A DECISION SUPPORT SYSTEM FOR MULTI-OBJECTIVE PROGRAMMING PROBLEMS

by

MOETI JOSEPH RANGOAGA

submitted in part fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in the subject

OPERATIONS RESEARCH

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: PROF M K LUHANDJULA

NOVEMBER 2009

Declaration

I declare that A DECISION SUPPORT SYSTEM FOR MULTIOBJECTIVE PROGRAMMING PROBLEMS is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

Signature: .....
MJ RANGOAGA

Date: 06 May 2010

Abstract

Many concrete problems may be cast in a multi-objective optimisation framework. The redundancy of existing methods for solving multi-objective programming problems susceptible to inconsistencies, coupled with the necessity for making inherent assumptions before using a given method, make it hard for a nonspecialist to choose a method that fits the situation at hand well. Moreover, using a method blindly, as suggested by the hammer principle (when you only have a hammer, you want everything in your hand to be a nail) is an awkward approach at best and a caricatural one at worst. This brings challenges to the design, development, implementation and deployment of a Decision Support System able to choose a method that is appropriate for a given problem and to apply the chosen method to solve the problem under consideration. The choice of method should be made according to the structure of the problem and the decision maker's opinion. The aim here is to embed a sample of methods representing the main multi-objective programming techniques and to help the decision maker find the most appropriate method for his problem.

Key terms:

Achievement function; Database; Decision maker; Decision support system; Lexicographic goal program; Modelbase; Multi-objective program; Pareto optimality; Slater constraint qualification; Software

Acknowledgemnents

I wish to thank the following people:

- my supervisor Prof MK Luhandjula for his advices;
- my colleagues in the Department of Decision Sciences;
- Elias Khensani Malebye for helping me to find the necessary software;
- Thabo Masemola and Tshegofatso Manakana for helping me to implement the Decision Support System described in this research work;
- Prof KM Miettinen and Prof R Caballero for their advices;
- my wife Nkefi, daughters Ofentse and Paballo, and brother Lesedi for their support.

Contents

Introduction

Mathematical programming is an important tool in the arsenal at a decision maker's disposal. Indeed, many real-life problems, see for example [22, 33, 49], may be cast in a mathematical programming framework.

Theoretical underpinnings for convex mathematical programming, particularly for linear programming, are now well established. An interested reader is referred to [40, 57] for further details on these matters.

As a result, a broader array of techniques has been developed. We mention, without any claim to exhaustivity, the simplex algorithm [21], the ellipsoid method [43] and the Karmarkar's method [34, 66] for linear programming; the cutting-plane methods [43] and the penalty methods [5] for nonlinear programming.

Software with powerful computational and visualisation capabilities like LINGO [58], CPLEX [53] and XPRESS [55] have also been developed. All the above-mentioned methods and software rely heavily on the assumption that there is only one function to optimise and that all parameters involved have well-known fixed values.

In some concrete real-life problems, the decision maker is confronted with several conflicting goals [51, 26]. Moreover, these problems may also contain some level of uncertainty about the values to be assigned to various parameters or about the layout of the problem components [44, 64]. In this dissertation, the focus is on the simultaneous incorporation of several conflicting goals in an optimisation setting.

The simplistic approach, consisting of substituting arbitrarily a single objective function for several conflicting ones, often leads to a bad caricature of the reality. Such an approach has no other option but to churn out meaningless outcomes.

The purpose of this research work is twofold. Firstly, it aims at raising awareness of the most important techniques used to single out satisfying solutions for a

mathematical program with several conflicting goals. Secondly, it describes a Decision Support System (DSS) able to help a user confronted with such a problem. The system should assist at two levels: it should be able to choose an appropriate technique for solving the problem at hand and it should help in singling out a satisfying solution that meets the decision maker's needs.
The dissertation is organised as follows: The first chapter introduces basic concepts on multi-objective optimisation. In the second chapter, we describe the most used methods for solving multi-objective programming problems. The DSS paradigm is discussed in chapter 3. Chapter 4 is devoted to a description of our own DSS for multi-objective programming problems. Finally, we make some concluding remarks along with suggestions for further developments in the field of multi-objective programming.

Chapter 1

Multi-objective programming problems

1.1 Problem setting and general notation

A multi-objective program is a problem of the form:

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x), \ldots, f_k(x)] \ , k \geq 2 \\ \text{subject to} \\ x \in X = \{x \in \mathbb{R}^n | g_j(x) \leq 0 \ ; \ j = 1, \ldots, m\} \end{cases} \tag{P}$$

where $f_i(x); \quad i = 1, \ldots, k$ and $g_j(x); \ j = 1, \ldots, m$ are real-valued functions of $\mathbb{R}^n$.

Many real-life problems may be cast in the form of (P). An interested reader is referred to [26, 48] for examples of such problems.

1.2 Background materials

Throughout this dissertation, all the vectors are assumed to be column vectors. If x and x^* are vectors of $\mathbb{R}^n$, the notation $x^T x^*$ (where T stands for the transpose) denotes the scalar product of x and x^*. Moreover, the following notation is used:

- $\mathbb{R}^n_+ \;=\; \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n \mid x_i \geq 0, i \;=\; 1, \ldots, n\}$.
- $|| \; x \; || \;\; = \;\; \left(\sum_{i \; = \; 1}^{n} x_i^2 \right)^{1/2}$.
- For $X \subseteq \mathbb{R}^n$, $\text{dist}(x^*, X) \;=\; inf_{x \in X} ||x - x^*||$.
- $B(x^*, \delta) \;=\; \{x \in \mathbb{R}^n : \; ||x - x^*|| < \delta\}$.

The following definitions in connection with the multi-objective program (P) are needed in the sequel.

Definition 1.2.1 *Let $d \in \mathbb{R}^n$, $d \neq 0$, we say that d is a* feasible direction *emanating from $x \in X$ if there exists $\alpha^* > 0$ such that*

$$x + \alpha d \in X \quad for \;\; all \; \alpha \in [0, \alpha^*].$$

Definition 1.2.2 *A constraint $g_i(x) \leq 0$ is said to be* active *at a point $x^* \in X$ if $g_i(x^*) = 0$.*
In what follows we denote by $J(x^*)$ the set of indexes *of active constraints at x^*. In other words,*

$$J(x^*) = \{j \in \{1, \ldots, m\} | g_j(x^*) \;=\; 0\}.$$

Definition 1.2.3 *Assume f is twice continuously differentiable at x^*. The gradient of f at x^* is denoted by $\nabla f(x^*)$ and the* Hessian matrix *of $f(x)$ at x^* is denoted by $\nabla^2 f(x^*)$. This means:*

$$\nabla f(x^*) \;\; = \;\; \begin{pmatrix} \dfrac{\partial f(x^*)}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(x^*)}{(\partial x_n)} \end{pmatrix}$$

and

$$\nabla^2 f(x^*) \;\; = \;\; \begin{pmatrix} \dfrac{\partial^2 f(x^*)}{\partial x_1^2} \cdots \dfrac{\partial^2 f(x^*)}{(\partial x_1 \partial x_n)} \\ \vdots \qquad \ddots \qquad \vdots \\ \dfrac{\partial^2 f(x^*)}{(\partial x_n \partial x_1)} \cdots \dfrac{\partial^2 f(x^*)}{\partial x_n^2} \end{pmatrix}$$

1.3 Convex multi-objective program

Definition 1.3.1 *A function* $f : \mathbb{R}^n \to \mathbb{R}$ *is* convex *if for all* $x, y \in \mathbb{R}^n$ *and* $\alpha \in [0, 1]$ *we have*

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y).$$

Definition 1.3.2 *A set* $C \subseteq \mathbb{R}^n$ *is* convex *if for all* $x, y \in C$ *and* $\alpha \in [0, 1]$*, we have*

$$\alpha x + (1 - \alpha)y \in C.$$

Definition 1.3.3 *Consider the multi-objective program (P).*
If the feasible set X *is convex and if the objective functions* $f_i(x);\ i = 1, \ldots, k$ *are convex, then (P) is said to be a* convex multi-objective program*.*

As will become evident in chapter 2, most of the methods described for multi-objective programming are applicable only to convex multi-objective programs. We will also indicate what can be done in the case where a multi-objective program is not convex.

1.4 Solution concepts used in multi-objective programming

In a multi-objective optimisation context, unless the objective functions are unconflicting, the optimum optimorum does not exist. As a matter of fact, owing to the conflict between the objective functions such an optimum optimorum lies outside the feasible set.
It is therefore worthwhile to make the meaning of optimality in this context explicit. Several optimality concepts are discussed in the literature [22, 26, 48]. We present those most frequently used in the following subsections.

1.4.1 Pareto optimality

Definition 1.4.1 *Consider the problem (P). $x^* \in X$ is said to be a* Pareto optimal (efficient) solution *for (P) if no other $x \in X$ exists, such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, \ldots, k$ and $f_j(x) < f_j(x^*)$ for some j.*

Remark 1.4.1 *A Pareto optimal solution is also called a* globally Pareto optimal solution.

Remark 1.4.2 *In the sequel, the set of all* efficient solutions *for (P) is denoted by E.*

Definition 1.4.2 *Consider the multi-objective program (P). $x^* \in X$ is* locally Pareto optimal *for (P) if $\delta > 0$ exists such that x^* is Pareto optimal for (P) in $X \cap B(x^*, \delta)$.*

Theorem 1.4.1 *Assume that the multi-objective program (P) is convex. Then every locally Pareto optimal solution for (P) is also globally Pareto optimal for (P).*

Proof.
Assume (P) is convex and let $x^* \in X$ be locally Pareto optimal for (P). We can then find $\delta > 0$ and a neighbourhood $B(x^*, \delta)$ of x^* such that there is no $x \in X \cap B(x^*, \delta)$ for which $f_i(x) \leq f_i(x^*)$ for all $i = 1, \ldots, k$ and
$f_j(x) < f_j(x^*)$ for some j.
Assume that x^* is not globally Pareto optimal for (P). Then there exists some other point $x^\circ \in X$ such that

$$f_i(x^\circ) \leq f_i(x^*) \text{ for all } i = 1, \ldots, k \text{ and } f_j(x^\circ) < f_j(x^*) \text{ for some j.} \quad (1.1)$$

Let $\hat{x} = \beta x^\circ + (1 - \beta)x^*$, where $\beta \in (0, 1)$ is chosen such that $\hat{x} \in B(x^*, \delta)$. Then, by the convexity of X and the convexity of f_i; $i = 1, \ldots, k$, we have:

$$\hat{x} \in X \quad (1.2)$$

and

$$f_i(\hat{x}) \leq \beta f_i(x^\circ) + (1-\beta) f_i(x^*) \leq \beta f_i(x^*) + (1-\beta) f_i(x^*) = \ f_i(x^*) \qquad (1.3)$$

for every $i = 1, \ldots, k$ and

$$f_j(\hat{x}) < \beta f_j(x^\circ) + (1-\beta) f_j(x^*) < \beta f_j(x^*) + (1-\beta) f_j(x^*) = \ f_j(x^*) \qquad (1.4)$$

for some j.

