρ norms gives

$$\begin{aligned}
L_1 \text{ norm}: \quad f_1 &= 0.181667x_1 + 0.607406x_2 - 0.210927x_3 \\
f_2 &= 0.44504x_1 - 0.499732x_2 + 0.0552279x_3 \\
L_2 \text{ norm}: \quad f_1 &= 0.271892x_1 + 0.909075x_2 - 0.31568x_3 \\
f_2 &= 0.662806x_1 - 0.74426x_2 + 0.0822519x_3 \\
L_\infty \text{ norm}: \quad f_1 &= 0.299087x_1 + x_2 - 0.347259x_3 \\
f_2 &= 0.890558x_1 - x_2 + 0.110515x_3
\end{aligned}$$

Note that the objective functions have become completely unrecognizable.

4.4.2 Scaling using a Power of 10

To overcome the problem of unrecognizable coefficients, Steuer [56] proposes scaling each objective function by a power of 10. In this way, the recognition of the objective function coefficients is retained because only the decimal place moves.

Example 4.4 Normalizing the objective functions 4.1 using the scaling by a power of 10 gives

$$\begin{aligned}
f_1 &= 0.27312x_1 + 0.91318x_2 - 0.31711x_3 \\
f_2 &= 0.83x_1 - 0.932x_2 + 0.103x_3
\end{aligned}$$

4.4.3 Scaling using Range Equalization

The purpose of the range equalization weights is to equalize the ranges of the objective functions. To compute the range equalization weights, let R_i denote the range of the ith objective function. i.e.

$$R_i = \max\{f_i | A\underline{x} \leq \underline{b}\} - \min\{f_i | A\underline{x} \leq \underline{b}\}$$

Then the range equalization weights are

$$\pi_i = \frac{1}{R_i} \Big/ \sum_{i=1}^{p} \frac{1}{R_i}$$

The objective functions can be scaled by multiplying each objective function coefficient by the corresponding range equalization factor i.e. the scaled objectives are $\pi_i f_i$.

This method is computationally more complex since we need to find the maximum and minimum objective function values subject to the constraint set, but tends to produce better results since it is also dependent on the constraint set.

4.5 Generating Weights from Λ

Some of the methods require us to obtain some weighting vectors from the set of feasible weighting vectors Λ. These weights λ_i are then used to form the surrogate objective function $\sum_{i=1}^{p} \lambda_i f_i$. These weighting vectors should be well dispersed and representative of the set Λ, so that the surrogate objective functions are well dispersed and representative of the criterion cone. We would thus like to discretize the set Λ into a number of points.

4.5.1 Predetermined Convex Combinations

A set can be characterized by an ordered set of its extreme points. For example, the criterion cone shown in Figure 4.1, can be characterized by the set $\{\varepsilon^1, \varepsilon^2, \varepsilon^3, \varepsilon^4, \varepsilon^5\}$.

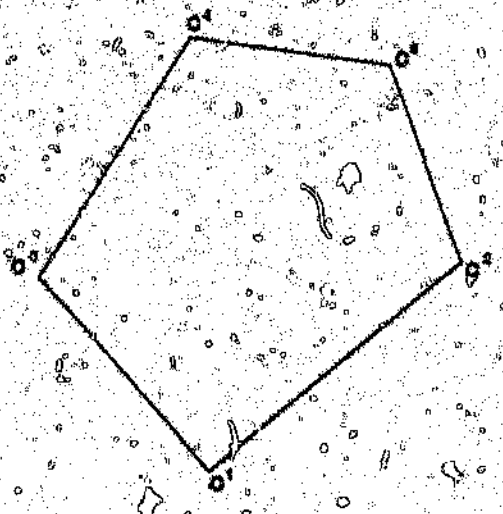


Figure 4.1: A Criterion Cone

In this method, we manufacture weighting vectors according to a predetermined formula. The weights are constructed to be well dispersed. The weight-

ing vectors normally used are those developed by Steuer [53], viz.

$$\begin{aligned}
\underset{\sim}{\lambda}^{1} &= (1, 0, 0, \ldots, 0) \\
\underset{\sim}{\lambda}^{2} &= (0, 1, 0, \ldots, 0) \\
&\vdots \\
\underset{\sim}{\lambda}^{p} &= (0, 0, \ldots, 0, 1) \\
\underset{\sim}{\lambda}^{p+1} &= \left(\tfrac{1}{p^2}, \tfrac{p+1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2}\right) \\
\underset{\sim}{\lambda}^{p+2} &= \left(\tfrac{p+1}{p^2}, \tfrac{1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2}\right) \\
&\vdots \\
\underset{\sim}{\lambda}^{2p} &= \left(\tfrac{p+1}{p^2}, \tfrac{p+1}{p^2}, \ldots, \tfrac{p+1}{p^2}, \tfrac{1}{p^2}\right) \\
\underset{\sim}{\lambda}^{2p+1} &= \left(\tfrac{1}{p}, \tfrac{1}{p}, \tfrac{1}{p}, \ldots, \tfrac{1}{p}\right)
\end{aligned}$$

It could be argued that these weighting vectors are well dispersed because they correspond to the extreme points, the mean point, and the centre points of the areas that have the mean and $2p$ of the $2p + 1$ extreme points as vertices. In Figure 4.2, where these weights have been applied to the cone defined in Figure 4.1, resulting convex combinations are seen to be not very well distributed. This method will always generate $2p + 1$ λ-vectors. If we wished to use this

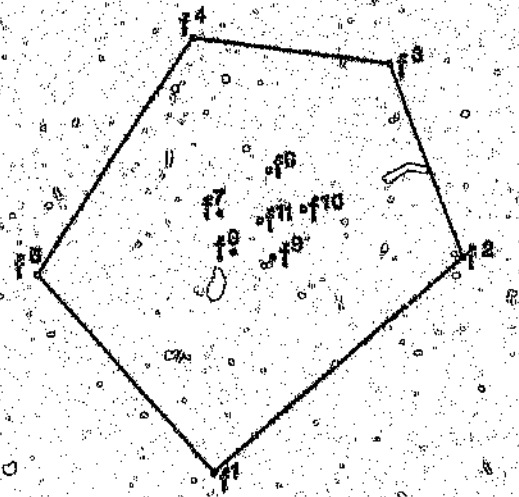


Figure 4.2: Applying Predetermined Convex Combinations

method to find some other number of λ-vectors, it might not have been so easy to manufacture the weighting vectors by formula.

4.5.2 Convex Combinations from the Uniform Distribution

A way to overcome the shortcomings of the previous method is as follows. Randomly generate an over-abundance of weighting vectors, then, from the set of convex combinations thus created, filter away the excess by removing those that are most redundant. The efficiency of the approach is dependant on the distribution that we use to compute the weights. To demonstrate, consider again the two-dimensional facet defined by $\underline{c}^1, \underline{c}^2, \underline{c}^3, \underline{c}^4, \underline{c}^5$ in Figure 4.1. To construct the convex combinations $\sum_{i=1}^{5} \lambda_i \underline{c}^i$, let the λ_i convex combination weight be determined by $\lambda_i = \frac{V_i}{\sum_{j=1}^{5} V_j}$ where V_i is drawn from the uniform distribution 0 to 1. As is shown in Figure 4.3, there is a multinormal type of buildup about the geometric centre of the facet when the λ_i convex combination weights are chosen in this way.

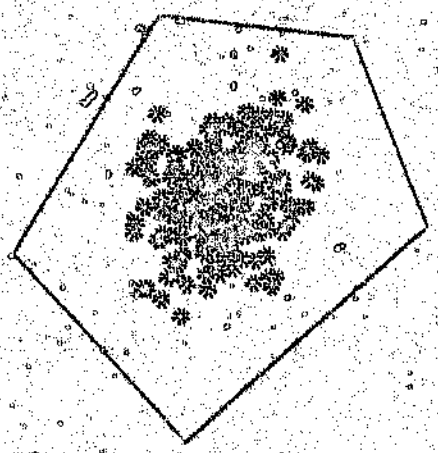


Figure 4.3: Convex Combinations Drawn from the Uniform Distribution

4.5.3 Convex Combinations from the Weibull Distribution

In an effort to counteract this type of buildup about the geometric centre, and place more convex combinations into the corners of a facet, consider making draws V_i from the Weibull distribution. A draw V_i from the Weibull distribution, can be obtained from the expression $V_i = b(-\ln(r))^{\frac{1}{c}}$, where r is a draw from the uniform distribution 0 to 1. Steuer [56] suggests values of b=0.1 and c=0.3. The pattern of resultant convex combinations would take on the form indicated in Figure 4.4.

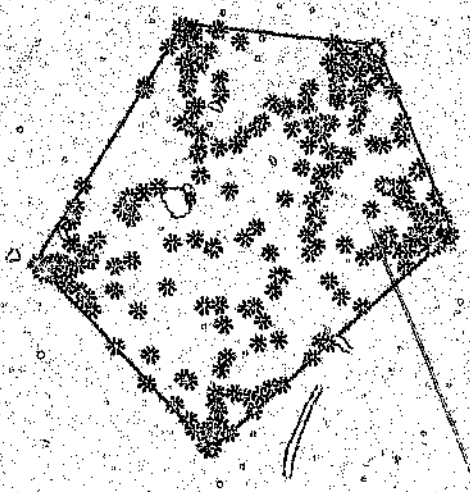


Figure 4.4: Convex Combinations Drawn from the Weibull Distribution

4.5.4 Mixed Strategy

A better idea might be to generate some of the convex combination vectors using draws from the uniform distribution, and the rest using draws from the Weibull distribution. In this way we will not only generate convex combinations in the centre of the polyhedron, but in its corners as well. Then when we forward filter the over-abundance of representatives down to a set of desired size, the representatives surviving the filter should provide a well-dispersed covering of the polyhedron. Figures 4.5 and 4.6 show the convex combinations

generated using a 50-50 and 30-70 strategy respectively. (30-70 strategy meaning 30% from Uniform distribution and 70% from Weibull distribution). The ADBASE MOLP package employs a 50-50 strategy, but it was found that a 30-70 strategy provides a better spread over the space.

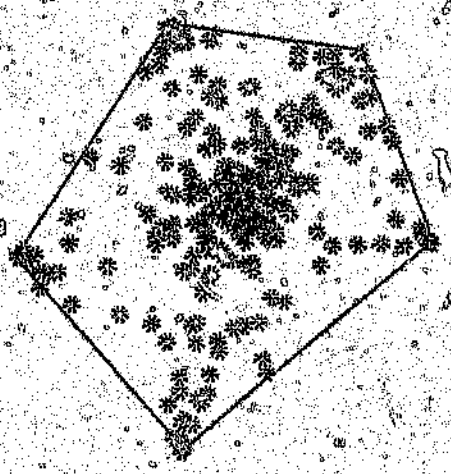


Figure 4.5: Convex Combinations — 50% from Uniform Distribution and 50% from Weibull Distribution

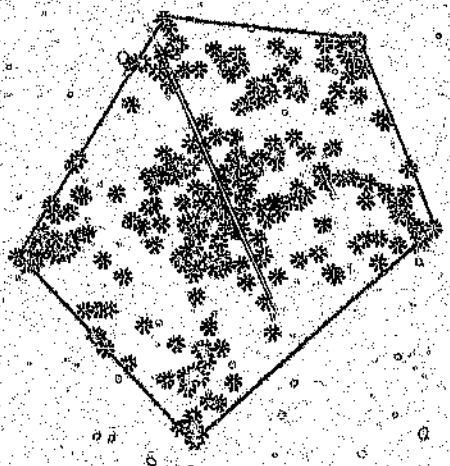


Figure 4.6: Convex Combinations — 30% from Uniform Distribution and 70% from Weibull Distribution

4.5.5 Grid Method

The question now arises whether it would not be better to generate the λ's systematically instead of randomly. We could select a number of values for λ_i from each interval $[l_i, r_i]$ for each i, and combine these in all possible ways that $\sum_{i=1}^{p} \lambda_i = 1$.

Example 4.5 Say $p = 3$ and $l_i = 0, r_i = 1, \forall i = 1 \ldots 3$. Selecting values $0, \frac{1}{2}, 1$ for λ_i and generating all possible combinations, yields the five weighting vectors

$$\begin{aligned}
\underset{\sim}{\lambda}_1 &= (0, 0, 1) \\
\underset{\sim}{\lambda}_2 &= (0, 1, 0) \\
\underset{\sim}{\lambda}_3 &= (\tfrac{1}{2}, 0, \tfrac{1}{2}) \\
\underset{\sim}{\lambda}_4 &= (\tfrac{1}{2}, \tfrac{1}{2}, 0) \\
\underset{\sim}{\lambda}_5 &= (1, 0, 0)
\end{aligned}$$

Applying the grid method with $\lambda_i = 0, \frac{1}{2}, 1 \; \forall i = 1 \ldots 5$, to the surface shown in Figure 4.1 yields the dispersion shown in Figure 4.7.



Figure 4.7: Convex Combinations using the Grid Method

As in the random selection of λ from the uniform distribution, there is a buildup about the geometric centre of the facet. Choosing values of λ near the edges of the interval does not help all that much.

Another disadvantage of this method is that we cannot predetermine the number of points that will be generated.

It is of course possible to generate an over-abundance of weighting vectors in this manner and filter them (as in randomly generated points) but in this case, it is usually easier to generate the points randomly than to generate them systematically.

4.6 Filtering

It was seen in the previous section what the filtering of vectors is used for. We now discuss various methods for achieving this.

Definition 4.6 Let V be a finite set of vectors. Let P denote the forward set size (the size of the subset to be computed.) In *filtering* the set V, we want to obtain the P vectors in V that are most different from one another.

Filtering is accomplished by finding the P vectors in V that are furthest apart from one another according to a given metric. The metric that is used is the weighted L_ρ metric as defined in definition 1.25, whereby the distance between $\underline{x}$ and $\underline{y}$ is

$$\|\underline{x} - \underline{y}\|_\rho = [\sum_{i=1}^{q} (\pi_i \mid x_i - y_i \mid)^\rho]^{\frac{1}{\rho}}$$

where q is the length of vectors being filtered, π_i are the range equalization weights described below, and ρ is the metric parameter.

4.6.1 Range Equalization Weights

We consider again the range equalization weights defined in section 4.4.3. The purpose of the range equalization weights is to equalize the ranges of the vectors in V so that the filtering procedure is not overly influenced by components with large ranges. To compute the range equalization weights, let R_i denote the range of the ith component of the vectors in V where $R_i = \max_{v \in V}\{v_i\} - \min_{v \in V}\{v_i\}$. Then the range equalization weights are

$$\pi_i = \frac{1}{R_i}[\sum_{j=1}^{p} \frac{1}{R_j}]^{-1}$$

4.6.2 Weighted Euclidean Distance Measure

Let $V \subset \Re^q$ be a finite set of vectors. The filtering process operates by comparing distances between pairs of vectors $\underset{\sim}{x}$ and $\underset{\sim}{z}$ according to a weighted metric. In accordance with the Euclidean weighted metric, the distance between $\underset{\sim}{x}$ and $\underset{\sim}{z}$ is given by

$$D = \sum_{i=1}^{q} \sqrt{\pi_i (x_i^2 - z_i^2)}$$

where q is the length of the vectors being filtered and π_i are the range equalization weights as discussed in section 4.6.1.

Notation 4.7 Denote the n vectors to be filtered as $\{\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots, \underset{\sim}{x}_n\}$ and the m vectors which are output by the filtering procedure as $\{\underset{\sim}{z}_1, \underset{\sim}{z}_2, \ldots, \underset{\sim}{z}_m\}$

4.6.3 Neighbourhood Based Algorithms

The Threshold Algorithm

The threshold algorithm is

1. Select a threshold value T
2. Let $\underset{\sim}{z}_1 = \underset{\sim}{x}_1$, $k = 1$, $m = 1$
3. Set $k = k + 1$
4. Calculate all distances D_{kj}, $j = 1 \ldots k-1$, i.e. the distances from $\underset{\sim}{x}_k$ to $\underset{\sim}{z}_j$. If all $D_{kj} > T$ then let $\underset{\sim}{z}_{m+1} = \underset{\sim}{x}_k$ and set $m = m + 1$
5. If $k = n$ then stop with m filtered vectors $\underset{\sim}{z}_1, \underset{\sim}{z}_2, \ldots \underset{\sim}{z}_m$, else go to 3

Example 4.8 Consider the following seven 2-D vectors

$$\begin{aligned}
\underline{x}_1 &= (1, \tfrac{1}{2}) \\
\underline{x}_2 &= (0, 0) \\
\underline{x}_3 &= (2, 0) \\
\underline{x}_4 &= (1, 1) \\
\underline{x}_5 &= (0, 2) \\
\underline{x}_6 &= (\tfrac{1}{2}, 1\tfrac{1}{2}) \\
\underline{x}_7 &= (2, \tfrac{1}{2})
\end{aligned}$$

These vectors are shown in Figure 4.8

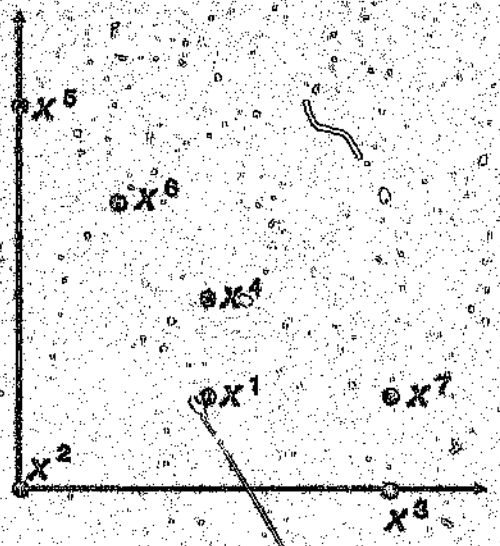


Figure 4.8: Points to be Filtered

Since the ranges are equal, we can ignore the range equalization weights.

Applying the threshold algorithm with $T = 1$:-

1. $T = 1$
2. $\underline{z}_1 = \underline{x}_1 = (1, \frac{1}{2}), k = 1, m = 1$
3. $k = 2$

4. $D_{21} = 1.118 > T$ so $z_2 = (0,0), m = 2$

3. $k = 3$

4. $D_{31} = 1.118 > T$
   $D_{32} = 2 > T$, so $z_3 = (2,0), m = 3$

3. $k = 4$

4. $D_{41} = 0.5 < T$
   $D_{42} = 1.414 > T$
   $D_{43} = 1.414 > T$

3. $k = 5$

4. $D_{51} = 1.803 > T$
   $D_{52} = 2 > T$
   $D_{53} = 2.283 > T$ so $z_4 = (0,2), m = 4$

3. $k = 6$

4. $D_{61} = 1.118 > T$
   $D_{62} = 1.581 > T$
   $D_{63} = 2.121 > T$
   $D_{64} = 0.707 < T$

3. $k = 7$

4. $D_{71} = 1 = T$
   $D_{72} = 2.06 > T$
   $D_{73} = 0.5 < T$
   $D_{74} = 2.5 > T$

5. Stop. The filter has returned the 4 vectors $(1, \frac{1}{2})$ $(0,0)$ $(2,0)$ $(0,2)$

Note that if $T = 1.5$ was used on the data given in Example 4.8, the filtering technique would only have returned the vectors $(1, \frac{1}{2})$ $(0,2)$.