Since x^* is locally Pareto optimal for (P) and $\hat{x} \in B(x^*, \delta)$, we must also have

$$f_i(\hat{x}) = f_i(x^*) \ ; \ i = 1, \ldots, k. \qquad (1.5)$$

But (1.4) contradicts (1.5) and we are done.■

1.4.2 Weak and proper Pareto optimality

Definition 1.4.3 *$x^* \in X$ is* weakly Pareto optimal *for (P) if another vector $x \in X$ such that $f_i(x) < f_i(x^*)$ for all $i = 1, \ldots, k$ does not exist.*

Definition 1.4.4 *$x^* \in X$ is* properly Pareto optimal *for (P) if it is Pareto optimal for (P) and if there is some real number $M > 0$ such that for each f_i and each $x \in X$ satisfying $f_i(x) < f_i(x^*)$ there is at least one f_j such that $f_j(x^*) < f_j(x)$ and*

$$\frac{f_i(x^*) - f_i(x)}{f_j(x) - f_j(x^*)} \leq M.$$

1.4.3 Ideal points of a multi-objective program

Definition 1.4.5 *Consider the multi-objective program (P) and assume that for each objective function the optimal objective value is finite and is equal to z_i^*. For each of the k objective functions, there is one different optimal solution.*

An objective vector constructed with these individual optimal objective values constitutes the ideal objective vector
$z^* = (z_1^*, \ldots, z_k^*)$.
$x^* \in X$ *such that* $f_l(x^*) = z_l^*$ *for all* $l \in \{1, \ldots, k\}$ *is called the* ideal point *of the multi-objective program (P).*

As a matter of fact such a x^* optimise all the objective functions. Unfortunately this point is often outside the feasible set.

1.5 Kuhn and Tucker conditions for Pareto optimality

In this section, we discuss necessary and sufficient conditions for Pareto optimality and for proper Pareto optimality for a multi-objective program. We assume that all the functions involved are continuously differentiable.

1.5.1 Necessary and sufficient conditions for Pareto optimality

Theorem 1.5.1 *A necessary condition for* $x^* \in X$ *to be Pareto optimal for (P) is that there are vectors* $\lambda \in \mathbb{R}^k$ *and* $\mu \in \mathbb{R}^m$ *with* $\lambda_i \geq 0; \quad i = 1, \ldots, k$ *and* $\mu_j \geq 0; \quad j = 1, \ldots, m$ *and* $(\lambda, \mu) \neq (0, 0)$*, such that the following conditions hold:*

$$\sum_{i=1}^{k} \lambda_i \nabla f_i(x^*) + \sum_{j=1}^{m} \mu_j \nabla g_j(x^*) = 0 \tag{1.6}$$

$$\mu_j g_j(x^*) = 0 \ ; \ j = 1, \ldots, m. \tag{1.7}$$

The proof of this result may be found elsewhere [19]. Equations (1.6) and (1.7) are called Kuhn and Tucker conditions for Pareto optimality.

Chapter 2

Methods for solving multi-objective programming problems

2.1 Preamble

This chapter is devoted to methods for finding a compromise solution for a multi-objective programming problem. A look at the literature (see for example [26, 33, 48, 64]) reveals that the most frequently used methods for solving multi-objective programs fit into five categories: no-preference methods, a priori methods, a posteriori methods, interactive methods and heuristics methods. In the following sections we briefly discuss the general methodological principle in each of these categories. It is worth remembering that the problem at hand is:

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x), \ldots, f_k(x)] \ , k \geq 2 \\ \text{subject to} \\ x \in X = \{x \in \mathbb{R}^n | g_j(x) \leq 0 \ ; \ j = 1, \ldots, m\}. \end{cases} \tag{P}$$

Where $f_i(x); \quad i = 1, \ldots, k$ and $g_j(x); \ j = 1, \ldots, m$ are real-valued functions of $\mathbb{R}^n$.

2.2 No-preference methods

In the no-preference methods, the analyst solves the multi-objective program (P) without any subjective information from the decision maker.The methods in this category include the compromise programming (CP) method [25, 48] and the multi-objective proximal bundle (MPB) method [41, 49].

2.2.1 Compromise programming method

Definition 2.2.1 *We call the* aspiration level z_i *the level of the objective function* $f_i(x)$ *that is satisfactory to the decision maker.*
If $z_i \;\; i = 1, \ldots, k$ *are aspiration levels, then* $z = (z_1 \ldots, z_k)$ *is called a* reference point.

The compromise programming method, also known as the global criterion method, picks a solution which is closest to some reference point for a given distance [22]. In what follows we use the following distance

$$d(x, y) = \left(\sum_{i=1}^{k} |f_i(x) - z_i^*|^p \right)^{1/p} \qquad p > 1.$$

Here we take the ideal vector z^* as the reference vector. The method proceeds as follows:

Step 1
Choose a distance d, fix a reference point z^* and solve one of the following optimisation problems according to the value of p.

$$\begin{cases} \text{Min} \left(\sum_{i=1}^{k} |f_i(x) - z_i^*|^p \right)^{1/p} \\ \qquad\qquad\qquad\qquad if \;\; p \in [1, \infty) \\ \text{subject to} \\ x \in X. \end{cases} \tag{P1}$$

$$\begin{cases} \text{Min} \max\limits_{i=1,\ldots,k} \quad [|f_i(x) - z_i^*|] \\ \qquad\qquad\qquad\qquad if\ p = \infty \\ \text{subject to} \\ x \in X. \end{cases} \tag{P2}$$

where z_i^* is the i-th component of the reference point.

Step 2
Present the solution obtained to the decision maker.

Step 3
Stop.

It is clear that (P2) is a nondifferentiable multi-objective program. Nevertheless, we can put this program in the following differentiable form:

$$\begin{cases} \text{Min} \quad \alpha \\ \text{subject to} \\ \alpha \geq f_i(x) - z_i^* \\ x \in X. \end{cases} \tag{P3}$$

With regard to (P1), when all the functions involved in (P) are linear and $p = 1$, then it is a linear program that can be solved by the simplex method [21, 66]. For $p = 2$, (P1) is a quadratic program problem that can be solved by the gradient method [43]. If some of the objective functions involved in (P) are nonlinear, the primal methods can be used to solve (P3) [43].
The next two theorems tell us something about the Pareto optimality of solutions obtained by this method.

Theorem 2.2.1 *If x^* is a solution of (P1), then x^* is Pareto optimal for (P).*

Proof
We proceed by ad absurdum reasoning.
Let $x^* \in X$ be a solution of (P1) and assume that x^* is not Pareto optimal for (P). There is then a point $x \in X$ such that $f_i(x) \leq f_i(x^*)$ for $i = 1, 2 \ldots, k$ and

$f_j(x) < f_j(x^*)$ for some j. Now, because $p \in [1, \infty)$, we have that

$$(f_i(x) - z_i^*)^p \leq (f_i(x^*) - z_i^*)^p \ ; \ i \ = \ 1, 2 \ldots, k. \tag{2.1}$$

$$(f_j(x) - z_j^*)^p < (f_j(x^*) - z_j^*)^p \ \text{ for some } j. \tag{2.2}$$

From (2.1) and (2.2) we can write that

$$\sum_{i=1}^{k} (f_i(x) - z_i^*)^p < \sum_{i=1}^{k} (f_i(x^*) - z_i^*)^p \, .$$

Hence

$$\left(\sum_{i=1}^{k} (f_i(x) - z_i^*)^p \right)^{\frac{1}{p}} < \left(\sum_{i=1}^{k} (f_i(x^*) - z_i^*)^p \right)^{\frac{1}{p}} .$$

This is in contradiction to the assumption that x^* is a solution of the problem (P1). Therefore, x^* is Pareto optimal for (P).■

Theorem 2.2.2 *Solutions to problem (P2) contain at least one Pareto optimal solution for the multi-objective program (P).*

Proof

We proceed again by ad absurdum reasoning.

Suppose that none of the solutions to problem (P2) is a Pareto optimal solution to (P). Let $x^* \in X$ be a solution of (P2). Then x^* is not Pareto optimal for (P). Therefore, a solution $x \in X$ exists for which

$$f_i(x^*) \geq f_i(x) \ ; \ i = 1, \ldots, k$$

and

$$f_j(x^*) > f_j(x) \ \text{ for some } j.$$

Now

$$f_i(x^*) - z_i^* \geq f_i(x) - z_i^* \ ; \ i = 1, \ldots, k$$

with the strict inequality holding for one j. Therefore,

$$\max_i \left[f_i(x^*) - z_i^* \right] \geq \max_i \left[f_i(x) - z_i^* \right] .$$

Since $x^* \in X$ is a solution for (P2), the last relation should be an equality and $x \in X$ is also a solution for (P2). This means that, if $x^* \in X$ is a Pareto optimal solution for (P2), then $x \in X$ is also Pareto optimal solution for (P). Therefore, the problem (P2) has at least one Pareto optimal solution for a multi-objective program (P).■

Example 2.2.1

Consider the multi-objective program:

$$\begin{cases} \text{Min } \left(\dfrac{x_1 + 2x_2}{x_1 + x_2 + 1}, \dfrac{2x_1 + x_2}{2x_1 + 3x_2 + 1} \right) \\ \text{subject to} \\ -x_1 + 2x_2 \leq 3 \\ 2x_1 - x_2 \leq 3 \\ x_1 + x_2 \geq 3 \\ 2 \leq x_1 \leq 25 \\ 1 \leq x_2 \leq 9,5 \end{cases} \tag{P4}$$

Let us take the ideal vector as reference vector, that is:

$$z^* \ = \ (z_1^*; z_2^*)^T = (1; 0,55)$$

Taking $p = 2$ and putting the problem into the form of (P1) and squaring the objective function yields the following program:

$$\begin{cases} \text{Min } \left(\left(\dfrac{x_1 + 2x_2}{x_1 + x_2 + 1} - 1 \right)^2 + \left(\dfrac{2x_1 + x_2}{2x_1 + 3x_2 + 1} - 0,55 \right)^2 \right)^{1/2} \\ \text{subject to} \\ -x_1 + 2x_2 \leq 3 \\ 2x_1 - x_2 \leq 3 \\ x_1 + x_2 \geq 3 \\ 2 \leq x_1 \leq 25 \\ 1 \leq x_2 \leq 9,5 \end{cases} \tag{P5}$$

Solving this program we obtain the solution $x^* = (3, 3)$ with an optimal value of $0,298214285$. By virtue of Theorem 2.2.1, this solution is Pareto optimal for (P4).

The compromise programming method brings extra parameters such as p and the reference point into the problem under consideration. This is one of the inconveniences of this method.

2.2.2 Multi-objective proximal bundle method

Consider again the multi-objective program (P). Let us recall that it has the form

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x), \ldots, f_k(x)] \ , k \geq 2 \\ \text{subject to} \\ x \in X = \{x \in \mathbb{R}^n | g_j(x) \leq 0 \ ; \ j = 1, \ldots, m\} \end{cases} \tag{P}$$

where $f_i(x); \quad i = 1, \ldots, k$ and $g_j(x); \ j = 1, \ldots, m$ are real-valued functions of $\mathbb{R}^n$. Before roughly discussing the multi-objective proximal bundle (MPB) method we need the following definitions.

Definition 2.2.2 *Consider (P). An* improvement function *with respect to (P) is a function $H : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined as follows:*

$$H(x, y) = max\{f_i(x) \ - \ f_i(y) : g_j(x) \leq 0 \text{ and } i = 1, \ldots, k, \ j = 1, \ldots, m\}.$$

Definition 2.2.3 *Assume that (P) is a convex multi-objective program. Then (P) satisfies the* Slater constraint qualification *if there exists some $x \in X$ satisfying conditions: $g_j(x) < 0; \ j = 1, \ldots, m$.*

The MPB method is based on the following theorem, the proof of which may be found in [48].

Theorem 2.2.3 *A necessary condition for a point $x^* \in \mathbb{R}^n$ to be weakly Pareto optimal for (P) is that the improvement function $H(x, x^*)$ attains its minimum at x^*. If (P) is a convex multi-objective program and the Slater constraint qualification is satisfied, then the condition above is also sufficient.*

The MPB method is as follows:

Step 1
Initialise $h = 0$ and let x^0 be the initial point.

Step 2
Let x^h be the current approximation to the solution of (P), at iteration h.

$$\begin{cases} \text{Min} H(x^h + d, x^h) \\ \text{subject to} \\ d \in \mathbb{R}^n \text{ and } d \neq 0 \end{cases} \tag{P6}$$

Step 3
The analyst use the line search algorithm to find the new solution.

Step 4
Present the solution to the decision maker. The decision maker accepts the solution as it is.

2.3 A priori methods

Unlike no-preference methods, the general principle of a priori methods is to first take into consideration the opinions and preferences of the decision maker before solving the multi-objective program. The most used a priori methods are goal programming and lexicographic goal programming methods [4, 14]. These methods are briefly presented in what follows.

2.3.1 Goal programming method

The main idea behind the goal programming method is to find solutions that are close to predefined targets. Therefore, in the goal programming method, the decision maker should fix the targets for each objective function. He then solves a single objective function program aiming at minimising the sum of deviations to the targets. If b_i is the target for objective i, the corresponding single objective program is as follows:

$$\begin{cases} \text{Min}\left[\sum_{i=1}^{k}\left(d_i^- + d_i^+\right)\right] \\ \text{subject to} \\ f_i(x) + d_i^- - d_i^+ \;=\; b_i \;;\; 1,\ldots,k \\ x \in X \\ d_i^-, d_i^+ \geq 0 \;;\; i = 1,\ldots,k. \end{cases} \tag{P7}$$

The variables d_i^- and d_i^+ are underachievement and overachievement of the ith goal respectively.
Using the goal programming method, the decision maker can allocate some weights to each objective so as to organise the objectives hierarchically, that is he can specify real numbers $w_i \;\; (i = 1,\ldots,k);\; w_i \geq 0$ and $\sum_{i=1}^{k} w_i = 1$. In this case, the corresponding single objective program is:

$$\begin{cases} \text{Min}[w_1 h_1(d_1^-; d_1^+) + w_2 h_2(d_2^-; d_2^+) + \ldots + w_k h_k(d_k^-; d_k^+)] \\ \text{subject to} \\ f_i(x) + d_i^- - d_i^+ \;=\; b_i \;;\; 1,\ldots,k \\ x \in X \\ d_i^-, d_i^+ \geq 0 \;;\; \;=\; 1,\ldots,k. \end{cases} \tag{P8}$$

Here $h_i(d_i^-; d_i^+);\; i \;=\; 1,\ldots,k$ are some linear functions of the deviational variables.
Under certain conditions, the solution to program (P8) is Pareto optimal for (P). This is the subject matter of the following theorem, the proof for which may be found elsewhere [48].

Theorem 2.3.1 *The solution to (P8) is Pareto optimal for (P) if all the deviational variables d_i^+ and d_i^- have positive values at the optimum.*

The goal programming method can be summarised as follows:

Step 1:

Set targets and weights for each objective function of (P).

Step 2:
Formulate and solve (P7) if weights are equal or formulate and solve (P8) if weights are not equal.

Step 3:
Present the solution to the decision maker. If he is happy with the solution, stop. Otherwise go back to step 1.

Example 2.3.1

Consider the following multi-objective program:

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x), f_4(x), f_5(x)] \ , \\ \text{subject to} \\ (x_1-3)^2+(x_2-3)^2+(x_3-3)^2 \leq 4 \\ x_1+x_2+x_3 \leq 15 \end{cases} \tag{P9}$$

where

$$\begin{aligned} f_1(x) &= (x_1+x_2+x_3)^{\frac{5}{2}} \\ f_2(x) &= (x_1-4)^4+(x_2-3)^4+x_3^4 \\ f_3(x) &= 3x_1^3+(x_2-1)^4+2(x_3-20)^4 \\ f_4(x) &= \frac{1}{x_1+x_2+x_3} \\ f_5(x) &= (x_1-3)^2+\ln(x_3+x_4) \end{aligned}$$

Assume that the targets for each objective have been set by the decision maker as follows: $b_1 = 80$, $b_2 = 20$, $b_3 = 0,15$, $b_4 = 75000$ and $b_5 = 60$. Assume also that the objectives have equal weights. Then (P7) reads

$$
\begin{cases}
\text{Min}\left[\sum_{i=1}^{5}(d_i^- + d_i^+)\right] \\
\text{subject to} \\
(x_1 + x_2 + x_3)^{\frac{5}{2}} + d_1^- - d_1^+ \;=\; 80 \\
(x_1 - 4)^4 + (x_2 - 3)^4 + x_3^4 + d_2^- - d_2^+ \;=\; 20 \\
3x_1^3 + (x_2 - 1)^4 + 2(x_3 - 20)^4 + d_3^- - d_3^+ \;=\; 0,15 \\
\dfrac{1}{x_1 + x_2 + x_3} + d_4^- - d_4^+ \;=\; 75000 \\
x_1 + x_2 + x_3 \leq 15 + d_5^- - d_5^+ \;=\; 60 \\
(x_1 - 3)^2 + (x_2 - 3)^2 + (x_3 - 3)^2 \leq 4 \\
x_1 + x_2 + x_3 \leq 15 \\
d_i^-, d_i^+ \geq 0 \;;\; i = 1, \ldots, k.
\end{cases}
\qquad (\text{P10})
$$

Solving this program yields the following solution:

$$
\begin{aligned}
& d_1^- = d_2^- = d_3^- = d_4^+ = d_5^+ = 0, \\
& d_4^- = 74999.9, \\
& d_5^+ = 0.278939, \\
& d_1^+ = 319.3253, \\
& d_2^+ = 606.0207, \\
& d_3^+ = 101347.2, \\
& x_1 = 2.987495, \\
& x_2 = 2.990755,
\end{aligned}
$$

$x_3 = 4.99994$ and

$x_4 = 0.8640776E + 26.$

Since d_i^+ and d_i^- have positive values at the optimum for (P10), according to Theorem 2.3.1, the solution obtained is Pareto optimal for the multi-objective program (P9).
The major strength of the goal programming method is its simplicity. Although this method offers a great deal of flexibility for solving multi-objective programming problems, there are some difficulties associated with it. These include the fact that it does not always produce Pareto optimal solutions for (P) and it may be difficult to set weights or targets for objectives.

2.3.2 Lexicographic goal programming method

The lexicographic goal programming method [4] deals with situations where the objective functions of a multi-objective program are arranged according to their importance while a target is not given for each objective function. Consider the multi-objective program (P) and assume that

$$f_1(x) \gg f_2(x) \gg \ldots \gg f_k(x),$$

which means that $f_1(x)$ is more important than $f_2(x)$, $f_2(x)$ is more important than $f_3(x)$ and so on.
In the following algorithm we need to check the uniqueness of the solution. In the linear case, the way to check uniqueness is as follows:

- Find feasible basic solution
- Compute value of the objective function at each feasible basic solution
- See if there is more than one solution with minimal value

In the nonlinear case, the way to check uniqueness is as follows:

- Find all stationary points of the objective function in the feasible set
- Compare value of the objective function at each stationary point

- See if there is more than one solution with minimal value at each stationary point

The lexicographic goal programming method proceeds as follows.

Step 1:
Solve

$$\begin{cases} \text{Min } f_1(x) \\ \text{subject to} \\ x \in X. \end{cases} \tag{P11}$$

Let x^* be the solution to (P11). If x^* is unique, then x^* is considered to be the preferred solution to the entire problem. If x^* is not unique, go to step 2.

Step 2:
Solve

$$\begin{cases} \text{Min } f_2(x) \\ \text{subject to} \\ x \in X \\ f_1(x) \ = \ f_1(x^*). \end{cases} \tag{P12}$$

The procedure is repeated until a unique solution has been obtained. If the procedure stops at objective $f_i(x)$ $i < k$, then the objectives that are less important than $f_i(x)$ will be ignored.
The question that can be raised now is that of the Pareto optimality of the solution obtained by this method. The answer to this question is given by the following theorem.

Theorem 2.3.2 *If the lexicographic goal programming method is used to solve the multi-objective program (P), then the solution obtained is Pareto optimal.*

Proof
Let $x^* \in X$ be a solution obtained by the lexicographic goal programming

method. Assume that x^* is not Pareto optimal for (P). Therefore there exists a point $x \in X$ such that

$$f_i(x) \leq f_i(x^*) \; i = 1, \ldots, k \tag{2.3}$$

and for one j we have:

$$f_j(x) < f_j(x^*) \tag{2.4}$$

Let $i = 1$, then

$f_1(x)$ attains its minimum at x^*. Since $f_1(x) \leq f_1(x^*)$, we should have that:

$$f_1(x) = f_1(x^*).$$

By a similar reasoning we also have

$$f_j(x) = f_j(x^*); \; j = 2, \ldots, m.$$

We therefore have

$$f_i(x) = f_i(x^*) \; ; \; i = 1, \ldots, k \tag{2.5}$$

wherever the lexicographic goal program stops.

(2.5) contradicts (2.4).