Comments on the Threshold Algorithm

The vectors returned by the threshold algorithm depend heavily on which vector is first in the list, the order of the other vectors, and the value of T. There is unfortunately no appropriate way to determine which value of T will return the required number of vectors. T is found by a 'trial-and-error' method described later.

Although this algorithm has some obvious limitations, it only requires one pass through the sample set to establish the points to be returned by the filter, and is hence computationally attractive. In practice however, the procedure normally requires extensive experimentation with various values of the threshold and (possibly) various starting points in order to achieve the required number of filter points.

Note that if a threshold value of T_1 is used, and P_1 filtered points are generated by the above method, then the above method will yield $P_2 \leq P_1$ points with a threshold of $T_2 < T_1$. These P_2 points are not a subset of the P_1 points generated with a threshold of T_1. Hence no information obtained from one execution of the algorithm is of use in another execution of the algorithm.

The Methods of Closest and Furthest Point Outside the Neighbourhood

These two methods are similar to the threshold algorithm, with the main difference being that the closest or furthest point outside the T-neighbourhood is chosen as the next point returned by the filter, as opposed to the first such vector with minimum distance greater than T.

The method of closest point outside the neighbourhood is :-

1. Select a threshold value T
2. Let $\underset{\sim}{z}_1 = \underset{\sim}{x}_1$. Set $m = 1$ and $V = \{\underset{\sim}{x}_2 \ldots, \underset{\sim}{x}_n\}$

3. Calculate all distances D_{im} for all i such that $\underset{\sim}{x}_i \in V$. If any $D_{im} \leq T$, remove $\underset{\sim}{x}_i$ from V. If V is empty stop with m filtered vectors $\underset{\sim}{z}_1, \underset{\sim}{z}_2, \cdots \underset{\sim}{z}_m$

4. Let $D_i = \sum_{j=1}^{m} D_{ij}$ $\forall i$ such that $\underset{\sim}{x}_i \in V$

5. Let $D_{imin} = \min D_i$ $\forall i$ such that $\underset{\sim}{x}_i \in V$ and let $imin$ be the value of i which minimizes D_i. Let $\underset{\sim}{z}_{j+1} = \underset{\sim}{x}_{imin}$. Remove $\underset{\sim}{x}_{imin}$ from V, set $m = m + 1$ and go to 3.

The method of furthest point outside the neighbourhood is identical to the above algorithm, replacing the minimization in step 5 by a maximization.

Example 4.9 To illustrate the difference between the threshold algorithm and the method of furthest point outside the neighbourhood, the latter is applied to the points given in Example 4.8.

1. $T = 1$

2. $\underset{\sim}{z}_1 = \underset{\sim}{x}_1 = (1, \frac{1}{2}), m = 1, V = \{(0,0), (2,0), (1,1), (0,2), (\frac{1}{2}, 1\frac{1}{2}), (2, \frac{1}{2})\}$

3. $D_{21} = 1.11 > T$
   $D_{31} = 1.118 > T$
   $D_{41} = 0.5 < T$
   $D_{51} = 1.803 > T$
   $D_{61} = 1.118 > T$
   $D_{71} = 1 = T$
   $V = \{(0,0), (2,0), (0,2), (\frac{1}{2}, 1\frac{1}{2})\}$

4. $D_2 = 1.118$
   $D_3 = 1.118$
   $D_5 = 1.803$
   $D_6 = 1.118$

5. $D = 1.803$, $\underset{\sim}{z}_2 = \underset{\sim}{x}_5 = (0,2)$, $V = \{(0,0), (2,0), (\frac{1}{2}, 1\frac{1}{2})\}$, $m = 2$

3. $D_{32} = 2 > T$
   $D_{42} = 2.828 > T$

$D_{62} = 0.7071 < T$

$V = \{(0,0),(2,0)\}$

4. $D_2 = 3.118$

$D_3 = 5.946$

5. $D = 5.946$, $z_3 = x_3 = (2,0)$, $V = \{(0,0)\}$, $m = 3$

3. $D_{23} = 2 > T$

$V = \{(0,0)\}$

4. $D_2 = 5.118$

5. $D = 5.118$, $z_4 = x_2 = (0,0)$, $V = \{\}$, $m = 4$

3. V is empty

4. Stop. The filter has returned the 4 vectors $(1,\frac{1}{2})(0,2)(2,0)(0,0)$

In this case the method of furthest point outside the threshold returns the same vectors as the threshold method.

Comments on the Closest and Furthest Point Methods

These two methods require more computing than the simple threshold method. These methods are still dependent on the first vector in the list and on the value of T, but not on the order of the other vectors.

Maximin Distance Algorithm

This algorithm is similar to the method of furthest point outside the neighbourhood, but we select the vector which maximizes the minimum distance to the filtered vectors, i.e. instead of using $D_i = \sum_{j=1}^{m} D_{ij}$ we use $D_i = \min_j D_{ij}$.

The maximin algorithm is therefore

1. Select a threshold value T
2. Let $z_1 = x_1$. Set $m = 1$ and $V = \{x_2, \ldots, x_n\}$
3. Calculate all distances D_{im} for all i such that $x_i \in V$. If any $D_{im} \leq T$, remove x_i from V. If V is empty stop with m filtered vectors $z_1, z_2, \ldots z_m$
4. Let $D_i = \min_j D_{ij}$ $\forall i$ such that $x_i \in V$
5. Let $D = \max D_i$ $\forall i$ such that $x_i \in V$ and let $imax$ be the value of i which maximizes D_i. Let $z_{j+1} = x_{imax}$. Remove x_{imax} from V, set $m = m + 1$ and goto 3.

Choosing a Threshold Size

We require T to be chosen such that we are left with *exactly* P filtered vectors. Unfortunately, there is no way to determine the appropriate value for T, other than by experimentation. Suppose we want to filter a list of vectors with a value of T and find that the filtered list is too long, then T would have to be increased and the vectors filtered once again. Suppose the new reduced list turns out too short, then T would have to be decreased and the vectors filtered yet another time, and so forth. Thus, when using neighbourhood-based methods, it is necessary to conduct multiple runs with different values of T until we converge to a reduced set of the desired size. To adjust T so that the process of filtering converges to a set of points of the required size, utilize a halving and doubling routine, as set out below.

1. Initialize T_0 and δT. Set $i = 0$.
2. Filter the vectors using threshold value T_0, to yield R_0 filtered vectors.
3. If $R_i < R$ then set $T_{i+1} = T_i - \delta T$.
   If $R_i > R$ then set $T_{i+1} = T_i + \delta T$.
   If $R_i = R$ then STOP.
   Set $i = i + 1$.

4. Filter vectors using threshold value T_i to obtain R_i filtered vectors.
   If $R > R_i \geq R_{i-1}$ or $R < R_i \leq R_{i-1}$ then set $\delta T = 2\delta T$ and go to 3. Else set $\delta T = \delta T/2$ and go to 5.

5. If $R_i < R$ then set $T_{i+1} = T_i - \delta T$.
   If $R_i > R$ then set $T_{i+1} = T_i + \delta T$.
   If $R_i = R$ then STOP.
   Set $i = i + 1$.

6. Filter vectors using threshold value T_i to obtain P_i filtered vectors. Set $\delta T = \delta T/2$ and go to 5.

Steuer [56] suggests that 'good' initial values are

$$T_0 = \frac{[\sum_{i=1}^{q}(\pi_i R_i)^p]^{\frac{1}{p}}}{4}$$

$$\delta T = \frac{T_0}{2}$$

Consider the randomly generated convex combinations distribution shown in Figure 4.6. The threshold algorithm, the methods of closest and furthest point outside the neighbourhood, and the maximin algorithm are applied to this data, together with the halving and doubling routine, to filter it down to 15 points. The results of the filtering are shown in Figures 4.9 to 4.12. The differences in results are not seen to be dramatically different. Steuer [56] claims that the method of furthest point outside the neighbourhood produces the best results.

Unattained Set Sizes

It may not be possible to produce a reduced set of given size using the four methods described above due to 'ties', i.e. if $D(\underset{\sim}{x}_i, \underset{\sim}{z}_j) = D(\underset{\sim}{x}_k, \underset{\sim}{z}_j)$ then either both $\underset{\sim}{x}_i$ and $\underset{\sim}{x}_k$ will be returned by the filter, or neither will be returned by the filter.

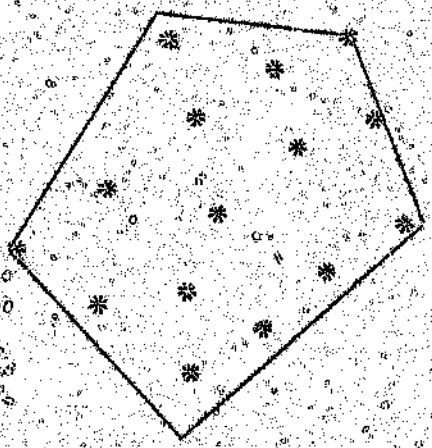


Figure 4.9: The 15 Points Returned by the Threshold Algorithm

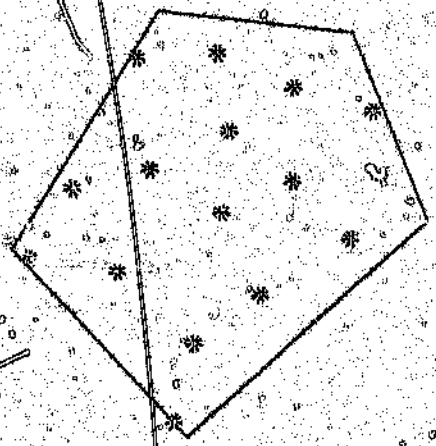


Figure 4.10: The 15 Points Returned by the Closest Point Outside the Neighbourhood Algorithm

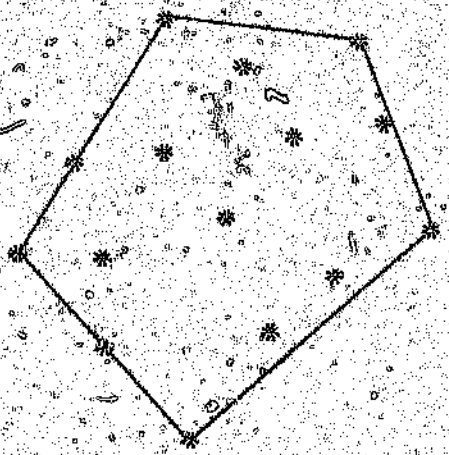


Figure 4.11: The 15 Points Returned by the Furthest Point Outside the Neighbourhood Algorithm

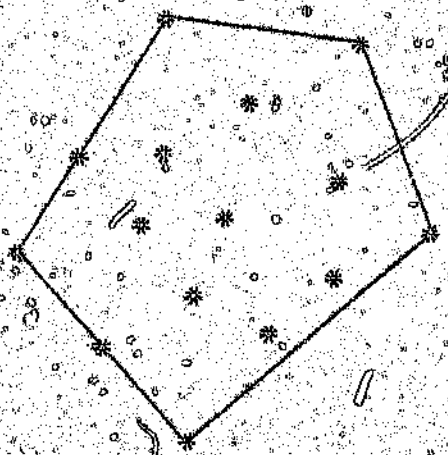


Figure 4.12: The 15 Points Returned by the Maximin Algorithm

Maximally Dispersed Subsets

The real goal of filtering is to produce maximally dispersed subsets. Let V be a finite set of vectors. By a maximally dispersed subset V of size P, we mean the set of P points whose members are further apart from one another than any set of points in V. To obtain a maximally dispersed subset requires considerable computation and hence we usually accept "approximately" maximally dispersed subsets. Such subsets can be produced by using one of the threshold algorithms discussed earlier in this section.

4.6.4 Means Algorithms

Means algorithms do not suffer from the problems of choosing a threshold value T since they *generate* P vectors, as opposed to threshold based algorithms which select some of the N vectors in V. In other words, threshold-based algorithms select P of the N vectors, while means-based algorithms generate P new vectors. Means-based algorithms are therefore not truly filtering algorithms but may be used in the case where the input vectors are randomly generated. While there are many means-based algorithms, we consider the simplest.

The K-Means Algorithm

1. Select P initial filtered vectors $z_1^{(1)}, \cdots z_P^{(1)}$. These are usually selected as the first P vectors in V, $(x_1, \cdots, x_P)$. Set $k = 1$.

2. Form P subsets of the vectors in V, using the relation

$$\underline{x} \in X_j^{(k)} \text{ if } \|\underline{x} - \underline{z}_j^{(k)}\| < \|\underline{x} - \underline{z}_i^{(k)}\| \text{ for all } i = 1, 2, \ldots, P, \; i \neq j$$

where $X_j^{(k)}$ denotes the subset of vectors which are closer to $z_j^{(k)}$ than any other filtered vector. Ties are resolved arbitrarily.

3. Compute new filtered vectors $z_j^{(k+1)}, j = 1 \ldots P$ as the mean of all vectors x in $X_j^{(k)}$. i.e. $z_j^{(k+1)} = \frac{1}{N_j} \sum_{x \in X_j^{(k)}} x \quad j = 1, \ldots, P$ where N_j is the number of vectors in $X_j^{(k)}$. The name "K-means" is derived from the manner in which filtered vectors are sequentially updated.

4. If $z_j^{(k+1)} = z_j^{(k)}$ for $j = 1 \ldots P$, the algorithm has converged and the procedure is terminated. Otherwise go to 2.

While the K-means algorithm is computationally simple, the results it produces are useless for the purpose we require. This is because the vectors returned by the filter are obtained by averaging vectors in V and hence tend to cluster toward the centre of the space. This negates the work we did by generating a number of points using the Weibull distribution (section 4.5.3). The results of using the K-means algorithm on our randomly-generated data is shown in Figure 4.13.

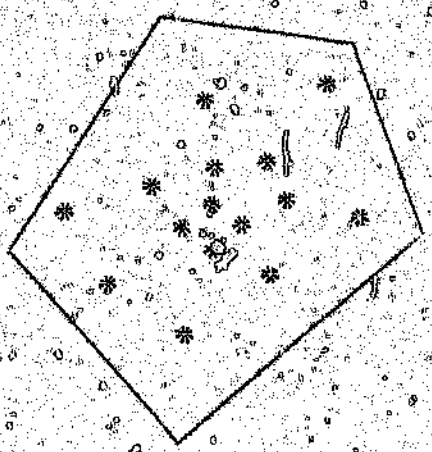


Figure 4.13: The 15 Points Returned by the K-means Algorithm

4.7 Contracting the Criterion Cone

4.7.1 Interval Weights Criterion Cone

Notation 4.10 Let us call the original criterion cone the *C-cone* and the reduced criterion cone the *D-cone*

Definition 4.11 Consider a convex combination weighting vector that, when applied to the generators of the C-cone, specifies a generator of the D-cone. Such a convex combination weighting vector is called a *critical vector*

With a critical vector for each generator of the interval criterion weights cone, criterion matrix D can be constructed as follows :-

1. Obtain all q critical vectors. Form the $q \times p$ matrix T whose ith row is the ith critical vector.
2. $D = TC$

The problem with this approach is that the number of generators of the D-cone increases rapidly with p.

Example 4.12 (From Steuer [56])
Consider the cross-section of the 3-D cone generated by $\underline{c}^1, \underline{c}^2, \underline{c}^3$ in Figure 4.14, along with

$$\lambda_1 \in [0.2, 0.7]$$
$$\lambda_2 \in [0.2, 0.5]$$
$$\lambda_3 \in [0.1, 0.8]$$

As seen from Figure 4.14, each $\underline{d}^j$ is defined by $p-1$ endpoints of the $[l_i, r_i]$ intervals. Generator $\underline{d}^j$ is defined by $l_1 = 0.2$ and $l_2 = 0.2$. This implies that

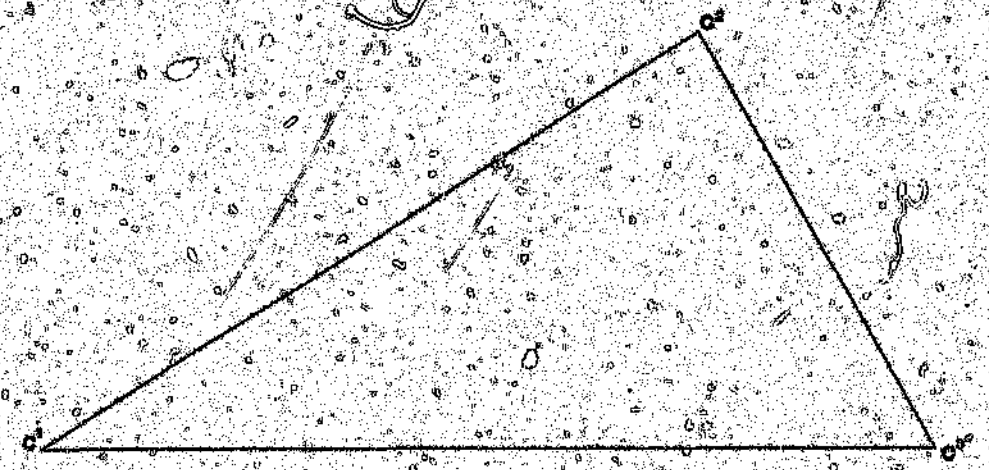


Figure 4.14: The C-cone

$(0.2, 0.2, 0.6)$ is a critical vector because

$$0.2\underline{c}^1 + 0.2\underline{c}^2 + 0.6\underline{c}^3 = \underline{d}^1$$

Similarly, $\underline{d}^2$ is defined by $l_{12} = 0.2, r_{22} = 0.5$ so $(0.2, 0.5, 0.3)$ is the critical vector for $\underline{d}^2$, and so forth.

$$\text{So } T \text{ is } \begin{bmatrix} 0.2 & 0.2 & 0.6 \\ 0.2 & 0.5 & 0.3 \\ 0.7 & 0.2 & 0.1 \\ 0.4 & 0.6 & 0.1 \end{bmatrix}$$

Now $\underline{D} = TC$ so our new criterion cone has 4 generators instead of 3.

The reduced cone is shown in Figure 4.15.