Therefore, our supposition is false and x^* is Pareto optimal for (P).■

Example 2.3.2

Consider the following multi-objective program:

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x)] \\ \text{subject to} \\ x_1^2 + x_2^2 \leq 9 \\ x_2 - x_1 x_2^3 \geq 0 \\ x_1 \geq 0,50 \\ x_2 \geq 0 \end{cases} \tag{P13}$$

where

$$\begin{aligned} f_1(x) &= (x_1-3)^4+(x_2-2)^4 \\ f_2(x) &= (x_1-4)^2+(x_2-3) \\ f_3(x) &= 3x_1^3+(x_2-1)^4 \end{aligned}$$

Assume that, according to the decision maker's preference, we have

$$f_1(x) \gg f_2(x) \gg f_3(x)$$

Let us now apply the lexicographic goal programming method for (P13). The first step consists of solving the following program:

$$\begin{cases} \text{Min } [(x_1-3)^4+(x_2-2)^4] \\ \text{subject to} \\ x_1^2+x_2^2 \leq 9 \\ x_2-x_1x_2^3 \geq 0 \\ x_1 \geq 0,50 \\ x_2 \geq 0 \end{cases} \tag{P14}$$

This program has a unique solution for $x^* = (1,083; 2,003)$.

2.4 A posteriori methods

A posteriori methods are concerned with finding all or most of the Pareto optimal solutions for a given multi-objective program. These solutions are then presented to the decision maker who has to choose one of them. The most important a posteriori methods described in the literature include the e-constraint method [25, 31], the adaptive search method [48], the hybrid method [15, 25], the Benson's method [22, 8] and the weighting method [29, 52]. In the following sections we restrict ourselves to the weighting method, which will be incorporated in our Decision Support System.

2.4.1 Weighting method

Consider the program (P). The weighting method transforms the original multi-objective problem into a single objective one. This is done by creating a new objective from the weighted sum of the k objectives. By doing so, the resulting program is:

$$\begin{cases} \text{Min} \quad F(x) \\ \text{subject to} \\ x \in X \end{cases} \tag{P15}$$

where $F(x) = \sum_{i=1}^{k} w_i f_i(x)$; $0 \leq w_i \leq 1, i, \ldots, k$ and w_i is the weight of the i-th objective of (P).

The following interesting result tells us that a solution for (P15) is Pareto optimal for (P).

Theorem 2.4.1 *If, $w_i > 0$ for all $i = 1, \ldots, k$, and if x^* is an optimal solution of (P15) then x^* is a Pareto optimal solution for (P).*

Proof
Let $x^* \in X$ be a solution for the weighting problem (P15) suppose that x^* is not Pareto optimal for (P). Therefore, there exists a solution $x \in X$ such that

$$f_i(x) \leq f_i(x^*) \ ; \ i = 1, \ldots, k$$

and

$$f_j(x) \ < \ f_j(x^*) \text{ for some } j.$$

From the fact that $w_i > 0 \ ; \ i \ = \ 1, \ldots, k$, it follows that

$$\sum_{i=1}^{k} w_i f_i(x) < \sum_{i=1}^{k} w_i f_i(x^*).$$

This contradicts the assumption that x^* is a solution for the weighting problem. Therefore, x^* is Pareto optimal for (P).■

Example 2.4.1

Consider again problem (P4) discussed in subsection 2.2.1:

$$\begin{cases} \text{Min } \left[\left(\dfrac{x_1+2x_2}{x_1+x_2+1}\right), \left(\dfrac{2x_1+x_2}{2x_1+3x_2+1}\right)\right] \\ \text{subject to} \\ -x_1+2x_2 \leq 3 \\ 2x_1-x_2 \leq 3 \\ x_1+x_2 \geq 3 \\ 2 \leq x_1 \leq 25 \\ 1 \leq x_2 \leq 9,5 \end{cases} \tag{P16}$$

In transforming (P16) using weight w_i; $i = 1, 2$, we get

$$\begin{cases} \text{Min} \displaystyle\sum_{i=1}^{2} w_i f_i(x) \\ \text{subject to} \\ -x_1+2x_2 \leq 3 \\ 2x_1-x_2 \leq 3 \\ x_1+x_2 \geq 3 \\ 2 \leq x_1 \leq 25 \\ 1 \leq x_2 \leq 9,5 \end{cases} \tag{P17}$$

where

$$f_1(x) = \frac{x_1+2x_2}{x_1+x_2+1}$$

$$f_2(x) = \frac{2x_1+x_2}{2x_1+3x_2+1}$$

A set of solutions for (P17) obtained by the parametric programming method is

given in table 2.1, where

$$F(x) = w_1 \left(\frac{x_1 + 2x_2}{x_1 + x_2 + 1} \right) + w_2 \left(\frac{2x_1 + x_2}{2x_1 + 3x_2 + 1} \right).$$

.

Table 2.1: A set of solutions for (P17)

w_1	w_2	x_1	x_2	$F(x)$
0	1	2	1	0,63
0,1	0,9	2	1	0,66
0,2	0,8	3	3	0,71
0,3	0,7	3	3	0,78
0,4	0,6	3	3	0,85
0,5	0,5	3	3	0,92
0,6	0,4	3	3	1,00
0,7	0,3	3	3	1,07
0,8	0,2	3	3	1,14
0,9	0,1	3	3	1,21
1	0	3	3	1,29

Since the objectives are to be minimised, the most preferred alternative is given by $w_1 = 0$, $w_2 = 1$ and $x_1 = 2$, $x_2 = 1$, which yield the lowest value of $F(x) = 0,63$. We obtain the same solution we obtained by the CP method. This solution is Pareto optimal for (P), because w_1 and w_2 are positive.

It is worth mentioning that for the weighting method, unlike in the goal programming method, the decision maker does not have to provide targets for each objective. Pareto optimal solutions for (P) are guaranteed if the weights are strictly positive. The weights are not easy for the decision makers to interpret or understand. It is also sometimes difficult for decision makers to choose a solution from a large number of generated alternatives.

2.5 Interactive methods

2.5.1 General philosophy of interactive methods

Interactive methods for solving a multi-objective programming problems consist of the following steps:

1 Find an initial solution.

2 Discuss the solution with the decision maker. If the decision maker is satisfied, stop. Otherwise go to the next step, after the decision maker has provided new targets for the objectives.

3 Obtain a new solution and go back to step 2.

Many interactive methods have been developed in the literature. Here are some of them. The STEP method [6, 27, 48], sequential proxy optimisation technique (SPOT) [11], the interactive surrogate worth trade-off (ISWT) method [48], the Geoffrion-Dyer-Feinberg (GDF) method [28, 48], the reference point (RP) method [32] and the Nondifferentiable Interactive Multi-objective BUndle-based optimisation System (NIMBUS) [50].

2.5.2 The reference point method

As an example of interactive method we describe below the reference point method.

1. Analysis

Before describing this method, we need the following definition.

Definition 2.5.1 *An* achievement function *for (P) is defined as follows:*

$$s_{\bar{z}}(z) = \max_{i=1,\ldots,k} [w_i(z_i - \bar{z}_i)]$$

where $\bar{z} \in \mathbb{R}^k$ *is an arbitrary reference point and* $w_i \quad (i = 1, \ldots, k)$; $w_i \geq 0$ *and* $\sum_{i=1}^{k} w_i = 1$.

Moreover the following program will also be needed:

$$\begin{cases} \text{Min } s_{\bar{z}}(z) \\ \text{subject to} \\ z \in Z \end{cases} \tag{P18}$$

where $Z = \{(f_1(x), \ldots, f_k(x)) |\ x \in X\}$.
As (P18) involves an achievement function, it is called an achievement problem. The aim of the reference method is to minimise the related achievement function. Remember that $s_{\bar{z}}(z)$ is the achievement function for program (P). The decision maker is requested to assist in the choice of the next reference point at each iteration.

2. Description of the method

Step 1 Present information about the problem to the decision maker. Set $h = 1$.

Step 2 Ask the decision maker to specify the reference point $\bar{z}^h \in \mathbb{R}^k$.

Step 3 Solve the mathematical program (P18), where $\bar{z} = \bar{z}^h$ and obtain the solution z^h with its corresponding x^h. Present x^h to the decision maker.

Step 4 If the decision maker is satisfied with this solution stop. x^h is the final solution. Otherwise go to step 5.

Step 5 Set $h = h + 1$ and go to step 2.

The reference point method is easy to understand and to implement and the decision maker is free to change his mind during the solution process. Unlike the goal programming method, the reference point method guarantees the Pareto optimality of the solutions for (P), depending on the increasing nature of the achievement function deployed. Some shortcomings of the reference point method include the fact that there are no criteria for the choice of aspiration levels and

for the elicitation of the achievement function, and there is no clear strategy for producing the final solution since the reference point method does not help the decision maker to find improved solutions.

2.5.3 The NIMBUS method

The Nondifferentiable Interactive Multi-objective BUndle-based optimisation System (NIMBUS) method has been designed to handle nondifferentiable functions of (P) efficiently. The decision maker is shown the value of the objective functions of (P) at solution x^h at iteration h. The decision maker then indicates desirable changes to be made in $f(x^h)$. This is done by dividing the objective functions into five classes as follows:

- the class of objectives which are allowed to change freely ($i \in I^{\circ}$)
- the class of objectives which are allowed to increase up to a certain upper bound ($i \in I^{>}$)
- the class of objectives which are satisfactory at the moment ($i \in I^{=}$)
- the class of objectives which should be decreased ($i \in I^{<}$)
- the class of objectives which should be decreased down until some target level ($i \in I^{\leq}$) is reached

For details of this method, the reader may consult [48].

Example 2.5.1

Consider the following multi-objective program:

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x), f_7(x)] \\ \text{subject to} \\ 0.0332x_1 + 0.0186x_2^2 \geq 3.5 \\ 0.1x_1 + 0.92x_2 \geq 6.3 \\ 0.03x_1 + 0.02x_2 + \geq 0.3 \end{cases} \tag{P19}$$

where

$$\begin{aligned} f_1(x) &= 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ f_2(x) &= -[x_1^2 + (x_2 - 1)^2 + x_2 - 1, -x_1^2 - (x_2 - 1)^2 +_2 +1] \\ f_3(x) &= -[x_1^2 + x_2^4, (2 - x_1)^2 + (2 - x_2)^2, 2e^{-x_1+x_2}] \\ f_4(x) &= -[5x_1 + x_2, -5x_1 + x_2, x_1^2 + x_1^2 + 4x_2] \\ f_5(x) &= -[x_1^2 + x_2^2, x_1^2 + x_2^2 + 10(-4x_1 - x_2 + 4), x_1^2 + x_2^2 - 10(x_1 + 2x_2 - 6)] \\ f_6(x) &= -[-x_1 - x_2, -x_1 - x_2, -x_1 - x_2 + (x_1^2 + x_2^2 - 1)] \\ f_7(x) &= x_1 - 20[x_1^2 + x_2^2 - 1] \end{aligned}$$