4.7.2 Enveloping Reduced Criterion Cone

To overcome the problem of the number of generators becoming large, we consider the *enveloping reduced criterion cone* (E-cone). The purpose of the

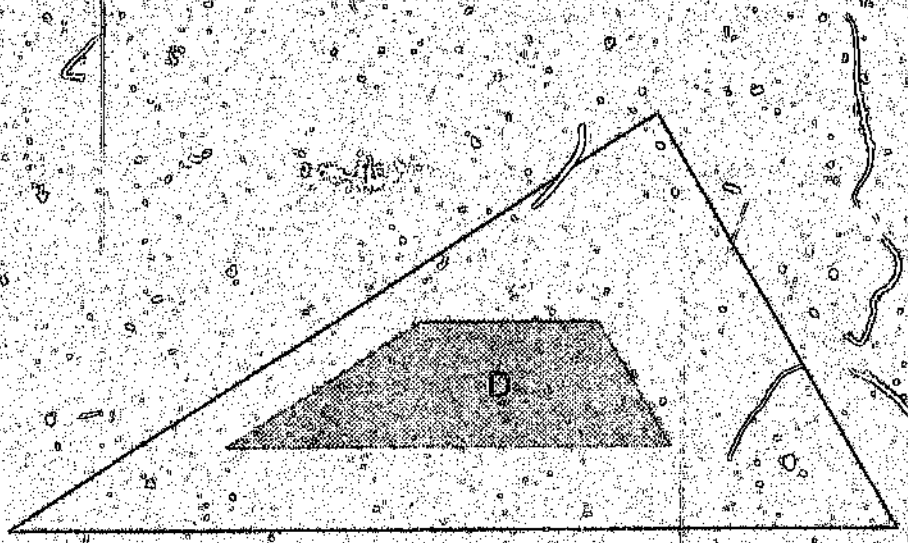


Figure 4.15: Reduced D-cone

E-cone is to envelop the D-cone in a larger cone that has at most p extreme rays.

Since E has, at most, p rows, we do not have the extra computational burden as with the D-cone, but the reduced cone is larger than required and hence more iterations may have to be done to achieve the same accuracy.

The critical vectors for the E-cone are determined rapidly by forming the intervals

$$\begin{matrix} [\hat{l}_1, 1] \\ [\hat{l}_2, 1] \\ \vdots \\ [\hat{l}_p, 1] \end{matrix}$$

where $\hat{l}_i = \max\{l_i, (1 - \sum_{\substack{j=1 \\ j \neq i}}^{p} r_j)\}$.

Then the ith row of T is

$$[\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_{i-1}, (1 - \sum_{\substack{j=1 \\ j \neq i}}^{p} \hat{l}_j), \hat{l}_{i+1}, \ldots, \hat{l}_p]$$

Then $E = TC$.

Example 4.13 Consider again the cone and weights given in example 4.12.

$$T \text{ is } \begin{bmatrix} 0.2 & 0.2 & 0.6 \\ 0.2 & 0.7 & 0.1 \\ 0.7 & 0.2 & 0.1 \end{bmatrix}$$

Now $E = TC$ so our new criterion cone has only 3 generators.

The reduced cone is shown in Figure 4.16. The dotted line indicates where the D-cone ended.

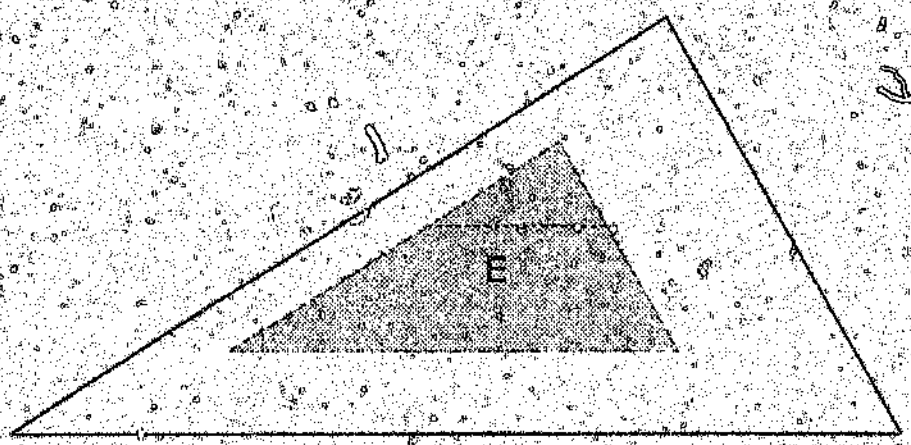


Figure 4.16: Reduced E-cone

4.7.3 Steuer's E-cone Contractions

Recall that Steuer's method of contracting cones constructs the $2p+1$ vectors

$$\begin{aligned}
\underset{\sim}{\lambda}^1 &= (1,0,0,\ldots,0)\\
\underset{\sim}{\lambda}^2 &= (0,1,0,\ldots,0)\\
&\vdots\\
\underset{\sim}{\lambda}^p &= (0,0,\ldots,0,1)\\
\underset{\sim}{\lambda}^{p+1} &= \left(\tfrac{1}{p^2},\tfrac{p+1}{p^2},\tfrac{p+1}{p^2},\ldots,\tfrac{p+1}{p^2}\right)\\
\underset{\sim}{\lambda}^{p+2} &= \left(\tfrac{p+1}{p^2},\tfrac{1}{p^2},\tfrac{p+1}{p^2},\ldots,\tfrac{p+1}{p^2}\right)\\
&\vdots\\
\underset{\sim}{\lambda}^{2p} &= \left(\tfrac{p+1}{p^2},\tfrac{p+1}{p^2},\ldots,\tfrac{p+1}{p^2},\tfrac{1}{p^2}\right)\\
\underset{\sim}{\lambda}^{2p+1} &= \left(\tfrac{1}{p},\tfrac{1}{p},\tfrac{1}{p},\ldots,\tfrac{1}{p}\right)
\end{aligned}$$

and each of these λ-vectors is used to generate a point on the criterion cone $\sum_{i=1}^{p}\lambda_i c^i$. Suppose that we wish to construct an E-cone whose cross-sectional volume is $\frac{1}{p}$th the cross-sectional volume of the C-cone, about one of these generated points, say the one generated by $\underset{\sim}{\lambda}^k$. The interval criterion weights that yield the critical vectors are (from Steuer[56] p. 252) :

$$[l_k, r_k] = \begin{cases} [r,1] & \text{if } 0 \le k \le p \\ [0,1] & \text{if } p+1 \le k \le 2p \\ [\frac{r}{k},1] & \text{if } k = 2p+1 \end{cases}$$

$$[l_i, r_i] = \begin{cases} [0,1] & \text{if } 0 \le k \le p \\ [\frac{r}{k-1}] & \text{if } p+1 \le k \le 2p \\ [\frac{r}{p},1] & \text{if } k = 2p+1 \end{cases} \qquad i = 1,\ldots,p, i \ne k$$

where $r = 1 - k\frac{1}{p+1}$

Example 4.14 Consider again the C-cone shown in Figure 4.14. Then the λ-vectors are

$$\begin{aligned}
\underset{\sim}{\lambda}^1 &= (1,0,0)\\
\underset{\sim}{\lambda}^2 &= (0,1,0)\\
\underset{\sim}{\lambda}^3 &= (0,0,1)\\
\underset{\sim}{\lambda}^4 &= (\tfrac{4}{9},\tfrac{4}{9},\tfrac{1}{9})\\
\underset{\sim}{\lambda}^5 &= (\tfrac{4}{9},\tfrac{1}{9},\tfrac{4}{9})\\
\underset{\sim}{\lambda}^6 &= (\tfrac{1}{9},\tfrac{4}{9},\tfrac{4}{9})\\
\underset{\sim}{\lambda}^7 &= (\tfrac{1}{3},\tfrac{1}{3},\tfrac{1}{3})
\end{aligned}$$

Applying these points to the cone generators yields the points shown on Figure 4.17. The reduced E-cone about f^2, f^4 and f^7 is shown in Figures 4.18, 4.19 and 4.20 respectively.

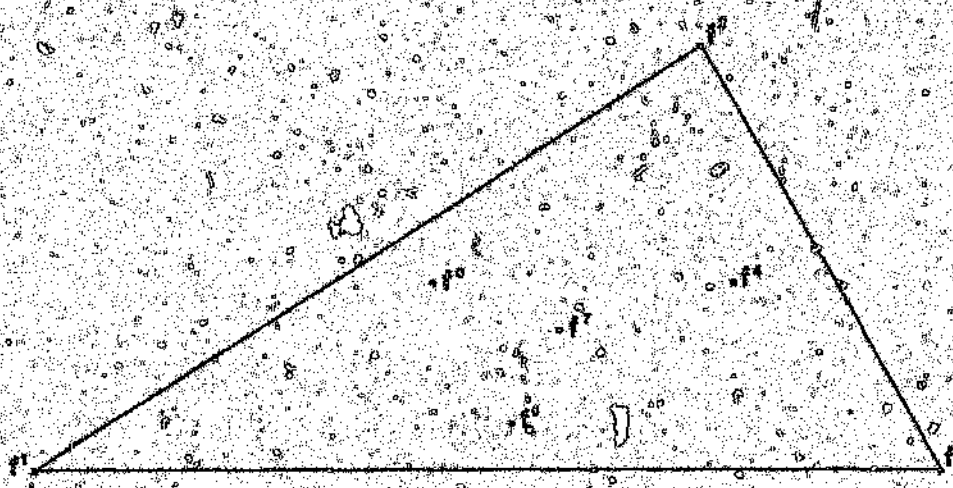


Figure 4.17: Points Generated by Steuer's Weighting Vectors

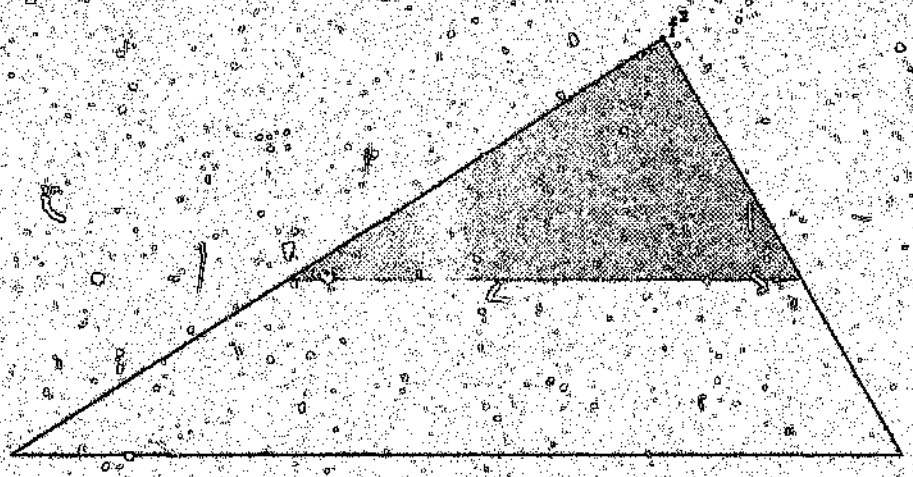


Figure 4.18: Reduced E-cone about f^2

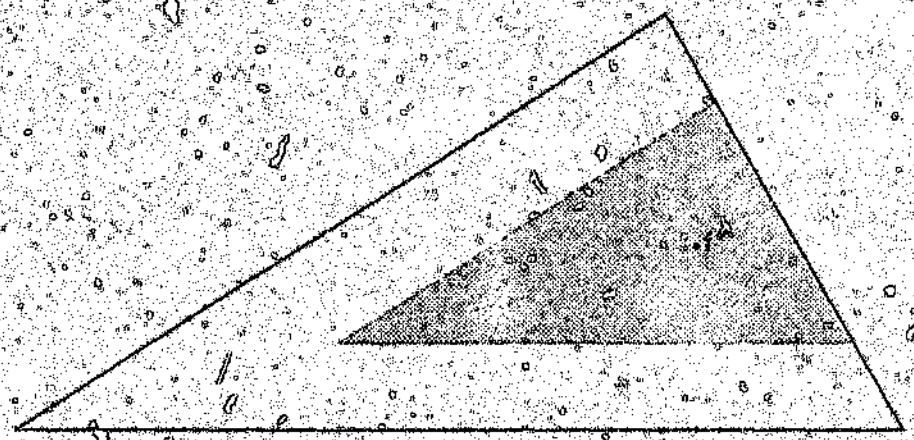


Figure 4.19: Reduced E-cone about f^4



Figure 4.20: Reduced E-cone about f^7

Chapter 5

Algorithm Breakdown

In order to compare the algorithms and identify their similarities and differences, as well as to be able to identify which algorithm is appropriate for a particular case, the algorithms are now broken down into their constituent steps.

The seven algorithms being considered all have the following basic structure

1. Generate weights $\underset{\sim}{\lambda}$ subject to some constraints.
2. Use $\underset{\sim}{\lambda}$ to generate some feasible solution $\underset{\sim}{x}$ with corresponding objective function values $\underset{\sim}{z}$, also subject to certain constraints.
3. Check whether $(\underset{\sim}{x}, \underset{\sim}{z})$ is the best compromise solution. If so, stop, otherwise go to 1.

Four out of the seven methods have an additional step, viz. add additional constraints which restrict the weights or solutions which can be generated in steps 1 and 2.

These four procedures are investigated individually.

In addition, we consider the way in which the algorithm interacts with the DM. This interaction is crucial in the sense that the algorithms are usually heavily reliant on the information obtained. The easier it is for the DM to provide the required information, the greater the accuracy of the information, and hence the better the algorithm is expected to perform. In addition, DMs tend to judge algorithms by their method of interaction as this is the only direct association they have with the algorithm.

5.1 Generating Weights

Four ways of generating weights occur in the algorithms under consideration, viz.

1. Generate k such λ-vectors and allow the DM to choose his preferred one.
2. Ask the DM certain questions, and constrain the weights on the basis of the DM's responses. Find some $\underset{\sim}{\lambda}$ which satisfies these constraints.
3. Obtain a reference point from the DM and use this to generate $\underset{\sim}{\lambda}$.
4. Ask the DM to enter the weights directly.

We consider these individually.

5.1.1 Letting the DM Choose λ from a Set

The principle here is simple; discretize the set of all allowable $\underset{\sim}{\lambda}$'s i.e. represent the set by a number of points in the set, and then allow the DM to choose one. In practice, since the DM usually does not know what his true weights are, each set of weights is used to generate a feasible solution. These are presented to the DM who must indicate which he likes most. The set of weights which generated the DM's preferred solution is assumed to be the DM's preferred set of weights. The ways in which the set of allowable $\underset{\sim}{\lambda}$'s can be discretized is described in section 4.5.

This technique is used by Steuer's Contracting Cones method, the Interactive Weighted Sums/Filtering approach and the Interactive Weighted Tchebbycheff method.

5.1.2 Choosing λ on the Basis of DM's Responses to Various Questions

Assume we are at a solution $\underset{\sim}{x}^h$, and the DM is shown an alternative solution $\underset{\sim}{x}^q$. Assuming that the DM's utility function at $\underset{\sim}{x}^h$ can be approximated by $\sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}^h)$, then the DM should prefer $\underset{\sim}{x}^h$ to $\underset{\sim}{x}^q$ if

$$\sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}^h) > \sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}^q),$$

$\underset{\sim}{x}^q$ to $\underset{\sim}{x}^h$ if

$$\sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}^h) < \sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}^q),$$

and be indifferent if

$$\sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}^h) = \sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}^q).$$

Equivalently, if we let $\underset{\sim}{t} = \underset{\sim}{f}(\underset{\sim}{x}^h) - \underset{\sim}{f}(\underset{\sim}{x}^q)$, then $\underset{\sim}{t}$ represents the tradeoff vector showing how the DM may trade one objective off against another by heading towards $\underset{\sim}{x}^q$.

Then the DM should like the tradeoff $\underset{\sim}{t}$ if

$$\sum_{i=1}^{p} \lambda_i t_i > 0,$$

dislike the tradeoff if

$$\sum_{i=1}^{p} \lambda_i t_i < 0,$$

and be indifferent if

$$\sum_{i=1}^{p} \lambda_i t_i = 0.$$

We can thus estimate $\underset{\sim}{\lambda}$ by presenting the DM with some alternative solutions or tradeoffs, formulating the constraints as above according to his responses, and then finding a set of weights $\underset{\sim}{\lambda}$ which satisfy the constraints.

This constraint set on the λ's is a slice through the p-dimensional cone passing through the origin (or arbitrarily close to the origin). The method of Zionts and Wallenius solve a linear programming problem to find λ satisfying these constraints, hence one of the extreme point weights will be selected. Consider, for example, the three dimensional one shown in figure 5.1. The weights

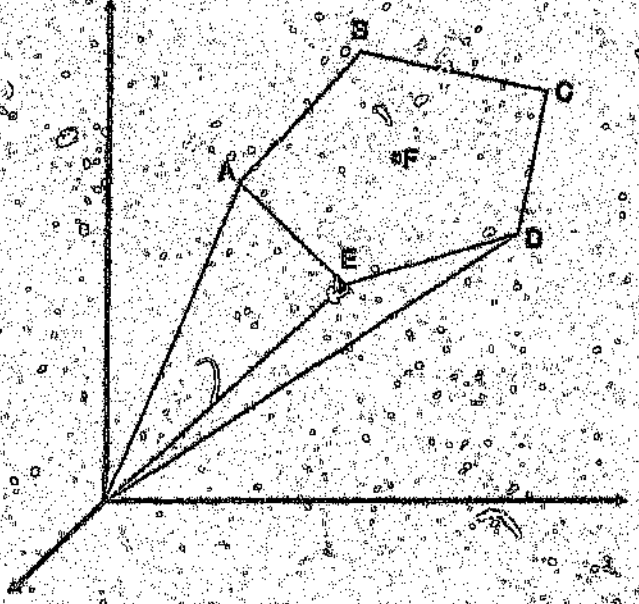


Figure 5.1: The Weights which Satisfy the DM's Constraints

selected will be one of the points A, B, C, D or E.

The Cutting Plane method would generate the point F. This is done by adding/subtracting positive slack/surplus variables to each constraint, and

maximizing the sum of these variables, i.e. by solving

$$\begin{aligned}
\max \quad & \alpha \\
\text{subject to} \quad & \sum_{i=1}^{p} t_{ij}\lambda_i + s_j = 0 \quad \forall \text{ tradeoffs } t_j \text{ the DM likes} \\
& \sum_{i=1}^{p} t_{ij}\lambda_i - s_j = 0 \quad \forall \text{ tradeoffs } t_j \text{ the DM dislikes} \\
& \sum_{i=1}^{p} \lambda_i = 1 \\
& s_j \geq \alpha \qquad \forall j \\
& \lambda_i \geq \alpha \qquad \forall i = 1 \ldots p \\
& \alpha \geq 0
\end{aligned}$$

5.1.3 Using a Reference Point

If the DM expresses his aspirations as a criterion vector $\underline{y}$, where $\underline{y} < \underline{z}^*$. Then we can use weights

$$\lambda_i = \begin{cases} \dfrac{1}{|z_i^* - y_i|} \left[\displaystyle\sum_{j=1}^{p} \frac{1}{|z_j^* - y_j|} \right]^{-1} & \text{if } y_i \neq z_i^* \ \forall i \\ 1 & \text{if } y_i = z_i^* \\ 0 & \text{if } y_i \neq z_i^* \text{ but } \exists j \text{ such that } y_j = z_j^* \end{cases}$$

This choice of $\underline{\lambda}$ is used when the Tchebbycheff norm is used to produce a new iterate. This particular choice is selected because then $\underline{y}$ will be a *definition point* of the contours of the weighted Tchebbycheff norm. In other words, if $\underline{y}$ is feasible and nondominated, then this choice of weights is the one which produces $\underline{y}$ when solving

$$\begin{aligned}
\min \quad & \max_{i=1,\ldots,p} \lambda_i \left| \underline{z}^* - \underline{f}(\underline{x}) \right| \\
\text{s.t.} \quad & A\underline{x} \leq \underline{b} \\
& \underline{x} \geq \underline{0}
\end{aligned}$$

This approach is used by the Reference Point method and the Interactive Weighted Tchebbycheff method. The former uses a user-given reference point,

while the latter presents the DM with a number of solutions from which he selects his most preferred. This selected solution is then used as the reference point.