The starting point has been set by the decision maker as $x^\circ = (5.0, 5.0)$. The corresponding criterion vector is $f(x^\circ) = (40016.0, 45, 0, 650.0, 70.0, 50.0, 39.0, 975.0)$. When all the objective functions are minimised at the first iteration ($I^< = \{1, 2, 3, 4, 5, 6, 7\}$), we obtain $f(x^1) = (222.26, 5.27, 17.72, 14.99, 9.8, 2.42, 122.39)$. In the first classification we set $I^< = \{1, 7\}$ with the weighting coefficients $w_1 = 0.5$ and $w_2 = 0.3$ and $I^\leq = \{3, 4\}$ with aspiration levels $\overline{f}_3 = 10.0$ and $\overline{f}_4 = 10$ and the weighting coefficients $w_3 = 0.1$ and $w_4 = 0.1$. In addition, $I^> = \{2\}$ with $\varepsilon_2 = 10$ and $I^\diamond = \{5, 6\}$. The criterion vector obtained is $(5.08, 0.43, 3.97, 4.39, 44.63, -1.21, -0.8)$. Therefore, we continue from $(f(x) = 0.57, 0.91, 2.31, 5.76, 34.4, -1.13, 13.52)$. The next classification is $I^< = \{5\}$, $I^\leq = \{7\}$ with $\overline{f}_7 = 10.0$ without any weighting and $I^> = \{1, 2, 3, 4, 6\}$ with $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = \varepsilon_6 = 10$. The resulting criterion vector is $f(x^3) = (10.0, 1.32, 2.38, 5.7, 33.36, -1.14, 12.99)$ and this is accepted for continuation. The following classification is $I^\leq = \{7\}$ with $\overline{f}_7 = 10.0$, $I^= = \{1\}$, $I^> = \{5\}$ and $\varepsilon_5 = 35.0$ and $I^\diamond = \{2, 3, 4, 6\}$. The criterion vector obtained is $f(x^4) = (0.92, 1.31, 2.55, 5.33, 34.57, -1.21, 10.0)$ and this is accepted as the final solution. The corresponding variable is $x^4 = (0.8, 0.95)$, which is Pareto optimal for (P).

The strong points of the NIMBUS method are that Pareto optimal solutions for (P) can be obtained and it is able to handle nondifferentiable functions.
The drawbacks of the NIMBUS method are that it depends on other methods such as the multi-objective proximal bundle method to solve (P). It also transforms (P) into at least five single objective problems, which is confusing for the decision maker, who may find it difficult to classify the objective functions.

2.6 Meta-heuristic methods for the multi-objective programming problem

Most of the methods described in the preceding sections apply for convex multi-objective programs. In the case of nonconvex multi-objective programs meta-heuristic methods may be considered [10, 9]. A meta-heuristic is a method that seeks to find a satisfying solution to a problem at a reasonable computational cost. A meta-heuristic often has an intuitive justification and therefore a mathematical proof cannot be constructed to guarantee the Pareto optimality of the solution found [10]. The most used meta-heuristic are simulated annealing [24], tabu search [46] and genetic algorithm (GA) [10]. In what follows we restrict ourselves to the genetic algorithm method, which will be incorporated in our Decision Support System.

Genetic algorithm

A genetic algorithm is a stochastic search method for problems based on the mechanisms of natural selection and natural genetics (that is, survival of the fittest) [42]. To be able to solve program (P) using a genetic algorithm, it is important to first convert it into a single objective program. One of the most important notions in the genetic algorithm is that of chromosomes. A chromosome is a string of numbers or symbols. The genetic algorithm starts with an initial set of randomly generated chromosomes which are called a population. The population size is the number of individuals in that population. All chromosomes are evaluated by an evaluation function and the selection process is used to form a new population, which uses a sampling mechanism based on fitness values. The term "generation" is used to describe the cycle from one population to another.

The crossover and mutation operations are used to update all chromosomes and the new chromosomes are called offspring. The new population is formed when the selection process selects new chromosomes. After a given number of generations, the best chromosome is decoded into a solution for (P). A genetic algorithm usually follows the following steps:

Step 1 Initialise chromosomes at random.

Step 2 Update the chromosomes by crossover and mutation operations.

Step 3 Calculate the objective values of all the chromosomes.

Step 4 Compute the fitness of each chromosome via the objective values.

Step 5 Select the best chromosome using the tournament selection method, ranking selection method or roulette wheel method.

Step 6 Repeat the step 2 to step 5 the fifth steps for a given number of cycles.

Step 7 Report the best chromosome as the compromise solution.

Example 2.6.1

Consider the multi-objective program

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x)] \\ \text{subject to} \\ -x_1^2 + x_2^2 \geq 10 \\ x_1^2 + x_2^2 + x_3^2 \leq 6.3 \\ x_1, x_2, x_3 \geq 0 \end{cases} \tag{P20}$$

where

$$\begin{aligned} f_1(x) &= 3 - \sqrt{x_1} \\ f_2(x) &= 4 - \sqrt{x_1 + 2x_2} \\ f_3(x) &= 5 - \sqrt{x_1 + x_2 x_2 + 3x_3} \end{aligned}$$

Using a genetic algorithm to solve this program leads to the solution of

$x = \{x_1, x_2, x_3\} = \{9, 3.5, 2.597\}$.

The advantage of the genetic algorithm is that it can be used to solve non-convex multi-objective programs. Unfortunately, it cannot guarantee the Pareto optimality of the solution obtained.

Chapter 3

The Decision Support System paradigm

3.1 Introduction to Decision Support Systems

A Decision Support System (DSS) is a computer-based system that helps decision makers to confront ill-structured problems through direct interaction with data and analysis models [13]. In [60], a framework for thinking about and developing DSS is provided. A DSS should have the following characteristics:

- Ability to support interaction with non-expert users, it should have access to a *wide variety of data*, and it should provide *analysis and modelling* in a variety of ways.
- Capability to evolve and adapt as the situation changes.
- Presence of a set of data sources as well as possibility of interactions with different people.

In the 1980s, important new developments appeared on the decision support scene. One of these was expert systems (ES), which capture the experience, knowledge, training, and judgement of a human expert in a computer program. Since then, work on ES has been confined to high-profile research laboratories conducted by highly skilled professionals, supported by multimillion dollar budgets

and specialised hardware and software, on "rocket scientist" types of application. It is only recently that ES have entered the mainstream of computer applications. In the late 1980s, another development emerged called the executive information systems (EIS) [63]. This was designed to support the information needs of a firm's senior executives. The development of EIS was influenced by a number of factors, including a growing understanding of the nature of executives and executive work, an increasing number of EIS success stories, and advances in hardware and especially in software.
Given the large amount of time that people spend working in groups, it is easy to see why systems that have the potential for improving group efficiency and effectiveness are attracting considerable attention.

3.2 Architecture of a Decision Support System

The general architecture of a DSS (see figure 3.1) consists of database, a modelbase and a software system. In the following section we briefly discuss each of these components.

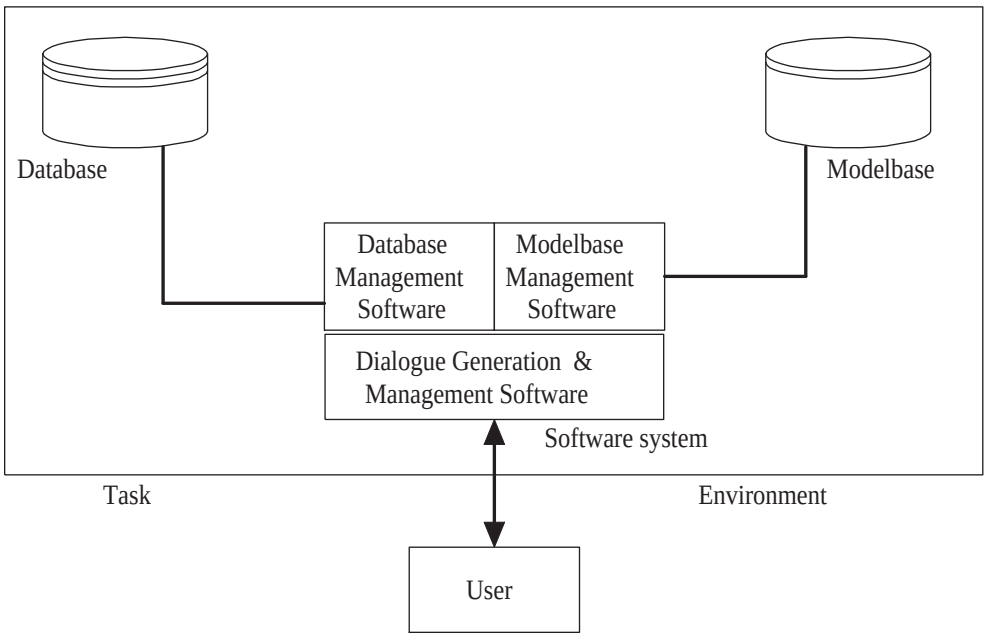


Figure 3.1: The general architecture of a decision support system

3.2.1 Database

Data from internal and external sources are stored in the database subsystem. The database management software must allow for rapid retrieval of data as well as additions and changes in response to user requests.
A DSS database should combine a variety of data sources quickly and easily. It should also be able add and delete data sources quickly and update, rearrange or respond to inquiries and retrieval of data. It should portray logical data structures in user terms and manage a wide variety of data with a full range of data management functions.
These features should be in accordance with the modelbase management software which is discussed next.
Figure 3.2 below shows an example of a database for a DSS.

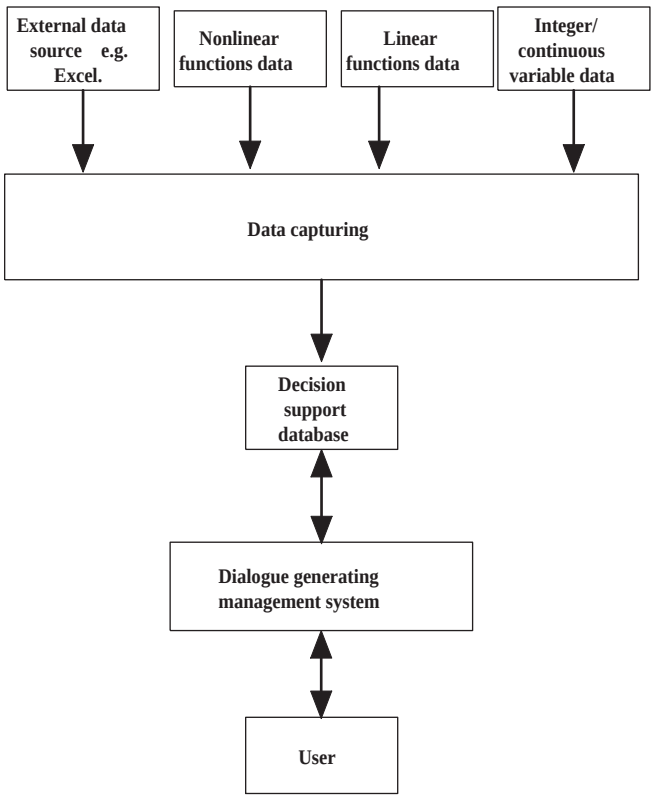


Figure 3.2: Components of a DSS database subsystem.