5.1.4. Asking the DM for Weights

This approach requires the DM to estimate his weights at a point. Although this approach seems like the simplest, it is the most difficult approach for the DM.

This approach is used by the Visual Interactive Method.

5.2 Generating Solutions

Having found some estimate $\underset{\sim}{\lambda}$ of the true weights $\nabla U(\underset{\sim}{f}(\underset{\sim}{x}))$, we now wish use this to generate some feasible solution $\underset{\sim}{x}$ with corresponding criterion vector $\underset{\sim}{z}$.

5.2.1 Weighted Sums LP

Since our aim is to solve the problem

$$\begin{aligned} \max \quad & z = U(\underset{\sim}{f}(\underset{\sim}{x})) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

we solve the approximation to the above problem

$$\begin{aligned} \max \quad & z = \sum_{i=1}^{p} \lambda_i f_i(\underset{\sim}{x}) \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

Note that this method will only generate nondominated corner point solutions.

This technique is used by the Interactive Weighted Sums/Filtering approach, and the method of Zionts and Wallenius.

5.2.2 Reduced Criterion Cone Weighted Sums LP

A variation of the above technique is to solve the weighted sums LP, but with a reduced criterion cone, i.e.

$$\begin{aligned} \max \quad & z = \underset{\sim}{\lambda}^T TC\underset{\sim}{x} \\ \text{subject to} \quad & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

where T is the reduction matrix. The various ways in which T can be defined are described in section 4.7

Note that if T is the identity matrix, then this approach is identical to the previous approach. In effect we are solving the weighted sums LP but using slightly altered objective functions.

This method will also only generate nondominated corner point solutions.

This approach is taken by Steuer's Contracting Cones method.

5.2.3 Increasing the Utility Function

If U is assumed to be concave, then

$$\begin{aligned} U(\underline{f}(\underline{x}^*)) - U(\underline{f}(\underline{x}^{(h)})) &\leq (\underline{f}(\underline{x}^*) - \underline{f}(\underline{x}^{(h)}))^T \nabla U(\underline{f}(\underline{x}^{(h)})) \\ &\leq \max_{\underline{x} \in X} \{(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(h)}))^T \nabla U(\underline{f}(\underline{x}^{(h)}))\} \end{aligned} \tag{5.1}$$

If $\max_{\underline{x} \in X} \{(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(h)}))^T \nabla U(\underline{f}(\underline{x}^{(h)}))\}$ has a positive optimal objective value, then its optimal solution, $\underline{y}$ say, yields a direction of improvement $(\underline{y} - \underline{f}(\underline{x}))$.

Given a set of previous iterates $\underline{x}^{(j)}$, $j = 1 \ldots h$, we attempt to maximize the minimum value of $(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(j)}))^T \nabla U(\underline{f}(\underline{x}^{(j)}))$, $j = 1 \ldots h$ over $\underline{x} \in X$ in order to obtain a direction that is improving with respect to each of the previously generated iterates.

This is achieved by solving

$$\begin{aligned} \max \quad & \alpha \\ \text{subject to} \quad & \alpha \leq (\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(j)}))^T \nabla U(\underline{f}(\underline{x}^{(j)})) \quad \forall j = 1 \ldots h \\ & A\underline{x} \leq \underline{b} \\ & \alpha \geq 0 \\ & \underline{x} \geq \underline{0} \end{aligned}$$

where $\lambda_i^{(k)} = \frac{\partial Q}{\partial f_i} / \frac{\partial Q}{\partial f_1}$

This technique generates nondominated solutions which do not necessarily lie on a corner point of the feasible set.

Only the Cutting Plane method uses this technique.

5.2.4 Reference Point Projection

Another technique is to choose a solution such that we minimize the distance to some reference point. This is Wierzbicki's scalarizing function approach [63]. All the algorithms under consideration which use this approach use the weighted Tchebbycheff norm as a scalarizing function, i.e.

$$s(\underset{\sim}{f}(\underset{\sim}{x}) - \underset{\sim}{y}, \lambda) \stackrel{\Delta}{=} \max_{i=1,\ldots,p} \lambda_i \mid \underset{\sim}{f}(\underset{\sim}{x}) - \underset{\sim}{y} \mid$$

where $\underset{\sim}{y}$ is the reference point.

This is equivalent to solving

$$\begin{aligned} \min \quad & \alpha \\ \text{subject to} \quad & \alpha \geq \lambda_i(y_i - z_i) \quad \forall i \\ & z_i = f_i(\underset{\sim}{x}) \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \end{aligned}$$

In practice we use the augmented Tchebbycheff norm (see definition 1.27) to ensure that only nondominated solutions are generated. i.e. we solve the problem

$$\begin{aligned} \min \quad & \alpha + \epsilon \sum_{i=1}^{k} z_i \\ \text{subject to} \quad & \alpha \geq \lambda_i(y_i - z_i) \quad \forall i \\ & z_i = f_i(\underset{\sim}{x}) \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \\ & \epsilon > 0 \text{ and small} \end{aligned} \tag{5.2}$$

- This technique generates nondominated solutions which do not necessarily lie on a corner point of the feasible set.

This technique is used by the Reference Point method and the Interactive Weighted Tchebbycheff algorithm. The former uses a DM-given reference point while the latter uses the ideal solution as a reference point.

5.2.5 Unbounded Line Segment Projection

In the above technique, a reference point was projected onto the nondominated criterion space. Consider now projecting an entire unbounded line segment onto the nondominated criterion space.

Problem (5.2) can be rewritten as

$$\begin{aligned} \min \quad & \alpha + \varepsilon \sum_{i=1}^{k} z_i \\ \text{subject to } & z_i + \frac{\alpha}{\lambda_i} \geq y_i \quad \forall i \\ & z_i = f_i(\underset{\sim}{x}) \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \\ & \varepsilon > 0 \text{ and small} \end{aligned} \tag{5.3}$$

If we wish to project the unbounded line segment emanating at some reference point $\underset{\sim}{y}$ in the direction $\underset{\sim}{d}$, then (5.3) becomes the achievement scalarizing parametric program

$$\begin{aligned} \min \quad & \alpha + \varepsilon \sum_{i=1}^{k} z_i \\ \text{subject to } & z_i + \frac{\alpha}{\lambda_i} \geq y_i + \theta d_i \quad \forall i \\ & z_i = f_i(\underset{\sim}{x}) \\ & A\underset{\sim}{x} \leq \underset{\sim}{b} \\ & \underset{\sim}{x} \geq \underset{\sim}{0} \\ & \varepsilon > 0 \text{ and small} \end{aligned} \tag{5.4}$$

θ is the parameter which gives the varibus feasible solutions as it is varied from 0 to ∞.

This technique generates a whole series of nondominated solutions which can lie anywhere in the feasible set.

This technique is used by the Visual Interactive Method.

5.3 Imposing Additional Constraints

The philosophy behind constraining solutions which can be generated, is to exclude solutions which are definitely not optimal. The constraints can be imposed in terms of the weight space, the variable space, or the criterion space, but they all have the same effect, viz. preventing certain solutions from being generated. This constraining continues until we "home in" on the optimal solution.

The various reduction techniques employed by the algorithms are now discussed briefly.

5.3.1 Reducing the Weight Space Using Bounds

As discussed in section 5.1.1, one of the methods of selecting $\underset{\sim}{\lambda}$ is to generate numerous $\underset{\sim}{\lambda}$'s and ask the DM to choose his most preferred. Assume the DM chooses $\underset{\sim}{\lambda}^{(h)}$. We assume that the further any $\underset{\sim}{\lambda}$ is from $\underset{\sim}{\lambda}^{(h)}$, the less the DM likes it and hence we "cut out" the $\underset{\sim}{\lambda}$'s which are most different from $\underset{\sim}{\lambda}^{(h)}$. We use the following technique

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

where $\underset{\sim}{\lambda}^{(h)}$ is the preferred set of weights, r is a reduction factor, h is the iteration counter and $[l_i^{(h)}, r_i^{(h)}]$ are the bounds on λ_i at the hth iteration.

It is possible that the optimal weights, and hence the optimal solution, could be cut out by the above reduction.

This technique is used by the Interactive Weighted Sums/Filtering approach and the Interactive Weighted Tchebbycheff method.

5.8.2 Reducing the Criterion Cone

Having found the DM's preferred weights, we could use these to tighten the bounds on the λ_i's, as above. This tightening of the bounds is effectively the same as contracting the cone. Consider generating weights randomly where $[l_i, r_i] = [0, 1] \quad \forall i$. The resulting convex combinations $\sum_{i=1}^{p} \lambda_i \xi^i$ are seen in figure 5.2. If we now generate weights randomly where $[l_i, r_i] = [0.05, 0.95] \quad \forall i$,

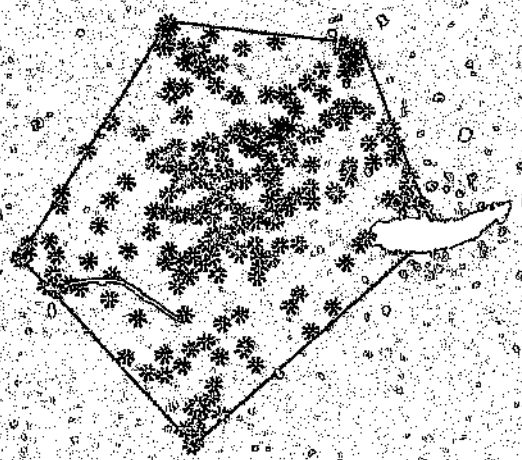


Figure 5.2: Convex Combinations when $\lambda_i \in [0, 1] \; \forall i$

we get the convex combinations shown in figure 5.3. Clearly, tightening the

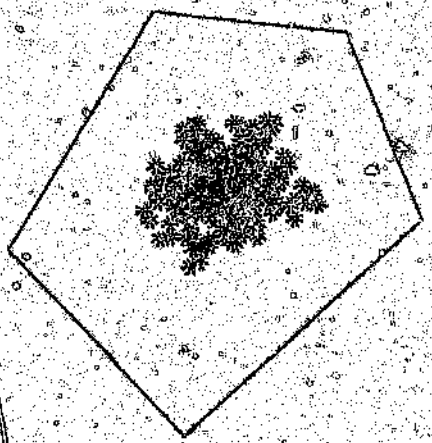


Figure 5.3: Convex Combinations when $\lambda_i \in [0.05, 0.95] \; \forall i$

bounds on the λ_i has the same effect as contracting the cone.

The way in which the cone can be contracted was discussed in section 4.7.

Once again it is possible for the optimal solution to be cut out by this reduction.

This method is used by Steuer's Contracting Cones method.

5.3.3 Reducing the Weight Space Using Constraints

Recall from section 5.1.2 that if the DM likes a tradeoff $\underline{t}$ then

$$\lambda_1 f_1(\underline{t}) + \lambda_2 f_2(\underline{t}) + \cdots + \lambda_p f_p(\underline{t}) > 0$$

and if the DM does not like the tradeoff then

$$\lambda_1 f_1(\underline{t}) + \lambda_2 f_2(\underline{t}) + \cdots + \lambda_p f_p(\underline{t}) < 0$$

Hence, once the DM expresses his opinion about a tradeoff, we set up the above constraints which reduce the weight space.

As with the previous weight-space reduction method, it is possible for the optimal weights, and hence the optimal solution, to be cut out.

This technique is the one used by the method of Zionts and Wallenius.

5.3.4 Reducing the Variable Space

Assuming that U is concave then

$$U[\underline{f}(\underline{x})] \leq U(\underline{f}(\underline{x}^{(h)})) + (\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(h)}))^T \nabla U(\underline{f}(\underline{x}^{(h)})) \quad (5.5)$$

Since we are interested in generating solutions which are better than the current one, we henceforth restrict ourselves to the region such that

$$U(\underline{f}(\underline{x})) > U(\underline{f}(\underline{x}^{(h)}))$$

where $\underline{x}^{(h)}$ is the current solution. From equation (5.5), this implies

$$(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(h)}))^T \nabla U(\underline{f}(\underline{x}^{(h)})) > 0$$

or, dividing through by $\frac{\partial U}{\partial f_1} / \frac{\partial U}{\partial f_i}$,

$$(\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(h)}))^T \lambda_i^{(h)} > 0$$

where $\lambda_i^{(h)} = \frac{\partial U}{\partial f_1} / \frac{\partial U}{\partial f_i}$.

These are constraints in the criterion space, which are in effect constraints on the variable space.

This technique will not cut out the optimal solution provided the utility function is concave.

This is the technique used by the Cutting Plane Algorithm.

5.3.5 No Reduction

This technique may be termed the "learning curve" technique. It presumes that the DM learns about the system and how it reacts to his input, and can therefore reach his optimal solution by giving a certain set of information to the algorithm. There is no reduction whatsoever. The advantage of this technique over the others is that there is no chance of "cutting out" the optimal solution. Its disadvantage however, is that it usually requires more complex information from the DM.

Neither the Reference Point method, nor the Visual Interactive method make use of any reduction.

5.4 Stopping Conditions

There are five simple stopping rules which are used by the algorithms discussed in this study.

5.4.1 Stopping When There is Only One Feasible Solution

When using one of the reduction methods described in the previous section, we reduce the set of feasible solutions continuously. Eventually, the set should contain only one solution. In methods which only generate solutions which lie on a corner point of the feasible set this usually occurs relatively quickly, but in methods which generate non-corner point solutions, this may only happen after an infinite number of iterations.

This condition is a stopping rule for Steuer's Contracting Cones method, the Interactive Weighted Sums/Filtering approach and the Interactive Weighted Tchebbycheff method.

5.4.2 Stopping When There are No Tradeoffs the DM Likes

In methods where the DM is asked about the desirability of tradeoffs, the algorithm can stop if the DM likes no such tradeoff, since this means that there is no solution which the DM will prefer to the current one. (See Zionts and Wallenius [71])

This rule is used as a stopping condition by the method of Zionts and Wallenius and the Cutting Plane algorithm.

This stopping condition could also be applied to the Visual Interactive Method but this has been omitted in this implementation. (See Equations 6-9 in section 3.3.2)

5.4.3 Stopping When No Increase in Utility is Possible

Recall from section 5.2.3 that we attempted to find the direction of maximum increase in utility by solving

$$\begin{aligned} \max \quad & \alpha \\ \text{subject to} \quad & \alpha \leq (\underline{f}(\underline{x}) - \underline{f}(\underline{x}^{(j)}))^T \underline{\lambda}^{(j)} \quad \forall j = 1 \ldots h \\ & A\underline{x} \leq \underline{b} \\ & \alpha \geq 0 \\ & \underline{x} \geq \underline{0} \end{aligned}$$

If α is zero, then from equation (5.1), $U(\underline{f}(\underline{x}^*)) - U(\underline{f}(\underline{x}^{(h)})) = 0$ so $\underline{x}^{(h)}$ is the point of maximum utility.

This is one of the stopping conditions of the Cutting Plane algorithm.

5.4.4 Stopping After a Fixed Number of Iterations

This criterion is self-explanatory; stop after a predetermined number of iterations of the algorithm have been done. This is a technique generally rejected since the DM is in no position to make such a decision. Consider the absurdity of the following situation: the DM decides on 20 iterations as the stopping criterion. After proceeding through 20 iterations the algorithm stops with an unacceptable solution. The DM now decides that 25 iterations are necessary. The first 20 iterations must be repeated before the DM can continue.

Steuer [56] presents the sample algorithms with this stopping condition. They have been presented here with a different stopping condition.

5.4.5 Stopping When the DM is Satisfied

This criterion is self-explanatory. The DM stops when he or she is satisfied with the current solution. This technique usually leads to the algorithm stopping with a solution the DM is satisfied with, as opposed to the optimal solution, but this should not pose too much of a problem since the DM normally satisfices rather than optimizes (Monarchi *et al.* [37]).

This stopping criterion can be used on its own or in conjunction with one of the other stopping criteria.

It is used by the three sample algorithms and the two projection algorithms.

5.5 Method of Interaction with the DM

As mentioned in the introduction to this chapter, the method of interaction and type of information the DM is expected to provide is extremely important. Shin and Ravindran [49] list eight different interaction types, only five of which occur in the methods under consideration. We consider these individually.

5.5.1 Pairwise comparison

The DM is presented with two criterion vectors and must express which of the two he prefers.

This method is not very demanding. Athan D.W. de Samblancx et al. [46] claim that this style of interaction "minimises the information to be given".

This method of interaction is used by the Zionts and Wallenius and Cutting Plane methods.

5.5.2 Vector comparison

The DM is presented with a set of criterion vectors and must express which he prefers.

This is simply an extension of the previous type of interaction and is also not too demanding on the DM provided that the number of solutions presented is not too large.

This is the method of interaction of the three sample algorithms, viz. Steuer's Contracting Cones method, the Interactive Weighted Sums method and the Interactive Weighted Tchebbycheff method.

5.5.3 Tradeoff evaluation

The DM is presented with a tradeoff vector and asked whether he thinks the tradeoff is favourable, unfavourable or neither.

Zionts and Wallenius [71] report that they found that DMs "prefer choosing between alternatives than evaluating tradeoffs".

The method of Zionts and Wallenius uses this interaction technique.

5.5.4 Aspiration levels

The DM is asked to enter aspiration levels for each of the objectives.

Although this type of interaction does not appear to be particularly demanding on the user, it was found in this study that DMs find it difficult to understand what aspiration levels should be given in order to obtain a desired result. (See Appendix C)

This is the type of interaction required in the Reference Point method, the Visual Interactive Method, and the Interactive Tchebbycheff Method.

5.5.5 Estimation of weights

The DM is asked to estimate his weights at a particular solution.

The Visual Interactive Method requires the DM to enter weights which are then used in the weighted Tchebbycheff norm. This is relatively easy for the DM.

Other methods also require the DM to enter weights, but require these to be estimates of the DM's MRS at a solution. This has been found to be difficult for the DM. Shin and Ravindran [49] report that many researchers believe that vector comparisons are easier to respond to than value tradeoffs.

5.6 Conclusion

We have just discussed the various methods for achieving each of the steps of the basic scheme given at the beginning of the chapter. These different methods could be put together in various combinations to form a number of algorithms. It should be clear though that certain methods for achieving one step are often specifically designed to work with certain methods for achieving another step and the methods are not always interchangeable.

A summary of how each algorithm approaches each of the four steps in the basic scheme, as well as the type of interaction, is given in table 5.1.

5.6 Conclusion

We have just discussed the various methods for achieving each of the steps of the basic scheme given at the beginning of the chapter. These different methods could be put together in various combinations to form a number of algorithms. It should be clear though that certain methods for achieving one step are often specifically designed to work with certain methods for achieving another step and the methods are not always interchangeable.

A summary of how each algorithm approaches each of the four steps in the basic scheme, as well as the type of interaction, is given in table 5.1.

Table 5.1: Algorithm Breakdown

Algorithm	Generate Weights	Generate Solutions	Reduction Technique	Stopping Criterion	Interaction Type
Steuer's Contracting Cones	DM chooses one out of a group	Reduced Criterion Cone LP	Criterion Cone Reduction	Stop when only one solution left	Vector Comparison
Interactive Weighted Sums	DM chooses one out of a group	Weighted Sums LP	Weighting Space Reduction	Stop when only one solution left	Vector Comparison
Interactive Weighted Tchebbycheff	Definition Point of Tchebbycheff	Reference Point Projection	Weighting Space Reduction	Stop when only one solution left	Vector Comparison
Zionts and Wallenius	Constrained by tradeoffs	Weighted Sums LP	Weighting Space Reduction	Stop when DM likes no tradeoff	Pairwise Comparison and Tradeoff Evaluation
Cutting Plane	Constrained by tradeoffs	Direction of local increase in utility function	Variable Space Reduction	Stop when DM likes no tradeoffs or no utility increase possible	Pairwise Comparison
Reference Point	From Reference Point	Reference Point Projection	No Reduction	Stop when DM satisfied	Aspiration Levels
Visual Interactive Method	DM supplied	Line Segment Projection	No Reduction	Stop when DM Satisfied	Aspiration Levels, Weights and Vector Comparison

Chapter 6

Algorithm Comparison and Evaluation

We now attempt to compare the algorithms with a view to identifying the most appropriate to apply. We first compare the algorithms analytically, then computationally, and finally present some results regarding decision makers' opinions of the algorithms.

Note 6.1 For simplicity, the names of the algorithms are abbreviated as follows in all the results presented.

Steuer's Contracting Cones	-	SCC
Interactive Weighted Sums	-	IWS
Tchebbycheff Method	-	TCH
Method of Zionts and Wallenius	-	ZW
Cutting Plane Method	-	CUT
Reference Point Method	-	REF
Visual Interactive Method	-	VIM

6.1 Analytic Comparison

In order to contrast the different features of the algorithms, we consider the following questions, some of which were obtained from Steuer [56] and Cohon and Marks [10].

1. Does the algorithm require the DM to be mathematically skilled or computer sophisticated ?
2. Is it enjoyable for the DM to interface with the algorithm[1] ?
3. Does the algorithm present one, two, or a group of solutions at each iteration ?
4. Does the algorithm ask the same pattern of questions at each iteration, or are different types of questions asked on different iterations ?
5. What is the DM required to do at each interaction ?
6. Does the algorithm provide sufficient information to the DM so that an informed decision can be made ?
7. Does the algorithm enable the DM to rapidly appreciate how the objectives tradeoff against one another ?
8. Is the algorithm an *ad hoc* procedure ? Steuer [56] defines a procedure as being ad hoc if the user would not know how to respond deterministically to the questions posed by the algorithm, even if the DM's utility function were known mathematically.
9. Does the algorithm allow the DM to change his aspirations during the solution process without necessitating an algorithm restart ?
10. Does the algorithm make specific assumptions about the shape of the DM's utility function ?
11. Does the algorithm have convergence proofs or only intuitive convergence properties ?

[1] The answer to question 2 was obtained from the experiment. See p. 188

12. Does the algorithm converge in a specified number of iterations ?

13. Does the algorithm generate dominated criterion vectors ?

14. Will the algorithm converge to the optimal solution if it lies on a corner point of the feasible set, assuming the DM does not give incorrect preference information ?

15. Will the algorithm converge to the optimal solution if it lies on a facet of the feasible set, assuming the DM does not give incorrect preference information ?

16. Will the algorithm converge to the optimal solution if it lies in the interior of the feasible set, assuming the DM does not give incorrect preference information ?

17. Is the algorithm consistent i.e. will the algorithm do exactly the same thing if it is run more than once with the same input, parameters and decision maker's input ?

18. Is the technique computationally efficient ?

19. Does the algorithm require a large amount of CPU time ?

20. Does the algorithm require specialized, as opposed to conventionally available, software ?

The answers to these questions, for each algorithm, are presented in Table 6.1.

Table 6.1: How the Algorithms Match Up

	Steiner's Contracting Cones	Interactive Weighted Sums	Tchebbycheff Algorithm	Zionts and Wallenius	Cutting Plane	Reference Point	Visual Interactive Method
1	No	No	No	No	No	No	No
2	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Group	Group	Group	Two	Two	One	Group
4	Same	Same	Same	Same	Same	Same	Same
5	See previous section						
6	Yes	Yes	Yes	Yes	Yes	No	No
7	Yes	Yes	Yes	Yes	Yes	No	Yes
8	No	No	No	No	No	Yes	Yes
9	No	Yes, if minor	Yes, if minor	Yes	No	Yes	Yes
10	No	No	No	Yes	Yes	No	Yes
11	No Proofs	No Proofs	No Proofs	Proofs	Proofs	No Proofs	Proofs
12	No	No	No	No	No	No	No
13	No	No	No	No	No	No	No
14	Possibly	Possibly	Possibly	Yes	Yes	Yes	Yes
15	No	No	Possibly	No	Yes	Yes	Yes
16	No	No	Possibly	No	Yes	Yes	Yes
17	No	No	No	Yes	Yes	Yes	Yes
18	Yes	No	No	Yes	Yes	Yes	Yes
19	Medium	Yes	Yes	No	No	No	Medium
20	No	No	No	No	No	No	No

6.2 Computational Comparison

We now consider the performance of the algorithms when the DM's utility function is known, and perfect preference information is supplied to the algorithms. Only the algorithms which are not ad-hoc procedures can be tested in this way since knowing the DM's utility function does not help in reaching the optimal solution in an ad hoc procedure.

We consider the algorithms' performance on

1. A small problem whose optimal solution lies on a corner point of the feasible set
2. A small problem whose optimal solution lies on a facet of the feasible set
3. A small problem whose optimal solution lies in the interior of the feasible set
4. A larger problem whose optimal solution lies on a facet of the feasible set

Note 6.2 The sample algorithms were run with a reduction factor of 0.8 and 6 solutions were presented to the DM at each iteration.

Example 6.3 Consider the problem given by Zionts and Wallenius [69].

$$\begin{aligned} \max \quad & f_1 = 3x_1 + x_2 + 2x_3 + x_4 \\ \max \quad & f_2 = x_1 - x_2 + 2x_3 + 4x_4 \\ \max \quad & f_3 = -x_1 + 5x_2 + x_3 + 2x_4 \end{aligned}$$

$$\begin{aligned} \text{subject to} \quad & 2x_1 + x_2 + 4x_3 + 3x_4 \leq 60 \\ & 3x_1 + 4x_2 + x_3 + 2x_4 \leq 60 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

The utility function is assumed to be

$$U = 0.58f_1 + 0.21f_2 + 0.21f_3$$

The optimal solution is $\underline{x} = (12, 0, 0, 12), \underline{z} = (48, 60, 12), U^* = 42.96$.

The utility function value of each iterate is given in table 6.2.

Both the method of Interactive Weighted Sums and Steuer's Contracting Cones generated the optimal solution in the first iteration but continued to iterate until this was the only one which had not been 'cut out'. Presumably the DM would have stopped once he realised that no better solutions were being generated. Note that the method of Interactive Weighted Sums stopped sooner than Steuer's Contracting Cones. The Tchebbychieff Method converged to a nonoptimal solution suggesting that the optimal solution was cut out in one of the early iterations. This could be remedied by running with a larger reduction factor but this would cause the program to iterate for longer. The Cutting Plane Method and Method of Zionts and Wallenius converged on the optimal solution after only one iteration.

Example 6.4 Consider the problem given by Zionts and Wallenius [71].

$$\begin{array}{lll} \max & f_1 = x_1 & \\ \max & f_2 = & x_2 \\ \max & f_3 = & x_3 \end{array}$$

$$\begin{array}{ll} \text{subject to} & 3x_1 + 2x_2 + 3x_3 \leq 18 \\ & x_1 + 2x_2 + x_3 \leq 10 \\ & 9x_1 + 20x_2 + 7x_3 \leq 96 \\ & 7x_1 + 20x_2 + 9x_3 \leq 96 \\ & x_1, x_2, x_3 \geq 0 \end{array}$$

The constraint set is as shown in figure 6.1.

The utility function is given by

$$U = \min\{3f_1, 5f_2, 3f_3\}$$

and with this utility function the optimal solution is $\underline{x}^* = (2.5, 1.5, 2.5)$ and $U^* = 7.5$. The extreme point solution of maximum utility is $\underline{x}^{ext} = (1, 4, 1)$ and $U^{ext} = 6$.

Table 6.2: Utility Function Value of Each Iterate for Example 6.3

	SCC	IWS	TCH	ZW	OUT
0				41.64	41.64
1	42.96	42.96	42.7697	42.96	42.96
2	42.96	42.96	42.6709		
3	42.96	42.96	42.7510		
4	42.96		42.7663		
5	42.96		42.7712		
6	42.96		42.7726		
7	42.96		42.7731		
8	42.96		42.7732		
9	42.96		42.7733		
10	42.96		42.7733		
11	42.96		42.7733		
12	42.96		42.7733		
13	42.96		42.7733		
14			42.7733		
15			42.7733		
16			42.7733		
17			42.7733		
18			42.7733		
19			42.7733		

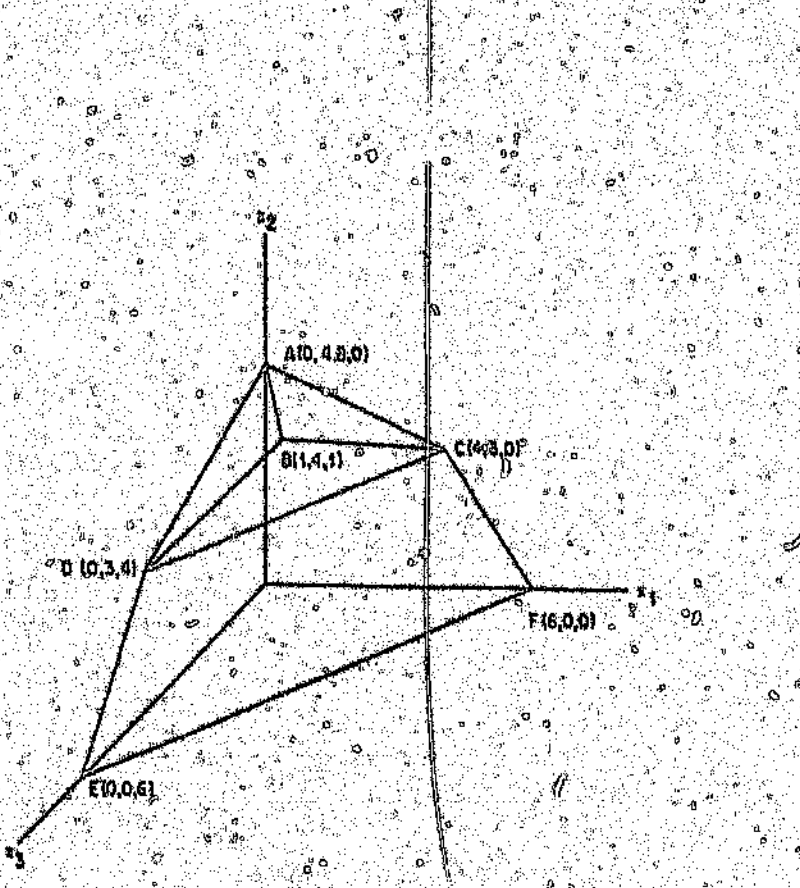


Figure 6.1: Constraint Set for Example 6.4

The utility function given above is concave but does not have continuous derivatives. The function

$$U = ((25 - 3f_1)^k + (25 - 5f_2)^k + (25 - 3f_3)^k)^{\frac{1}{k}}$$

with k sufficiently large is an almost concave utility function with continuous derivatives. Using this utility function with $k = 30$, $U^* = -18.4527$ and $U^{ex} = -22.5142$.

The utility function value of each iterate is given in table 6.3.

As expected, Steuer's Contracting Cones, the method of Interactive Weighted Sums and the method of Zionts and Wallenius did not converge to the optimal solution, but only the method of Zionts and Wallenius converged to the extreme point of maximum utility. Again, the method of Interactive Weighted Sums stopped sooner than Steuer's Contracting Cones. The Tchebbycheff method converged to a "near optimal" solution with utility higher than the best extreme point. The Cutting Plane method converged to a solution with higher utility than $\underline{x}^*$. (Recall that $\underline{x}^*$ is optimal for $U = \min\{3x_1, 5x_2, 3x_3\}$)

Table 6.3: Utility Function Value of Each Iterate for Example 1.4

	SCC	IWS	TCH	ZW	CUT
0				-25.0000	-25.0000
1	-25	-25	-25.0000	-22.5142	-25.0000
2	-25	-25	-18.8714		-25.0374
3	-25	-25	-18.3547		-21.1942
4	-25		-18.3051		-20.4200
5	-25		-18.2770		-20.4536
6			-18.2692		-19.6484
7			-18.2665		-18.6921
8			-18.2655		-18.8092
9			-18.2652		-18.1533
10			-18.2651		-18.2295
11			-18.2651		-18.1405
12			-18.2651		-18.1206
13			-18.2651		-18.1188
14			-18.2651		-18.1097
15			-18.2651		
16			-18.2651		
17			-18.2651		
18			-18.2651		
19			-18.2651		

Example 6.5 Consider the problem given by Benvenuti [2].

$$\begin{aligned} \max \quad & f_1 = x_1 \\ \max \quad & f_2 = x_2 \\ \max \quad & f_3 = -x_1 - x_2 \end{aligned}$$

$$\begin{aligned} \text{subject to} \quad & x_1 + 2x_2 \leq 4 \\ & -x_1 + x_2 \leq 1 \\ & x_1 - 2x_2 \leq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

The constraint set is as shown in figure 6.2.

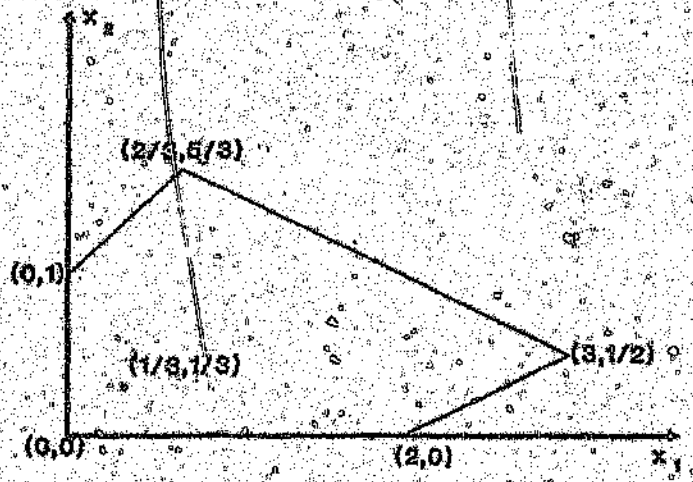


Figure 6.2: Constraint Set for Example 6.5

The utility function is given by

$$U = -((2f_1 - 1)^2 + (f_2 - 1)^2 + (f_3)^2)$$

The optimal solution is $\underline{x}^* = (\frac{1}{3}, \frac{1}{3})$, $\underline{z}^* = (\frac{1}{3}, \frac{1}{3}, -\frac{2}{3})$ and $U^* = -1$.

The utility function value of each iterate is given in table 6.4.

Once again, the method of Zionts and Wallenius, Steuer's Contracting Cones and the method of Interactive Weighted Sums converged on the extreme point

Table 6.4: Utility Function Value of Each Iterate for Example 6.5

	SCC	IWS	TCH	ZW	OUT
0				-2	-2.00000
1	-2	-2	-1.5227	-2	-37.50000
2	-2	-2	-1.02721		-8.76543
3			-1.00465		-10.12720
4			-1.00591		-2.00000
5			-1.00470		-1.03654
6			-1.00403		-1.62241
7			-1.00390		-1.15556
8			-1.00386		-1.08006
9			-1.00385		-1.00898
10			-1.00385		-1.00892
11			-1.00385		-1.00793
12			-1.00385		-1.00061
13			-1.00385		-1.00102
14			-1.00385		-1.00424
15			-1.00385		-1.00020
16			-1.00385		-1.00342
17			-1.00385		-1.00150
18			-1.00385		-1.00021
19			-1.00385		-1.00047
20					-1.00002
21					-1.00002
22					-1.00003
23					-1.00000

of maximum utility. As before, the Tchebbycheff method converged to a near optimal solution and the Cutting Plane algorithm converged to the optimal solution.

Example 6.6 Consider the problem given by Steuer *et al.* [54]

$$
\begin{array}{lll}
\max & f_1 = 4x_1 - x_2 + 3x_3 + 2x_5 - x_8 \\
\max & f_2 = 3x_2 - x_3 + 3x_5 + 4x_6 - x_7 \\
\max & f_3 = 2x_2 - 3x_3 + 4x_4 + 4x_5 + x_6 + 2x_7 + 2x_8
\end{array}
$$

$$
\begin{array}{lll}
\text{subject to} & x_1 + 4x_2 + 3x_4 + 5x_5 + 2x_7 + 8x_8 \leq 92 \\
& 5x_2 \leq 95 \\
& 5x_3 \leq 91 \\
& x_2 + 7x_4 + 8x_6 \leq 94 \\
& 3x_1 + 5x_2 + 7x_6 + 3x_7 \leq 94 \\
& 8x_5 + 4x_6 + 7x_8 \leq 73 \\
& 6x_1 + 8x_2 + x_3 + 2x_4 + x_5 + 2x_6 + 6x_7 + \leq 93 \\
& 4x_1 + 2x_2 + 2x_3 + 2x_4 + 5x_5 + 5x_6 + 2x_8 \leq 99 \\
& x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8 \geq 0
\end{array}
$$

Steuer *et al.* [54] claim that the problem has 77 efficient extreme points. Let us assume that the DM's utility function is given by

$$U = \prod_{i=1}^{3} \left[\frac{f_i + 15}{10} \right]$$

With this U, the DM's extreme point of greatest utility is given by

$$x^{ext} = \begin{bmatrix} 0.3125 \\ 0.0000 \\ 11.3750 \\ 0.0000 \\ 3.2500 \\ 11.7500 \\ 1.3515 \\ 0.0000 \end{bmatrix} \quad z^{ext} = \begin{bmatrix} 41.8750 \\ 44.0234 \\ 61.5781 \\ 51.5000 \end{bmatrix} \quad \text{and } U^{ext} = 1709.497$$

and the optimal solution is given by

$$x^* = \begin{bmatrix} 3.6197 \\ 3.8800 \\ 11.3750 \\ 0.0000 \\ 3.8041 \\ 6.9980 \\ 1.8443 \\ 0.0000 \end{bmatrix} \quad z^* = \begin{bmatrix} 52.3322 \\ 37.8254 \\ 67.7885 \\ 46.2751 \end{bmatrix} \quad \text{and } U^* = 1804.348$$

The utility function value of each iterate is given in table 6.5.