3.2.2 Modelbase

The modelbase subsystem consists of a collection of separate models. Each model deals with a specific part of a given problem. The modelbase subsystem should be able to

- create new models,
- catalogue a wide range of models,
- support all levels of decision making and
- manage modelbases with management function analogues.

For more details on database and model subsystems of a DSS the reader may consult [61].
For the database management system and the modelbase management system to communicate properly, a dialogue generating management system is needed. We discuss this subsystem in the next section. Figure 3.3 shows an example of the way the modelbase of a DSS that handles optimisation problems can be structured.

3.2.3 Dialogue generating management system

The dialogue generating management system consists of three parts, that is, action language, display language and a knowledge base. The action language refers to what the user can do to communicate with the computer. This can be done by using the keyboard, the mouse or the voice commands. The display language refers to what the user can read on the screen, which includes graphs and colour of display.

In Dos Santos and Holsapple's paper [23], they present a framework to facilitate the understanding and development of adaptive, easy-to-use interfaces of a DSS. They highlight the following aspects pointing out the important role of the user in the development of a DSS.

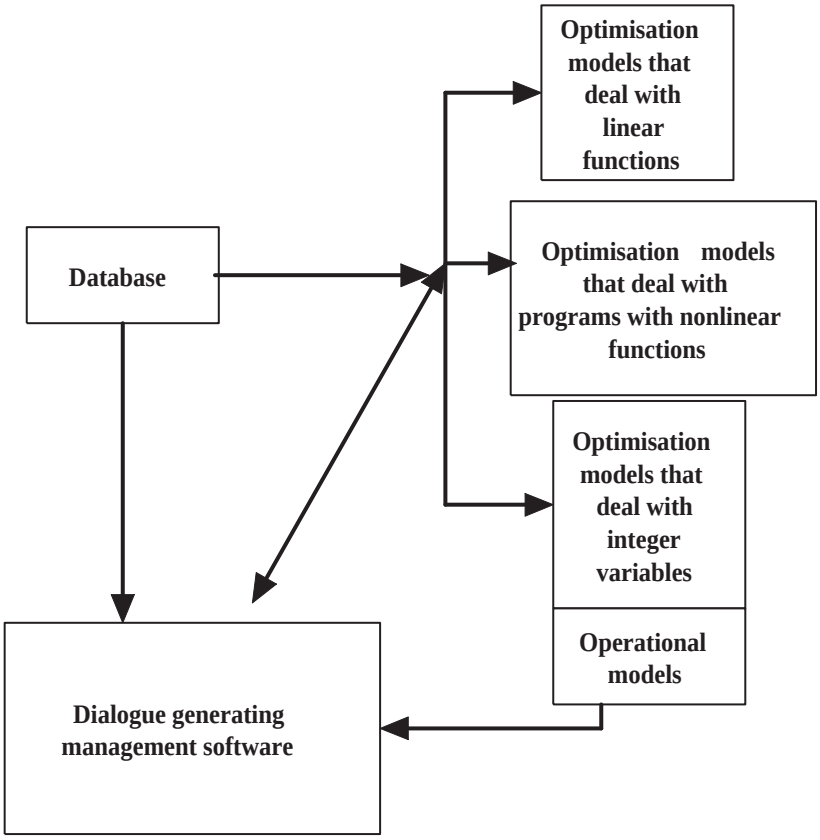


Figure 3.3: Structure of a modelbase of a DSS for optimisation

- A user of a DSS should be viewed as an active entity.
- Problem solving within a DSS should be an interactive process between the user and the computer.

3.3 Technology levels for a Decision Support System

Three levels of hardware/software which have been included in the label "DSS" will be discussed in the following subsections.

3.3.1 Specific Decision Support System

A *Specific DSS* is an application of DSS methodology to a particular decision problem. As an example, consider the portfolio management system described by [39]. Another example is the police beat allocation system used on an experimental basis by the City of San Jose, California [62]. In the latter example the system allowed a police officer to display a map outline and see most of the activities in that area. The interactive graphic capabilities of the system allowed the police officer to choose alternatives quickly.

3.3.2 Decision Support System generator

A *DSS generator* is a "set" of related hardware and software which provides the capabilities for creating a specific DSS. For example, the police beat system mentioned above was built from the geodata analysis and display system (GADS) [7]. By loading different maps, data, menu choices and procedures or command strings, GADS was later used to build a specific DSS to support the routing of IBM copier repairmen [62].

3.3.3 Decision support system tools

DSS tools is the third and the most fundamental level of technology applied to the development of a DSS. These are hardware or software elements which facilitate the development of a specific DSS or DSS generator. This category of technology has seen the greatest amount of recent development, including new special-purpose languages, improvements in operating systems to support conversational approaches, colour graphics hardware and supporting software, and so forth. For example, the GADS system described above was written in FORTRAN [16] using an experimental graphics subroutine package as the primary dialogue handling software, a laboratory-enhanced rasterscan colour monitor, and a powerful interactive data extraction/database management system.

3.3.4 Relationships between the three technology levels

As shown in figure 3.4., the DSS tools can be used to develop a specific DSS application directly. They can also help in developing a DSS generator. Both DSS tools and DSS generators must be used to build specific applications.

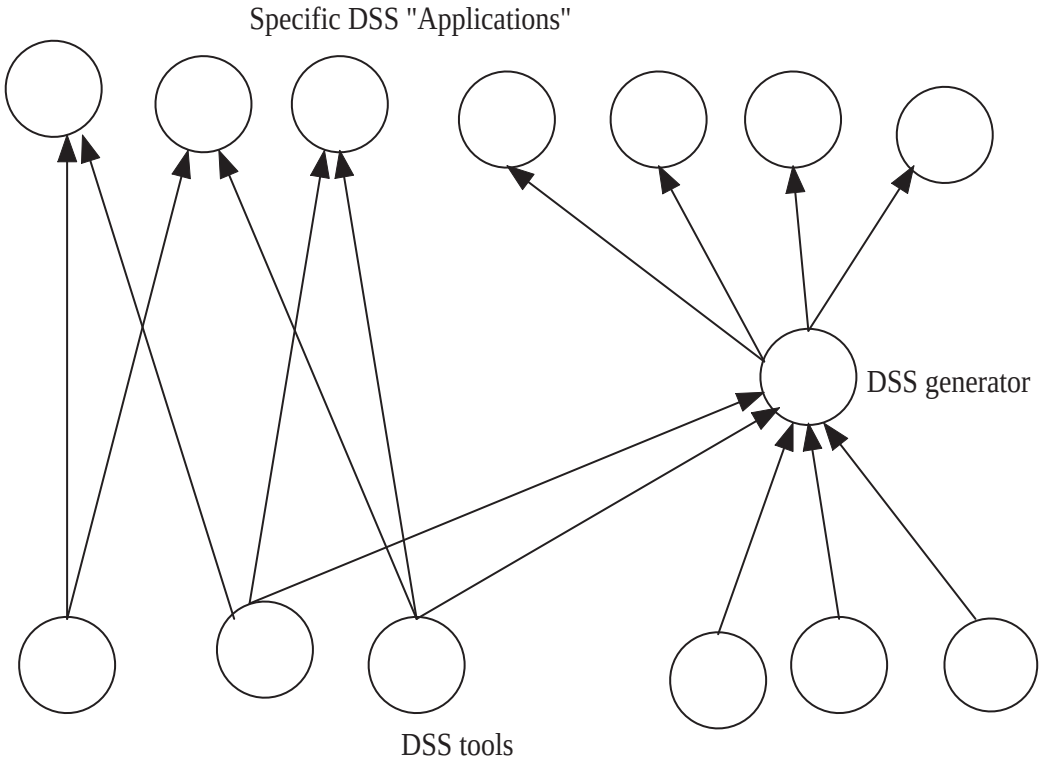


Figure 3.4: DSS technology levels

3.4 Roles in the Decision Support System development process

There are five main actors in the DSS development process: the manager or decision maker, the intermediary, the DSS builder, the technical supporter and the toolsmith. The level of their intervention is shown in figure 3.5.

A brief description of the role of each actor is given below:

- The *manager or decision maker* is the person who formulates the decision

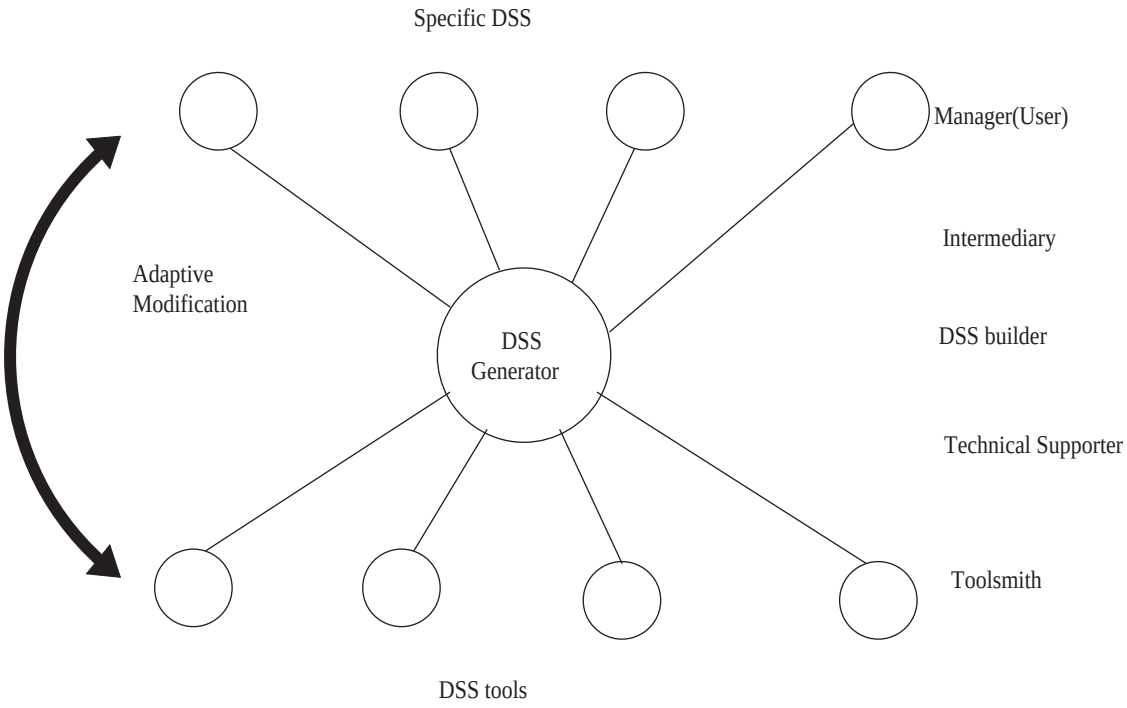


Figure 3.5: Levels of intervention and of actors involved in a DSS development process

problem.