This time, neither Steuer's Contracting Cones nor the method of Interactive Weighted Sums converged to the best extreme point, while the method of Zionts and Wallenius converged on it remarkably quickly. The Tchebbycheff method again converged to a solution which is not optimal but has a substantially higher utility than the extreme point of maximum utility. The Cutting Plane method converged to a near optimal solution.

6.2.1 Interpretation of Results

The examples have shown that the methods of Steuer's Contracting Cones and Interactive Weighted Sums are not particularly good at finding the optimal extreme point. The Tchebbycheff method finds better solutions but is extremely slow. The method of Zionts and Wallenius is a very fast and efficient way of finding the optimal extreme point. The Cutting Plane method converges to the optimal solution, first very quickly and then more slowly as we home in on the solution.

Table 6.5: Utility Function Value of Each Iterate for Example 6.6

	SCC	IWS	TCH	ZW	CUT
0				900.76	900.76
1	1017.75	1523.74	1304.46	1652.25	1065.02
2	1403.50	1523.74	1752.11	1709.51	1330.05
3	1523.74	1593.34	1772.30		1536.85
4	1528.74	1593.34	1784.97		1683.12
5	1523.74		1788.78		1350.18
6	1523.74		1789.89		1715.76
7			1790.11		1794.21
8			1790.21		1704.63
9			1790.23		1747.90
10			1790.24		1584.92
11			1790.25		1771.18
12			1790.25		1791.66
13			1790.25		1754.12
14			1790.25		1750.11
15			1790.25		1800.73
16			1790.25		1800.49
17			1790.25		1775.22
18			1790.25		1788.51
19			1790.25		1798.39
20					1798.38
21					1802.33
22					1802.69
23					1803.98
24					1802.84
25					1803.89
26					1804.23
27					1803.29
28					1803.44
29					1800.50
30					1802.95

6.3 An Evaluation of the Algorithms from a Decision Maker's Perspective

We now wish to determine the DMs' opinions of the various algorithms, specifically to determine whether the DMs are comfortable using the algorithms. The DM's comfort is after all a major factor in judging the success of an algorithm as a decision making tool. No matter how quickly the algorithm converges, and no matter how "good" the final solution, the method is unlikely to be implemented in practice if the DM is averse to using the algorithm.

Hence, an experiment to test the DM's perception of the algorithms was designed. The experiment was specifically intended to establish

1. How useful the DM found the method.
2. How satisfied the DM was with the final solution.
3. How easy the DM found it to use the algorithm.
4. How easy the DM found it to understand the logic of the algorithm.
5. How quickly the DM reached his final solution.
6. Whether the DM thought the time taken to reach the final solution was justified by the quality of that solution.
7. If the DM found it enjoyable to interact with the algorithm.
8. What the DM liked and disliked about each algorithm.
9. Which algorithm the DM liked most and why.
10. Which algorithm the DM liked least and why.
11. Which factors the DM considers important in judging an algorithm.

The ultimate aim in testing the algorithms on users is to clarify whether a procedure is acceptable to users, what kind of difficulties they face when using the method, and whether suggestions for improvement can be deduced from this.

6.3.1 Description of Experiment

The experiment was conducted using 23 undergraduate students acting as decision makers. The students had no experience with MOLP but were familiar with linear programming and the computer.

The seven algorithms were programmed on an IBM 370 mainframe in VS PASCAL. A REXX program was written to call the seven algorithms and to record the start and end times of the algorithms. The seven methods were presented to the DMs in a random order to prevent bias. When an algorithm terminated, the DM was asked if he wanted to use the same algorithm again. This was so that the DMs could vary their input parameters if they felt it was necessary. The algorithms were given fictitious names to prevent the DMs from being influenced by any prior information they may have had. The DMs completed a questionnaire after each algorithm had terminated and an algorithm rating after all seven algorithms had been tested. Copies of these questionnaires can be found in Appendices A and B respectively.

The problem used for the testing was the following nutrition problem from Masud and Hwang [36].

The problem is to determine the quantities of food that should be eaten in order to meet certain nutritional requirements so as to

1. minimize cost
2. minimize cholesterol intake
3. minimize carbohydrate intake

The problem constraints are to meet the minimum daily nutritional requirements and the upper limits on daily intake of individual foods. Information on the six foods is given in Table 6.6.

Converting to standard MOLP form, the problem becomes

Table 6.6: Nutrition and Cost Information on Six Foods

	Milk (pint)	Beef (pound)	Eggs (dozen)	Bread (ounce)	Lettuce and salad (ounce)	Orange juice (pint)	RDA for adults
Vitamin A (i.u.)	720	107	7080	0	134	1000	5000
Food energy (calories)	344	1460	1040	75	17.4	240	2500
Cholesterol (unit)	10	20	120	0	0	0	
Protein (g)	18	151	78	2.5	0.2	4	63
Carbohydrate (g)	24	27	0	15	1.1	52	
Iron (mg)	0.2	10.1	13.2	0.75	0.15	1.2	12.5
Cost (R)	0.75	7.34	2.67	0.33	0.17	0.87	
Maximum daily intake	6	1	0.25	10	10	4	

$$\min \quad f_1 = 0.225x_1 + 2.2x_2 + 0.8x_3 + 0.1x_4 + 0.05x_5 + 0.26x_6$$
$$\min \quad f_2 = 10x_1 + 20x_2 + 120x_3$$
$$\min \quad f_3 = 24x_1 + 27x_2 + 15x_4 + 1.1x_5 + 52x_6$$

$$\text{subject to} \quad 720x_1 + 107x_2 + 7080x_3 + 134x_5 + 1000x_6 \geq 5000$$
$$0.2x_1 + 10.1x_2 + 13.2x_3 + 0.75x_4 + 0.15x_5 + 1.2x_6 \geq 12.5$$
$$344x_1 + 1460x_2 + 1040x_3 + 75x_4 + 17.4x_5 + 240x_6 \geq 2500$$
$$18x_1 + 151x_2 + 78x_3 + 2.5x_4 + 0.2x_5 + 4x_6 \geq 63$$
$$x_1 \leq 6.0$$
$$x_2 \leq 1.0$$
$$x_3 \leq 0.25$$
$$x_4 \leq 10.0$$
$$x_5 \leq 10.0$$
$$x_6 \leq 4.0$$
$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$$

It was felt that the DM's would have difficulty in relating to the problem, i.e. they would find it difficult to trade off units of cholesterol and grams of carbohydrate because the values were meaningless to them. To overcome this problem, the DMs were given (fictitious) information as to how the levels of cholesterol and carbohydrate would affect their chances of having a heart attack and putting on weight respectively. These were two quantities the DMs had no difficulty in relating to.

The students were not graded on their efforts since Brockhoff [5] claims that DMs in an experiment should not have had problem related theoretical train-

ing, and their evaluations and performances should not be influenced by expectations on future grading.

Note 6.3 It must be stressed that the Visual Interactive Method was unfairly judged in this experiment. The computer terminals the experiment was conducted on did not support the use of graphics and hence the DM was presented with a table of criterion vectors to select from, instead of being shown a graph of the changing criterion values. In addition, the stopping criterion was altered (see section 3.3.2) thus placing more of a burden on the DM. In spite of this, some of the results obtained may still be valid.

6.3.2 Results of Experiment

Although the sample of only 23 students seems rather small, Dyer [11] has presented results from only 9 subjects, and de Samblancx et al. [46] used only 17 subjects.

The results presented are divided into three groups. The results in group A are those which are simply factual and do not represent the opinions of the DMs. They are presented purely for the sake of interest sake. The results in group B are those which represent the DMs' opinions of the algorithms and were obtained (using the questionnaire in Appendix A) from the DM after each algorithm had terminated. The results in group C also represent the DMs' opinions of the algorithms but were obtained (using the questionnaire in Appendix B) only after the DM had used all seven algorithms. In other words, the results in group B are the DMs' opinions of an algorithm considered in isolation, while those in group C are their opinions of an algorithm compared to all the others.

Analysis of Variance (ANOVA), together with the LSD T-test, is used to test whether the differences in scores between the various algorithms are significant, or simply due to random factors. Although this test assumes the data is distributed normally, it can still be used if the data is not normally distibuted on condition that that the data is perfectly balanced (Scheffé [47] p. 337). This

applies in our case since each DM tested each algorithm, so the ANOVA test can be used.

Group A Results

Number of solutions chosen to compare at a time (Only SCC, IWS and TCH)

min = 3
max = 9
mean = 5.797

Reduction factor chosen (Only IWS and TCH)

min = 0.2
max = 0.6
mean = 0.389

Average time taken to reach final solution

We would expect the algorithm completion times to be normally distributed but the actual completion time distribution, as shown in Figure 6.3, resembles a lognormal distribution.

The mean execution times (in minutes and seconds) for each algorithm as well as significant differences are presented below. Algorithms with the same letter are not significantly different at the 5% level.

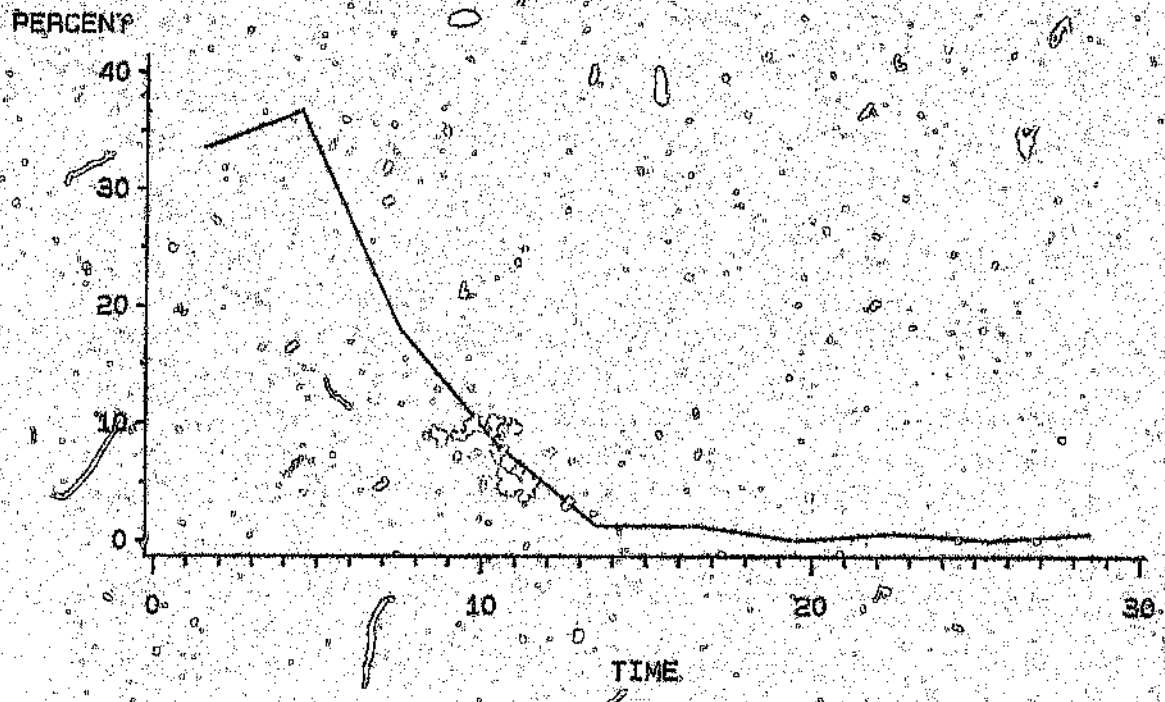


Figure 6.3: Distribution of Algorithm Completion Times

Method	ZW	SCC	REF	CUT	VIM	TCH	IWS	Average
Mean	2:59.2	3:10.1	4:43.5	4:44.0	6:25.6	6:29.0	6:50.1	5:03.1
Significant Differences	A	A	A B	A B	B	B	B	

In other words, ZW and SCC take less time than VIM, TCH and IWS. Other differences in mean execution time are not significant.

Group B Results

We now consider

- how useful the DMs thought each algorithm was
- how satisfied the DMs were with the final solution provided by the algorithm

- how easy the DMs found it to use the methods
- how easy the DMs found it to understand the methods
- how quickly they reached their final solution

The DMs rated each algorithm's performance on a scale of 1 to 10 and we would expect these scores to be normally distributed. The actual frequency distribution of the scores is shown in Figure 6.4. It is interesting to note that

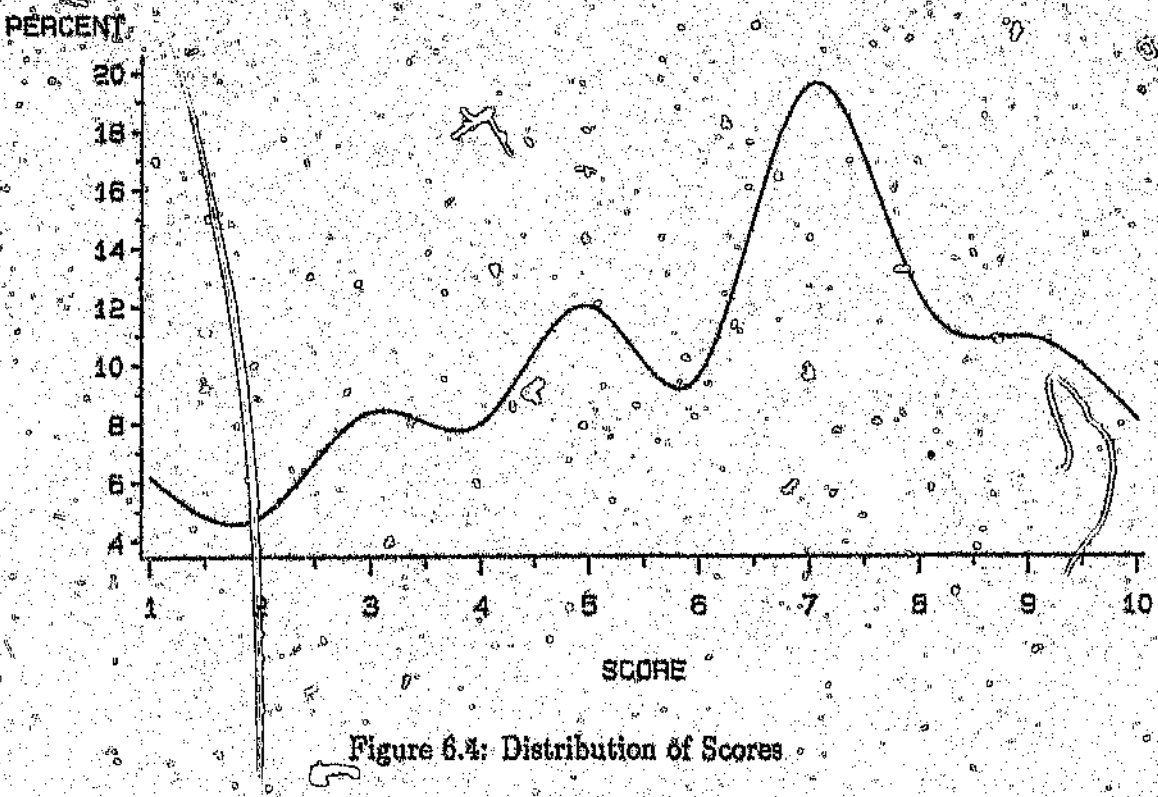


Figure 6.4: Distribution of Scores

odd scores were given far more often than even scores. Assuming DMs are biased against even scores, we consider the distribution of the scores when 1 and 2 are in the same class, 3 and 4 are in the same class, and so on. This distribution is shown in Figure 6.5.

The average scores obtained by each algorithm for the various criteria are presented in the tables below. The algorithms are listed in order of performance with the algorithm which performed best on the left. Algorithms with the same letter are not significantly different in that criterion at the 5% level.

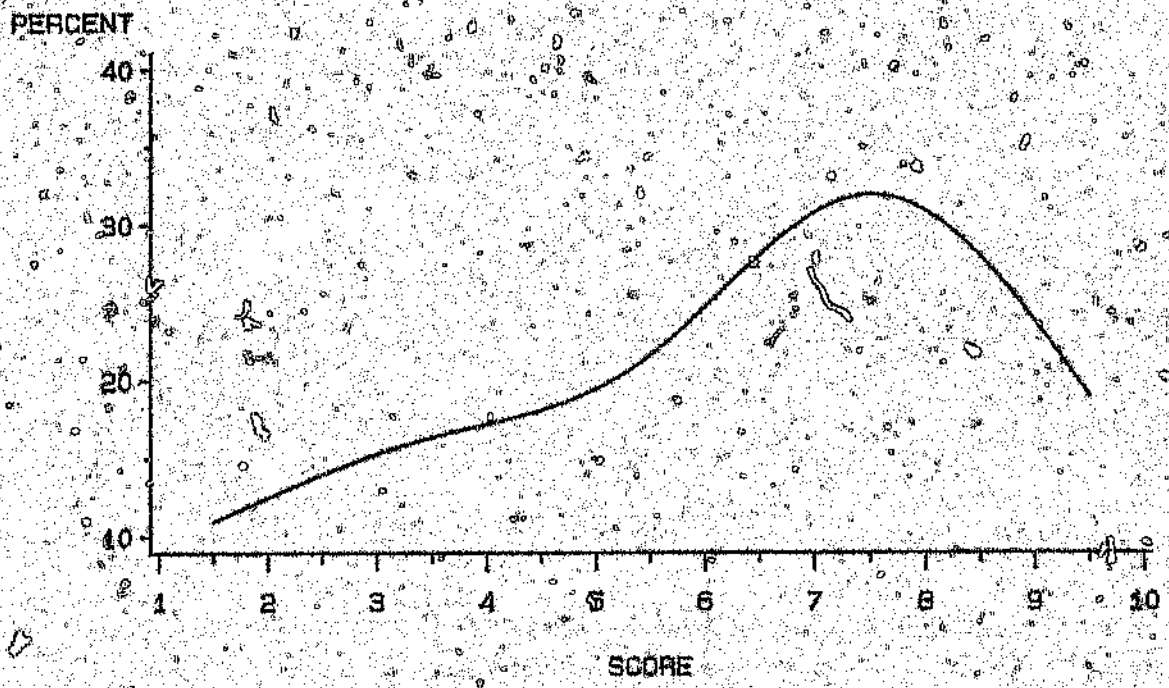


Figure 6.5: Distribution of Scores with Odd and Even Scores Combined

DMs' perceived usefulness of method (1=useless, 10=extremely useful)

Method	CUT	IWS	ZW	TCH	SCC	REF	VIM	Average
Mean	6.3043	5.9565	5.4783	4.9565	4.7826	4.5652	4.5217	5.2236
Significant Differences	A	A B	A B C	B C	B C	C	C	

In other words, DMs' found CUT to be more useful then TCH, SCC, REF and VIM, while they found IWS to be more useful than REF and VIM. The other differences in scale are not significant.

The relatively low scores for usefulness of method seem to suggest suspicion or unease on the part of DMs towards decision making techniques.

DMs' satisfaction with final solution (1=completely unsatisfied, 10=totally satisfied)

Method	CUT	IWS	ZW	TCH	REF	SCC	VIM	Average
Mean	6.7826	6.5217	5.9130	5.0870	4.9565	4.1304	4.0870	5.3454
Significant Differences	A	A B	A B C	B C D	C D	D	D	

This result is extremely interesting and rather unexpected. Only one of the four of the methods which can converge to nonextreme solutions is found in the best group, while two out of the three methods which only find the optimal extreme point are in this group. This would seem to suggest that an algorithm's inability to generate nonextreme solutions may not be as serious as thought. It is of course possible that the structure of the problem caused the DMs' optimal solutions to be on or "close" to an extreme point in all cases. Extensive experimentation would be necessary to determine whether this unexpected result holds in general.

DMs' perceived ease of understanding (1=very difficult, 10=very easy)

Method	ZW	CUT	IWS	SCC	REF	TCH	VIM	Average
Mean	7.2609	5.9565	5.4783	4.9565	4.7826	4.5652	4.5217	6.3664
Significant Differences	A	A	A	A	A	A	A	

The ANOVA test indicates that there is no significant difference between methods at the 5% level in terms of ease of understanding.

DMs' perceived ease of use (1=very difficult, 10=very easy)

Method	CUT	ZW	IWS	SCC	TCH	VIM	REF	Average
Mean	8.2609	7.9130	7.5217	7.4783	7.0870	6.8261	6.3478	7.3478
Significant Differences	A	A B	A B C	A B C	A B C	B C	C	

The DMs found all algorithms easy to use. CUT was easier to use than VIM and REF, and ZW was easier to use than REF, but otherwise all alorithms are approximately equally easy to use. The high scores given for ease of use are encouraging.

DMs' perceived speed of reaching final solution (1=very slow, 10=very fast)

Method	ZW	CUT	SCC	VIM	REF	IWS	TCH	Average
Mean	7.0870	7.0435	6.6522	6.0870	6.0434	4.6087	4.1348	5.9939
Significant Differences	A	A	A	A	A B	B C	C	

It is interesting to note that the ordering of algorithms in terms of time to reach the final solution does not correspond entirely with the above table. DMs perceived CUT to be faster than it actually was, and perceived TCH to be slower than IWS, when in fact it was faster. A possible explanation for this is that the DMs' perception of speed tends to be judged by the time taken between iterations rather than the overall solution time.

We now consider the replies to four yes/no/don't know questions. The algorithms are again presented in order of performance with the algorithm which performed best on the left.

Was time taken justified by quality of solution ?

Method	CUT	ZW	REF	IWS	SCC	VIM	TCH
Yes	73.9%	56.5%	43.5%	39.1%	39.1%	34.8%	30.4%
No	13.0%	26.1%	43.5%	34.8%	47.8%	52.2%	52.2%
Don't know	13.0%	17.4%	13.0%	26.1%	13.1%	13.0%	17.4%

Is it necessary to rerun the algorithm in order to get a solution you are satisfied with ?

Method	IWS	CUT	ZW	TCH	VIM	SCC	REF
Yes	26.1%	30.4%	39.1%	43.5%	43.5%	47.8%	60.9%
No	60.9%	43.5%	43.5%	39.1%	30.4%	34.8%	30.4%
Don't know	13.0%	26.1%	17.4%	17.4%	26.1%	17.4%	8.7%

Is it enjoyable to interact with the algorithm ?

Method	ZW	CUT	IWS	SCC	REF	TCH	VIM
Yes	78.3%	78.3%	69.6%	69.6%	65.2%	56.5%	47.8%
No	13.0%	17.4%	13.0%	21.7%	26.1%	34.8%	39.1%
Don't know	8.7%	4.3%	17.4%	8.7%	8.7%	8.7%	13.0%

Would the DM use this algorithm to solve a decision making problem ?

Method	CUT	ZW	IWS	SCC	VIM	REF	SA
Yes	52.2%	52.2%	34.8%	34.8%	30.4%	21.7%	47.8%
No	13.0%	26.1%	21.7%	43.5%	52.2%	52.2%	17.3%
Don't know	34.8%	21.7%	43.5%	21.7%	17.4%	26.1%	34.8%

This result again suggests uneasiness with formal decision making techniques. In only two of the algorithms, would more than half the DMs use the algorithm to make a decision.

The DMs were asked what they liked and disliked about each algorithm. A brief mention of these is given for each algorithm. A few representative quotes to illustrate the DM's likes and dislikes are listed in Appendix C.

Steuer's Contracting Cone

In general, DMs found this method very fast but were not satisfied with the final solution presented to them. They liked the method of interaction.

Interactive Weighted Sums

DMs were relatively pleased with the solutions generated by this method but found it a little slow. They liked the method of interaction.

Tchebbycheff Method

The DMs were unanimous in the opinion that the algorithm was too slow. Reaction to the solutions produced were mixed.

The Method of Zionts and Wallenius

Some DMs liked only having two solutions to compare, others did not. Con-

trary to popular belief, many DMs liked evaluating tradeoffs. DMs were a little distressed by the jumping around of the solutions.

Cutting Plane Method

Most DMs liked the solutions produced by this algorithm, but a few found the interaction tedious and boring.

Reference Point Method

Most DMs liked the idea of entering a reference point but found it difficult to ascertain which reference point values would drive the solution in the direction they desired.

Visual Interactive Method

As with the Reference Point method, most DMs liked the idea of entering a reference point and weights, but found it difficult to ascertain which values would drive the solution in the direction they desired.

Group C Results

Overall Rank

The DMs were asked to rank the seven algorithms from 1 to 7 where 1 is the algorithm they liked best and 7 is the algorithm they liked least. The resulting mean ranks are presented below. Algorithms with the same letter are not significantly different at the 5% level.

Method	CUT	ZW	IWS	SCG	VIM	REF	TCH
Mean	2.8095	3.2381	3.4762	4.5238	4.5714	4.5714	4.8095
Significant Differences	A	A	A B	B C	B C	B C	C

This result indicates that the DMs prefer CUT and ZW to all other algorithms otherthan IWS. Note that CUT and ZW have been in the most preferred group in every significance test, while IWS was in the most preferred group in every significance test with the exception of those relating to speed.

Importance of factors for judging the performance of an algorithm

When the DMs were asked to rate the importance of four factors in judging the performance of an algorithm on a scale of 1 to 10, the following results were obtained.

Factor	Quality of Solution	Ease of Understanding	Ease of Use	Speed
Mean	9.5652	7.5217	7.4783	6.8696
Significant Differences	A	B	B	B

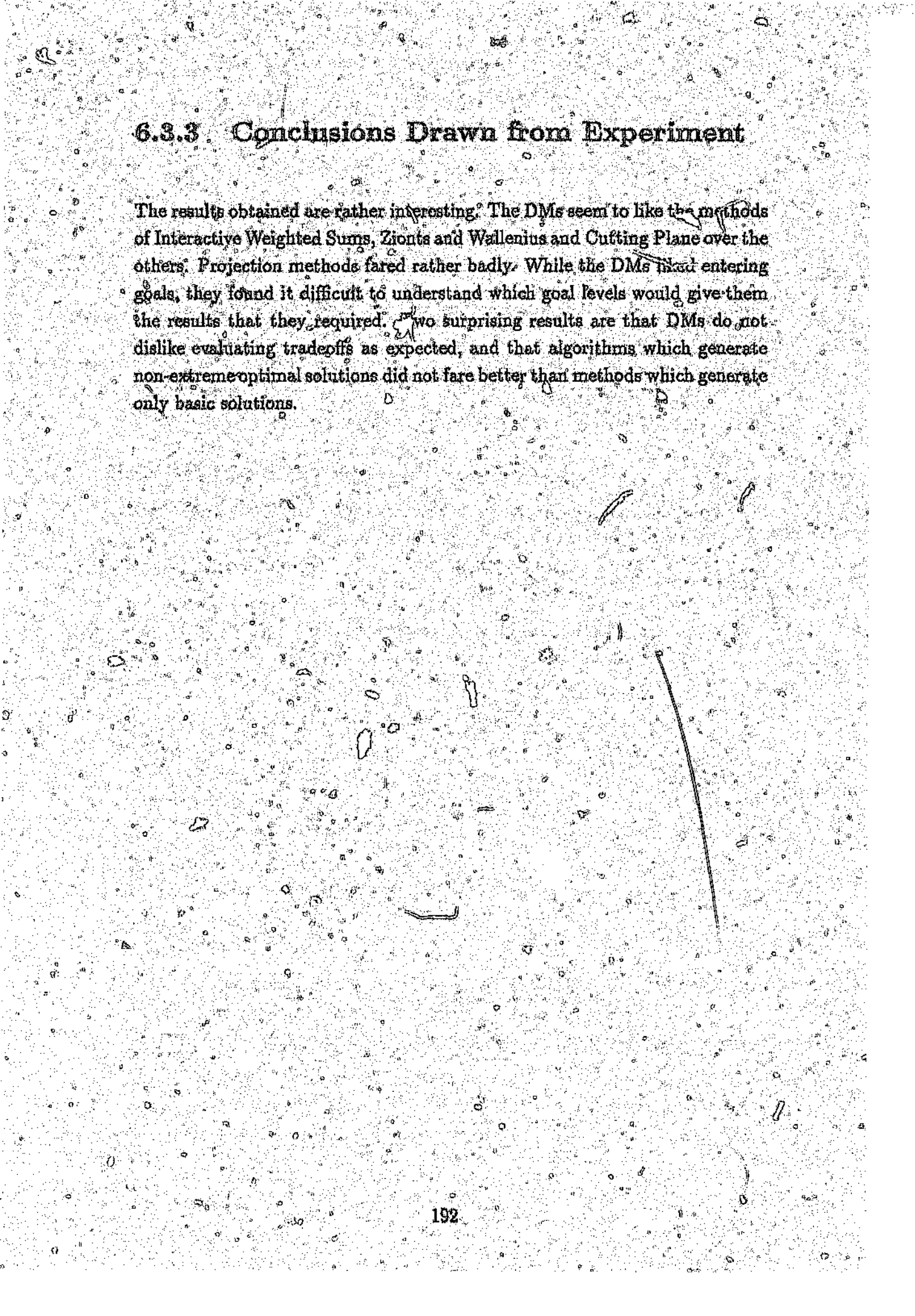
DMs were also asked why they liked the algorithm they ranked 1 and disliked the algorithm they ranked 7. The only interesting result of this question is that the most common reason for disliking an algorithm is that it was too slow, and yet speed received the lowest score of the four factors given above.

The DMs were also asked which algorithm(s) they felt had the best way of extracting information from the user. The number of votes each received was

Method	VIM	ZW	CUT	SCO	IWS	REF	TCH
Votes	8	7	6	6	4	4	4

6.3.3 Conclusions Drawn from Experiment

The results obtained are rather interesting. The DMs seem to like the methods of Interactive Weighted Sums, Zionts and Wallenius and Cutting Plane over the others. Projection methods fared rather badly. While the DMs liked entering goals, they found it difficult to understand which goal levels would give them the results that they required. Two surprising results are that DMs do not dislike evaluating tradeoffs as expected, and that algorithms which generate non-extreme optimal solutions did not fare better than methods which generate only basic solutions.



6.4 Conclusion

To conclude the algorithm comparison, we briefly summarize the most important facts established.

Steuer's Contracting Cones is a very fast method but does not even converge to the extreme point of maximum utility. The method of Interactive Weighted Sums is much slower than Steuer's Contracting Cones but has a better chance of converging to the extreme point of maximum utility. Both methods however suffer from the problems associated with weighted sums algorithms. (see Steuer [56] pp. 193-200)

The Tchebbycheff method is extremely slow and will in general converge to a "near optimal" solution. The larger the reduction factor used, the better the solution will be, but the longer it will take to get there.

The method of Zionts and Wallenius is fast, easy to use and popular with DMs, but can only converge to the extreme point of maximum utility.

The Cutting Plane method is also fast and popular with DMs. It converges to the optimal solution, but generally very slowly as you approach the optimal solution. It requires many interactions per iteration whereas the Tchebbycheff method requires only one.

DMs liked the idea of entering a reference point in the Reference Point method but were dissatisfied with the solutions obtained. They found it difficult to use.

DMs liked the Visual Interactive Method's method of interaction with them but also found it extremely difficult to reach their optimal solution.

The Cutting Plane method, the method of Zionts and Wallenius and the Interactive Weighted Sums/Filtering approach were the preferred algorithms in all criteria with the exception of speed, where the latter method fared badly.

Chapter 7

Conclusion

In this research report our attention has been focused on interactive methods for solving multiple objective linear programming problems.

We first identified our area of interest within MCDM and defined a number of preliminary concepts necessary for the study. This was followed by a literature survey in which the various approaches in MOLP problem solving were described. Seven of these methods were selected for further analysis and presented fully. Various implementation issues were then discussed in some detail. Finally, the algorithms were compared in three ways, viz.

- Analytically — the algorithms were broken down to to their common basic scheme and we compared how the various steps were achieved by each algorithm. In addition, a series of questions relating to algorithm performance and interaction with decision makers were answered for each algorithm;
- Computationally — the algorithms were applied to a number of problems with known utility functions;
- From a decision maker's perspective — the algorithms were used by decision makers to solve a specific problem and their reactions to the algorithms were gauged.

This comparison showed that when the decision maker's underlying utility function is linear, all five methods tested computationally obtain the correct solution and do so rather quickly. However, when the decision maker's utility function is nonlinear but concave, only the Cutting Plane algorithm converges to the optimal solution. The Interactive Weighted Tchebbycheff method converges to a "near optimal" solution. The quality of this solution could be improved by altering the reduction factor but this would slow the algorithm down substantially. The method of Zionts and Wallenius converges on the extreme point of maximum utility extremely quickly. Steuer's Contracting Cones method and the Interactive Weighted Sums method do not even necessarily converge on the optimal extreme point.

Although decision makers found all algorithms equally easy to understand and enjoyable to use, few would use the algorithms to make a decision. decision maker found the Cutting Plane method, the method of Zionts and Wallenius, and the Interactive Weighted Sums method the most useful, and felt that they provided the most satisfactory final solutions. The decision makers' preferred method of interaction was that of specifying a reference point, but they found it difficult to understand how the projection methods use the reference point.

It should of course be stressed that the above results take only the seven algorithms discussed in this document into account. These represent a small subset of the large number of algorithms in existence.

An interesting result was that the decision makers' perception of algorithm speed did not correlate entirely with the actual speed of reaching the final solution. They judged algorithms' speed by the time between interactions rather than by the overall solution time, suggesting that they disliked having to wait for the algorithm.

It is also interesting to note that the decision makers in this study did not seem to mind evaluating tradeoffs as has been widely believed (Zionts and Wallenius [71]).

The most surprising result of all is that algorithms which only converge on the optimum extreme point were not, in general, judged worse by the decision

makers than algorithms which converge on the optimal solution.

One may, at this stage, be tempted to ask, "After all that analysis, discussion and comparison, which method is best ? " The question is well answered by MacCrimmon [84].

> "Not all multiple objective decision situations are the same, since the characteristics of both the environment and the decision maker can vary. Some methods are more suitable under particular conditions than others."

Before selecting which algorithm to use, one must consider the problem and decision maker's requirements and skills. If the problem is small, then one of the sample algorithms may well be appropriate. The larger the problem, the less viable it is to use a sample algorithm. Another important consideration is the decision maker's requirements. Although the aim is to find the *optimal* solution, it was found that most decision makers involved in the test were quite prepared to settle for a solution that they found to be satisfactory, even if it wasn't optimal. If this is the case, then an algorithm which finds the optimal extreme point may be adequate, while if the *optimal* solution is required, a method which can generate non-extreme solutions is necessary. A third factor to consider is the skill of the decision maker. While most decision makers in the test were ill-at-ease with the projection methods, a more sophisticated decision maker may find projection methods extremely useful.

In general, an algorithm should be chosen by considering the characteristics of the problem and/or decision maker together with the characteristics of the algorithms.

This study has confirmed that there is no one interactive multiple objective linear programming algorithm which is superior to all others, in all situations. Different methods are better in different situations with different decision makers. The way ahead seems to entail using a combination of algorithms. This would entail switching from one algorithm which is suitable in the early stages of the solution procedure, to one (or more) which appear to be suitable at a

later stage. Towards this end, Steuer *et al.* [54] have presented a combined Tchebbycheff/Reference Point algorithm.

Jordi's comment [23] is offered in conclusion

> "...even if the decision analyst has decided on a certain method, he should approach the overall solution procedure with an open mind and be prepared to adapt or change the method he uses if it becomes apparent that this could facilitate the entire procedure. PRAGMATISM IS THE KEY."

Appendix A

Algorithm Questionnaire

IMOLP QUESTIONNAIRE

Method: _____

Starting Time _____

The number of solutions you chose to compare at a time (if applicable) _____

Reduction factor you used (if applicable) _____

Final Solution given by the method

Cost _____

Cholesterol _____

Carbohydrate _____

No of Iterations _____

Ending Time _____

How useful do you think this method is ?

1 2 3 4 5 6 7 8 9 10

(1=totally useless, 10=very useful)

How satisfied were you with the final solution given to you ?

1 2 3 4 5 6 7 8 9 10

(1=not satisfied, 10=completely satisfied)

How easy did you find it to understand the method ?

1 2 3 4 5 6 7 8 9 10

(1=very difficult, 10=very easy)

How easy did you find it to use the method ?

1 2 3 4 5 6 7 8 9 10

(1=very difficult, 10=very easy)

How quickly did you reach your final solution ?

1 2 3 4 5 6 7 8 9 10

(1=very slowly, 10=very quickly)

Was the time taken to reach your final solution justified by the level of satisfaction the solution provided ?

YES NO DON'T KNOW

Do you think that it is necessary to run the program more than once in order to get a solution you are completely satisfied with ?

YES NO DON'T KNOW

Did you find it enjoyable to interact with the algorithm ?