- The *intermediary* is the person who helps the decision maker, perhaps merely as an analyst to make the necessary calculations.
- The *DSS builder* develops the necessary capabilities from the DSS generator to "configure" the specific DSS. This person must have some knowledge of the problem area. He should also be comfortable with the information system technology components and capabilities.
- The *technical supporter* develops additional information system capabilities when there is a need. New databases, new analysis models and additional data display formats will be developed by the technical supporter. These require strong computer literacy.
- The *toolsmith* develops new technology and new languages, new hardware and software in order to improve the efficiency of linkages between subsystems.

Two observations about this spectrum of roles are appropriate. The first is that it is possible for one person to assume several roles and one role can be filled by several people. The second observation is that these roles appear similar to those present in traditional systems development, but there are subtle differences. The appropriate role assignment will generally depend on the nature of the problems, the nature of the person and the strength of the technology available.

3.5 Management involvement in the design of a Decision Support System

Based upon and extending beyond existing research and speculation in the literature, this work proposes that the involvement of management in the design of a DSS separates logically into four areas. The manager, in relation to the DSS, serves as an approver, administrator, developer, operator and user of output. As a result of these four roles, the decision maker is proposed as being a logical candidate for the management task with regard to DSS. Much of the subclassification of issues for the "developer" role and the "user of output" role is derived from the work of Sprague [60]. In the following subsections, issues relevant to management's involvement in DSS with regards to the approver, developer, operator and user roles will be presented, along with supporting evidence from the literature.

3.5.1 Approver and administrator

In general it is the responsibility of the decision maker to judge the relative merits of alternative organisational investments and to accept these investments when it is to the benefit of the organisation. In addition, DSS become an additional organisational entity requiring a position within the organisation's structure and requiring the forming of relationships with other interacting organisational units. In terms of the approver role, there are two basic issues to be considered: Firstly, who determines the company's DSS policy and how is it done?

- According to Carlson and Sprague [13], DSS planning should be incorporated into corporate planning.

- Moreover, in planning to implement a DSS into an organisation, the political impact on the power base should be examined [1].

Secondly, how does the decision maker evaluate a DSS in terms of its gain/loss to the company?

- The financial evaluation of a DSS is difficult and is not done frequently. See [2, 36, 47] for details of these claims.
- DSS evaluation should be based upon both value-addition and cost-reduction [36, 38].
- A "portfolio" approach should be used which considers risk and reward as suggested by Gremillion and Pyburn [30].

3.5.2 Developer

Given that the function of a DSS is to support the decision maker's decision-making responsibilities, and given that decision making is a difficult task to specify or structure, it should not seem illogical that the decision maker would need to play a substantial role in the DSS development process. There are four concerns relevant to the decision maker's role as DSS developer.

Firstly, at what points in the DSS development process is decision maker involvement required? Secondly, how much time is required of management in developing the DSS? Thirdly, how are the personal needs and style of the manager/decision maker incorporated into the DSS?
Finally, what technology is used by managers in developing the DSS?
With regard to the first issue we may point out the following:

- Management involvement should take place throughout development. See for instance Alter [3].
- Management should take the lead in DSS development [39].
- DSS development is often promoted and fostered by the advocacy of an "organisational champion". This is explained in Curley and Gremillion [20].

- DSS development should include a fostering of cooperation and coordination between user/managers and technical designers [1].
- User-led development approaches often lead to an improved DSS [35].

As far as the second issue is concerned, we may point out the following:

- Total development time is short (1 day to 20 weeks). See for instance Keen [36].

As regards the last two points we may indicate the following:

- The DSS should be developed to include the personal decision-making style of the manager [36, 47].
- The nominal level of available DSS technology has a significant impact on the extent of user and manager involvement [45].

3.5.3 Operator of a Decision Support System

The actual operator of the DSS requires skills that are very dissimilar to those typically required of managers.
There are different "levels" of technical sophistication in a DSS which require different levels of ability for use. It is expected that some level of operational ability and actual operation of the DSS may be desired by managers.
In terms of managerial interest in the system operation, how much will management generally be required to operate the DSS? In response one might speculate that, managers may prefer to turn the operation of the DSS over to their staff [65] otherwise there may be a significant number of senior managers sitting at terminals [37].

3.5.4 User of a Decision Support System's output

In making decisions, managers typically make use of a variety of sources of information. The whole purpose of a DSS is to support the manager in his decision-making process by attempting to supply the information the managers request.

In the final analysis, management is most interested in the outputs of a DSS.
The last of the four roles, namely user of output, is concerned with three aspects.
Firstly, how is DSS output utilised by management vertically throughout the organisation?
Secondly, does the DSS support managers in both individual and group decision making?
Thirdly, how does the DSS support management in the different phases of decision making?
What follows is a summary of the responses to these questions

- The DSS should be used in support of managerial decision making at all levels of the organisation [39].
- The DSS should support both individual and group ("sequential interdependent" and "pooled interdependent") decision making [17].
- The DSS should be able to assist the manager during all the phases of the decision making process [39].

3.6 Examples of Decision Support Systems

In the following section we list some existing Decision Support Systems along with their characteristics.

- Microsoft Windows-based Group Decision Support Systems (WINGDSS). This system allows a group of decision makers to work on the same project at the same time. Individual decision making is also possible. More information about this DSS may be found in Csaki, Kapsak, Turchanyi and Vermes [18].
- Representation and Maintenance of Process knowledge (REMAP) [56] is a DSS for companies that supply services for residential and industrial clients. The types of services it can handle include electric power outages, appliance repairs and changes of service location. The main aim of this DSS is to respond promptly to processing requests from clients and to give accurate information about the status of their requests for service.

- Dynamic Interactive Network Analysis System (DINAS) [54] is able to solve multi-objective programming problems with up to seven objective functions. It is able to solve a transportation network problem with up to a hundred nodes and up to three hundred arcs. It consists of three programs which are combined by C programming language.

Chapter 4

DSS4MOPP: A Decision Support System for multi-objective programming problems

DSS4MOPP is a decision support system for multi-objective programming problems. The main aim of this decision support system is to help decision makers faced with problems that may be cast into a multi-objective programming framework to choose an appropriate technique for dealing with the problem at hand to solve it.
DSS4MOPP assumes that the multi-objective program to be solved has been completely formulated by the decision maker and that information concerning the decision makers preferences is available.

4.1 Components of DSS4MOPP

The DSS4MOPP has three main components, namely a database, a modelbase and a software system. In the following subsections we describe each of these components.

4.1.1 Database

The DSS4MOPP database is able to store a collection of data files. These data files contain a combination of numerical and alphabetical data. Although some of the data is stored directly in the computer, some of it may be stored on the internet. The files are protected by a security code activated by the decision maker.

4.1.2 Modelbase

The modelbase consists of the following methods: the compromise programming method, the genetic algorithm, the goal programming method, the lexicographic goal programming method, the method for generating efficient solutions, the multi-objective proximal bundle method, the NIMBUS method, the reference point method and the weighting method.
All these methods have been described in detail in chapter 2. The tools used to develop DSS4MOPP are Linear Interactive Discrete Optimiser (LINDO) [58], nondifferentiable interactive multi-objective bundle-based optimisation system (NIMBUS) [51] and Multi-Objective Programming ENvelopment (MOPEN) [12].

4.1.3 Software subsystem

The software subsystem of DSS4MOPP consists of three components: data base management software (DBMS), model base management software (MBMS) and Dialogue Generating Management Software (DGMS).
Through these components, the interface with the analyst is realised by a sequence of windows, with each window being regarded as a step. Each window has a distinct expression which helps considerably to facilitate the analyst's work. The analyst is able to move between the next window and the previous one, which helps him to make adjustments or corrections about information already entered into the computer. The solution process (which refers to the steps that the analyst follows to solve the problem) of a given multi-objective problem can be stopped at any step and can be continued again at a later stage. This allows the analyst to update, rearrange, retrieve and inquire about the objectives, constraints, variables or solution of the problem.

The model base management software of DSS4MOPP is able to manage the methods by choosing an appropriate one for a given problem. It should also be able to link the method to relevant data. The DSS4MOPP has good printing functions, which allow printing at any step. This is an advantage for the analyst who prefers to read on paper instead of the computer screen.
The dialogue generating management system of DSS4MOPP provides the dialogue between the analyst and the computer.The dialogue generating management system ensures that there is interaction between the DSS4MOPP, the analyst and the operating system. This interaction refers to entering data from the program (P) and information about the preferences of the decision maker. The interaction is also about the visualisation of the current solution on the screen. This is in connection with reading, storing files, and printing solutions. Figure 4.1 shows the first window of DSS4MOPP.

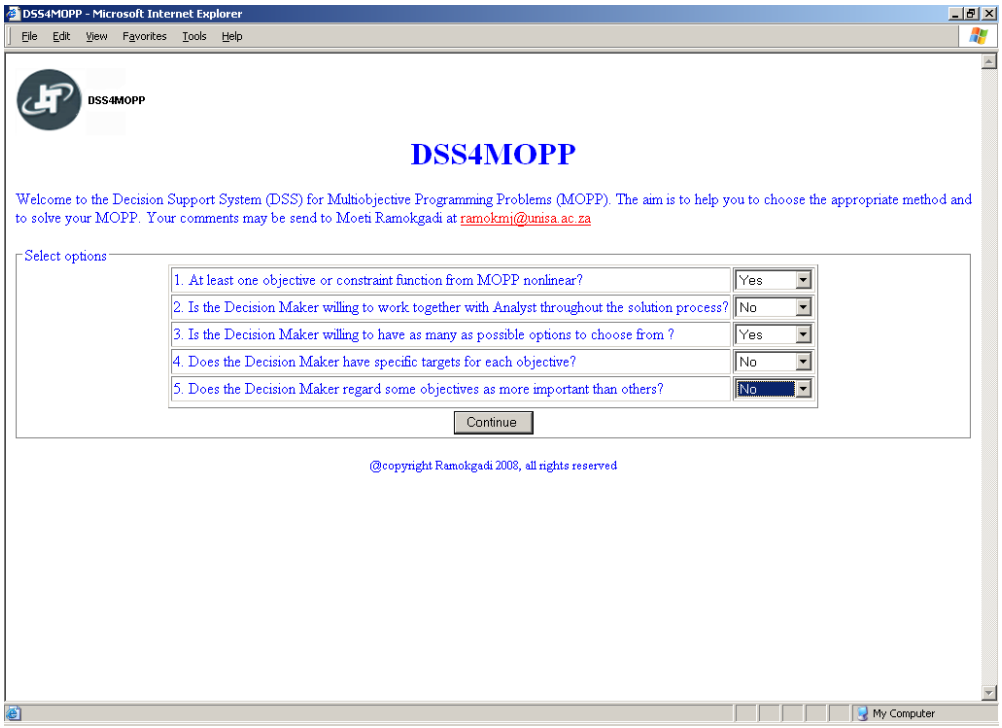


Figure 4.1: Form-based input page of DSS4MOPP

4.2 Functioning of DSS4MOPP

To solve a multi-objective program say (P), the following basic actions are performed:

- Analyst answers questions about the problem (P) and about preferences of the decision maker.
- Analyst enters data into DSS4MOPP.
- DSS4MOPP checks data for more details.
- DSS4MOPP solves (P) according to the method chosen.