YES NO DON'T KNOW

Would you use this method to solve decision making problems ?

YES	NO	DON'T KNOW

Which aspects of the algorithm did you like ?

Which aspects of the algorithm did you NOT like ?

Other comments

Appendix B

Algorithm Rating Form

IMOLP ALGORITHM RATING

Rank the following algorithms from 1 to 7, where 1 is the algorithm you liked most and 7 is the algorithm you liked least.

The Oscillating Hyperplane Method _____

Abraham's Method of Indignant Convergence _____

The Volatile Fixed Variable Method _____

The Method of Accelerating Disparity _____

Stokes' Asymptotic Method _____

The Method of Consecutive Zeros _____

Riemann's Method of Aggregated Constraints _____

What do you like about the algorithm which you ranked '1' ?

_____

_____

_____

What do you dislike about the algorithm which you ranked '7' ?

_____

_____

_____

Which algorithm(s) do you think had the best way of extracting information from the DM ?

How would you rate the importance of the following criteria in judging the performance of an algorithm ? (1=unimportant, 10=extremely important)

Speed

1 2 3 4 5 6 7 8 9 10

Quality of solution

1 2 3 4 5 6 7 8 9 10

Ease of understanding

1 2 3 4 5 6 7 8 9 10

Ease of use

1 2 3 4 5 6 7 8 9 10

Please write any other comments on back of page.

Appendix C

Selected Decision Makers' Comments

Steuer's Contracting Cones

"Very quick but didn't give such a good solution. Could have been slower but a better solution"

"None of the original solutions were very attractive and it didn't get any better"

"Liked the wide range of choices"

"Wildly oscillating solutions"

"Did not take my choices into account ... at least I thought not"

"Eliminated alternative solutions too quickly"

"Liked comparing several solutions at a time"

Interactive Weighted Sums

"I always found one good solution in list of possible solutions"

"There were periods when I became agitated due to the length of the computation process"

"Range of choices is large and adjusTable"

"The algorithm seems to be slow but sure. The trade-off in time against satisfaction seems reasonable"

"I wasn't satisfied with the algorithm but couldn't pinpoint what I didn't like"

"Relatively slow between iterations"

"It stopped changing the solutions after a few iterations"

Tchebbycheff Method

"The final solution did not seem to satisfy me entirely, and I felt that even if I had let the algorithm iterate, a satisfactory solution would not have popped up"

"Takes a very long time to run"

"You need to run the program a couple of times to be sure – and given the slowness, it might be unrealistic for big solutions"

"We can choose the direction the method will take"

"The algorithm gave me a satisfactory solution on the first go. The second time I ran it, it gave less than satisfactory answers – the program seems to depend on some random factor"

"This seems to be the slowest (but surest) algorithm so far"

"Much too slow"

The Method of Zionts and Wallenius

"I like the options you get – tradeoffs and selection"

"The algorithm seems to jump the values quite radically"

"Seemed to deviate from the direction I wished to head in"

"Simple to operate – few inputs besides choice of solution. Also choice between two solutions is less confusing"

"Disliked having only two solutions to compare"

"Jumps around a bit – from 'close to optimal' to far from it"

"Method is quick and very easy to use"

"Liked evaluating tradeoffs"

Cutting Plane Method

"The way in which I had to make a choice became boring after a while, not much interaction"

"It seemed to react in the way I expected it to. In some of the other methods the output did not seem to match the input"

"Improvement on final solution all the time"

"It takes a lot of time and is frustrating to keep choosing between A and B"

"Although many iterations, each iteration is almost instantaneous – method converges rapidly"

"Ability to direct towards preferred solution"

"Fast. Produced good solution"

Reference Point Method

"Liked specifying a goal"

"I couldn't really find out how to adjust the different goals. It didn't seem to have a pattern"

"Solutions are achieved very quickly but it seems to optimize one objective only"

"The solution wasn't good enough for the amount of time spent working on it"

"One has to juggle the objective function values quite a lot"

"Did not give good approximations to the solutions that were asked for"

"Every guess is treated separately with no information coming through from previous guesses"

Visual Interactive Method

"Liked entering my own values and rank"

"Was unsure of which information would give me the solution I was looking

for"

"You had to do a lot of juggling of objective and weight values to get a satisfactory solution (if at all)"

"The user has to do a lot more work on this one than the others to find an optimal solution. Not nice"

"Liked the fact that we are allowed to enter our objectives"

"The method ignores your objective values"

"I ran it twice and it seemed to give me exactly the same solutions even though I changed the weights"

References

[1] Benayoun R, de Montgolfier and Tergny J. "Linear Programming with Multiple Objective Functions : Step Method (STEM)". *Mathematical Programming*, Vol. 1, No. 3, 1971, pp. 366-375.

[2] Benvenuti F R. "A 'New Improved' Algorithm for Multiobjective Optimization". B.Sc. Honours Research Report, University of the Witwatersrand, Johannesburg, 1988.

[3] Benvenuti F R. "Cutting Plane Algorithms". B.Sc. Project Report, University of the Witwatersrand, Johannesburg, 1986.

[4] Benvenuti F R and Jordi K C. "An Interactive Algorithm for Multiple Objective Linear Programming Based on the Method of Zionts and Wallenius". *Proceedings of the Operations Research Society of South Africa Conference, 1990.* Kruger National Park, 1990.

[5] Brockhoff K. "Experimental Test of Multiple Criteria Decision Making Algorithms in a Modular Approach". *European Journal of Operational Research* , Vol. 22, 1985, pp. 159-166.

[6] Chankong V and Haimes Y Y. *Multiobjective Decision Making : Theory and Methodology.* North-Holland, 1983.

[7] Chankong V, Haimes Y Y, Thadathil J and Zionts S. "Multiple Criteria Optimization: A State of the Art Review". In : Haimes Y Y and Chankong V (eds), *Decision Making with Multiple Objectives.* Springer-Verlag, 1984, pp. 36-90.

[8] Clarke G M and Cooke D. *A Basic Course in Statistics - 2nd Edition.* Edward Arnold, London, 1983.

[9] Clímaco J N and Antunes C. "A Comparison of Microcomputer Implemented MOLP Methods Based on a Case Study". In : Bana e Costa C A (ed), *Readings in Multiple Criteria Decision Aid*. Springer-Verlag, 1990, pp. 445-470.

[10] Cohon J L and Marks D H. "A Review and Evaluation of Multiobjective Programming Techniques". *Water Resources Research*, Vol. 11, No. 2, 1975, pp. 208-220.

[11] Dyer J S. "An Empirical Investigation of a Man-Machine Interactive Approach to the Solution of the Multiple Criteria Problem". In : Cochrane J L and Zeleny M (eds), *Multiple Criteria Decision Making*. University of South Carolina Press, 1973, pp. 202-216.

[12] Dyer J S. "A Time-sharing Computer Program for the Solution of the Multiple Criteria Problem". *Management Science*, Vol. 19, No. 12, 1973, pp. 1379-1383.

[13] Evans G W, "An Overview of Techniques for Solving Multiobjective Mathematical Programs". *Management Science*, Vol. 30, No. 11, 1984, pp. 1268-1282.

[14] French S. "Interactive Multi-Objective Programming: Its Aims, Applications and Demands". *Journal of the Operational Research Society*, Vol. No. 9, 1984, pp. 827-834.

[15] Geoffrion A M, Dyer J S and Feinberg A. "An Interactive Approach Approach for Multi-Criterion Optimization, with an Application to the Operation of an Academic Department". *Management Science*, Vol. 19, No. 4, 1972, pp. 357-368.

[16] Gershon M and Duckstein L. "An Algorithm for the Choosing of a Multiobjective Technique". In : Hansen R (ed), *Essays and Surveys on Multiple Criteria Decision Making*, Lecture Notes in Economics and Mathematical Systems, Vol. 209, Springer-Verlag.

[17] Goicoechea A, Hansen D R and Duckstein L. *Multiobjective Decision Analysis with Engineering and Business Applications*. John Wiley and Sons, 1982.

[18] Haimes Y Y. "The Surrogate Worth Trade-off (SWT) Method and its Extensions". In : Fandel G and Gal T (eds), *Multiple Criteria Decision Making Theory and Application*. Springer-Verlag, 1980, pp. 85-108.

[19] Haksever C and Ringuest J L. "Computational Efficiency and Interactive MOLP Algorithms : An Implementation of the SIMOLP Procedure". *Computers and Operation Research*, Vol. 17, No. 1, 1990, pp. 39-50.

[20] Hansen P, Labbé M and Wendell R E. "Sensitivity Analysis in Multiple Objective Linear Programming : The Tolerance Approach". *European Journal of Operational Research*, Vol. 38, 1989, pp. 63-69.

[21] Hobbs B F. "Experiments in Multicriteria Decision Making and What we can Learn from Them : An Example". In : Haimes Y Y and Chankong V (eds), *Decision Making with Multiple Objectives*. Springer-Verlag, 1984, pp. 400-423.

[22] Hwang C L and Masud A S M. *Multiple Objective Decision Making : Methods and Applications*. Springer-Verlag Lecture Notes in Economics and Mathematical Systems, Vol. 64, Springer-Verlag, 1979.

[23] Jordi K C. "Interactive Multiple Objective Linear Programming". Ph.D. Thesis, University of the Witwatersrand, Johannesburg, 1985.

[24] Kallio M and Soismaa M. "Some Improvements to the Reference Point Approach for Dynamic MOLP". In : Carlsson C and Kochetkov Y (eds), *Theory and Practice of Multiple Criteria Decision Making*. North-Holland, 1983, pp. 277-90.

[25] Keeney R L and Raiffa H. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs* . John Wiley and Sons, 1976.

[26] Kok M. "The Interface with Decision Makers and Some Experimental Results in Interactive Multiple Objective Linear Programming Methods". *European Journal of Operational Research*, Vol. 26, 1986, pp. 96-107.

[27] Korhonen P J and Laakso J. "On Developing a Visual Interactive Multiple Criteria Method - An Outline". In : Haimes Y Y and Chankong V (eds), *Decision Making with Multiple Objectives*. Springer-Verlag, 1984, pp. 272-281.

[28] Korhonen P J and Laakso J. "A Visual Interactive Method for Solving the Multiple Criteria Problem". *European Journal of Operational Research*, Vol. 24, No. 2, 198[illegible] pp. 277-287.

[29] Korhonen P J and Laakso J. "A Pareto Race". *Naval Research Logistics*, Vol. 35, 1988, pp. 615-623.

[30] Korhonen P. "The Multiobjective Linear Programming Decision Support System VIG and its Applications". In : Bana e Costa C A (ed), *Readings in Multiple Criteria Decision Aid*. Springer-Verlag, 1990, pp. 471-491.

[31] Lazimy R. "Interactive Relaxation Method for a Broad Class of Integer and Continuous Nonlinear Multiple Criteria Problems". *Journal of Mathematical Analysis and Applications*, Vol. 116, 1986 pp. 553-573.

[32] Lofti V, Stewart T J and Zionts S. "An Aspiration-Level Interactive Model for Multiple Criteria Decision Making". Working Paper No. 701, August 1988.

[33] Loganathan G V and Sherali H D. "A Convergent Interactive Cutting-Plane Algorithm for Multiobjective Optimization". *Operations Research*, Vol. 35, No. 3, 1987, pp. 365-377.

[34] MacCrimmon K R. "An Overview of Multiple Objective Decision Making". In : Cochrane J L and Zeleny M (eds), *Multiple Criteria Decision Making*. University of South Carolina Press, 1973, pp. 18-44.

[35] Marcotte O and Soland R M. "An Interactive Branch-and-Bound Algorithm for Multiple Criteria Optimization". *Management Science*, Vol. 32, No. 1, 1986, pp. 61-75.

[36] Masud A S and Hwang C L. "Interactive Sequential Goal Programming". *Journal of the Operational Research Society*, Vol. 32, No. 5, 1981, pp. 391-400.

[37] Monarchi D E, Kisiel C C and Duckstien L. "Interactive Multiobjective Programming in Water Resources : A Case Study". *Water Resources Research*, Vol. 9, No. 4, 1973, pp. 837-850.

[38] Monarchi D E, Weber J E and Duckstien L. "An Interactive Multiple Objective Decision-Making Aid Using Nonlinear Goal Programming". In :

Zeleny M (ed), *Multiple Criteria Decision Making, Kyoto 1975*. Springer-Verlag, 1976, pp. 235-253.

[39] Murty K G. *Linear Programming*. John Wiley and Sons, 1983.

[40] Musselman K and Talavage J. "A Tradeoff Cut Approach to Multiple Objective Optimization". *Operations Research*, Vol. 26, No. 6, 1980, pp. 1424-1435.

[41] Oppenheimer K R. "A Proxy Approach to Multi-Attribute Decision Making". *Management Science*, Vol. 24, No. 6, 1978. pp. 675-689.

[42] Reeves G R and Gonzalez J J. "A Comparison of two Interactive MCDM Procedures". *European Journal of Operational Research*, Vol. 41, 1989, pp. 203-209.

[43] Rosinger E E. "Interactive Algorithm for Multiobjective Optimization". *Journal of Optimization Theory and Applications*, Vol. 35, No. 3, 1981, pp. 339-365.

[44] Sakawa M. "Interactive Multiobjective Decision Making by the Sequential Proxy Optimization Technique: SPOT". *European Journal of Operations Research*, No. 9, 1982, pp. 386-396.

[45] Sakawa M, Yano H and Yumine T. "An Interactive Fuzzy Satisficing Method for Multiobjective Linear-Programming Problems and its Application". *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-17, No. 4, 1987, pp. 654-661.

[46] de Samblanckx S, Depractere P and Muller H. "Critical Considerations Concerning the Multicriteria Analysis by the Method of Zionts and Wallenius". *European Journal of Operational Research*, Vol. 10, 1982, pp. 70-76.

[47] Scheffé H. *The Analysis of Variance*. John Wiley and Sons, 1959.

[48] Seo F and Sawaka M. *Multiple Criteria Decision Analysis in Regional Planning*. D Reidel, 1988.

[49] Shin W S and Ravindran A. "Interactive Multiple Objective Optimization : Survey I — Continuous Case". *Computers and Operations Research*, Vol. 18, No. 1, 1991, pp. 97-114.

[50] Starr M.K and Zeleny M. "MCDM – State and Future of the Arts", *TIMS Studies in the Management Sciences*, Vol. 6, 1977, pp. 5-29.

[51] Steuer R E. "Multiple Objective Linear Programming with Interval Criterion Weights". *Management Science*, Vol. 23, No. 3, 1976, pp. 305-316.

[52] Steuer R E and Gardiner L R. "Interactive Multiple Objective Programming: Concepts, Current Status, and Future Directions". In : Bana e Costa C A (ed), *Readings in Multiple Criteria Decision Aid*. Springer-Verlag, 1990, pp. 413-444.

[53] Steuer R E. "An Interactive Multiple Objective Linear Programming Procedure". *TIMS Studies in the Management Sciences*, Vol. 6, 1977, pp. 225-239.

[54] Steuer R E, Whisman A W and Silverman J. "A Combined Tchebbycheff/Reference Point Interactive Multiobjective Programming Procedure". Unpublished Paper, January 1987.

[55] Steuer R E and Choo E. "An Interactive Weighted Tchebbycheff Procedure for Multiple Objective Programming". *Mathematical Programming*, Vol. 26, 1983, pp. 345-362.

[56] Steuer R E. *Multiple Criteria Optimization : Theory, Computation and Application*. John Wiley and Sons, 1986.

[57] Steuer R E and Harris F W. "Intra-set Point Generation and Filtering in Decision and Criterion Space". *Computers and Operations Research*, Vol. 7, No. 1, 1980, pp. 41-53.

[58] Stewart T J. "An Interactive Multiple Objective Linear Programming Method Based on Piecewise-Linear Additive Value Functions". *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-17, No. 5, 1987, pp. 799-804.

[59] Tou J T and Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, 1974.

[60] Wallenius J. "Comparative Evaluation of some Interactive Approaches to Multicriterion Optimization". *Management Science*, Vol. 20, No. 12, 1975, pp. 1387-1396.

[61] Wierzbicki A P. "The use of Reference Objectives in Multiobjective Optimization". In : Fandel G and Gal T (eds), *Multiple Criteria Decision Making Theory and Application*. Springer-Verlag, 1980, pp. 468-486.

[62] White D J. "A Selection of Multi-Objective Interactive Programming Methods". In : French S, Hartley R, Thomas L C and White D J (eds), *Multi-Objective Decision Making*. Academic Press, 1983, pp. 99-126.

[63] Wierzbicki A P. "A Mathematical Basis for Satisficing Decision Making". *Mathematical Modelling*, Vol. 3, No. 5, 1982, pp. 391-409.

[64] Yu P L. "Multiple Criteria Decision Making : Five Basic Concepts". In : Nemhauser G L, Rinooy Kan A H G and Todd M J (eds), *Optimization*. North-Holland, 1989.

[65] Zeleny M. "The Theory of the Displaced Ideal". In : Zeleny M (ed), *Multiple Criteria Decision Making, Kyoto 1975*. Springer-Verlag, 1976, pp. 153-206.

[66] Zeleny M. *Multiple Criteria Decision Making*. McGraw-Hill, 1982.

[67] Zionts S. "Multiple Criteria Mathematical Programming : An Updated Overview and Several Approaches". In : Serafini P (ed), *Mathematics of Multi Objective Optimization*. Springer-Verlag, 1985.

[68] Zionts S. *Linear and Integer Programming*. Prentice-Hall, 1974.

[69] Zionts S and Wallenius J. "An Interactive Programming Method for Solving the Multiple Criteria Problem". *Management Science*, Vol. 22, No. 6, 1976, pp. 652-663.

[70] Zionts S. "Methods for Solving Management Problems Involving Multiple Objectives". In : Fandel G and Gal T (eds), *Multiple Criteria Decision Making Theory and Application*. Springer-Verlag, 1980, pp. 540-558.

[71] Zionts S and Wallenius J. "An Interactive Multiple Objective Linear Programming Method for a Class of Underlying Nonlinear Utility Functions". *Management Science*, Vol. 29, No. 5, 1983, pp. 519-529.

Author: Benvenuti Fabrizio Roberto.
Name of thesis: A Comparative Study Of Interactive Algorithms For Multiple Objective Linear Programming.

PUBLISHER:
University of the Witwatersrand, Johannesburg
©2015

LEGALNOTICES:

Copyright Notice: All materials on the University of the Witwatersrand, Johannesburg Library website are protected by South African copyright law and may not be distributed, transmitted, displayed or otherwise published in any format, without the prior written permission of the copyright owner.

Disclaimer and Terms of Use: Provided that you maintain all copyright and other notices contained therein, you may download material (one machine readable copy and one print copy per page)for your personal and/or educational non-commercial use only.

The University of the Witwatersrand, Johannesburg, is not responsible for any errors or omissions and excludes any and all liability for any errors in or omissions from the information on the Library website.