When starting with the process, the analyst first has to answer the following questions with a "yes" or "no":

- Is at least one objective or constraint of the problem at hand nonlinear?
- Is the decision maker willing to work through the solution process together with the analyst?
- Is the decision maker willing to have as many options as possible to choose from?
- Does the decision maker have specific targets for each objective?
- Does the decision maker regard some objectives as more important than others?

The aim of these questions is to enable DSS4MOPP to choose a suitable method for solving the multi-objective program. DSS4MOPP makes a choice between the methods in the modelbase. Figures 4.2 and 4.3 show a flowchart that summarises the DSS4MOPP functioning.

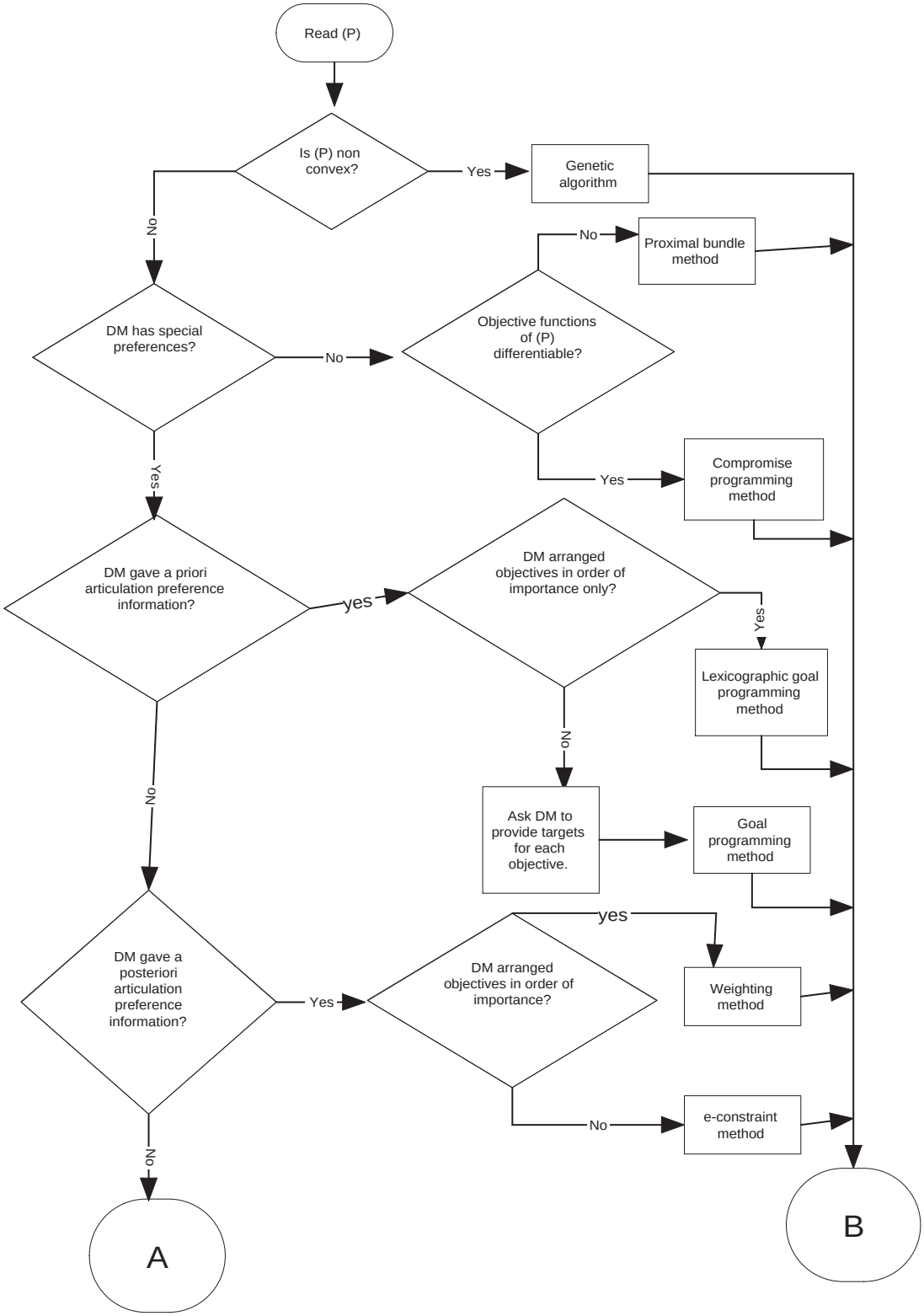


Figure 4.2: DSS4MOPP functioning flowchart

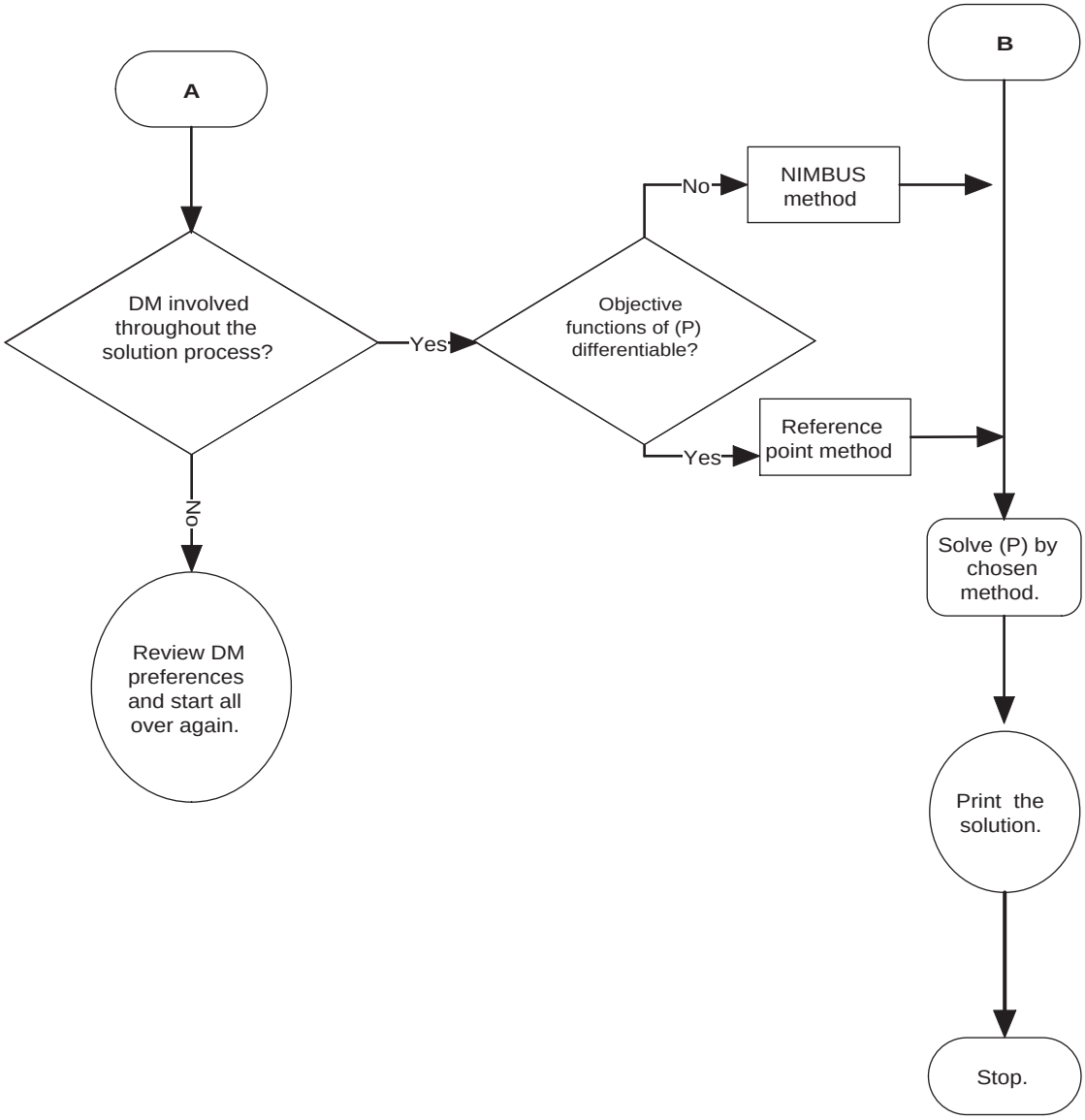


Figure 4.3: DSS4MOPP functioning flowchart continued

4.3 Implementation of DSS4MOPP

To build DSS4MOPP we used TextPad [59] program. TextPad is able to edit files up to the limits of virtual memory, and will work with MS Windows 9x, ME, NT 4, 2000 and XP.

The application is web-based and it is written using Hyper Text Markup Language (HTML). This means that this application can be deployed on the internet

and therefore be accessed anywhere as long as the user has access to the internet. HTML mainly presents the graphical user interface to the users and displays input controls such as text boxes, buttons and links. The look and feel of the interface design is implemented using a Cascading Style Sheet (CSS). Users can also easily customise the look and feel of the application as this is simple to implement.

A CSS is applied on HTML to do formatting of text and controls. Colours are also part of the CSS functionality. The business logic implementation is created using JavaScript, which means that the code is not hidden and the analyst may change the application if necessary to suit his needs. This is a programming language which can be embedded in the HTML files to perform functions such as extracting input from controls and manipulating data.

Using different control structures this will then decide according to the input what methodology is most appropriate. Subsequently the relevant application program will be launched.

It is worth mentioning that DSS4MOPP is able to deal with multi-objective programming problems that are nonconvex and nondifferentiable. Some of its tools are taken from the internet. Sensitivity analysis is also possible with DSS4MOPP. It also has standard Microsoft Windows applications menu and functions such as “File”, “Edit”, “View” , “Window” and “ Help”. These functions facilitate interaction between DSS4MOPP and the analyst.

DSS4MOPP has been designed to recognise files with extensions such as “*.ltf”, “*.lng”, “*.ltf” and “*.lg4”. If the user double clicks on a file with an extension, DSS4MOPP starts automatically.

Files saved on the internet server of DSS4MOPP do not have extensions and are accessed in the internet webpage only.

Every objective function and constraint of a multi-objective program to be solved is entered separately in their respective text field. Therefore, checking and correcting syntax error is simplified.

By double clicking on the objective function or constraint entered, the editing area where corrections are to be made is activated.

Different tools for solving problems in DSS4MOPP have been integrated, which provides the analyst with the potential to set the DM’s preferences for the most preferred solution.

The interface of DSS4MOPP facilitates the operation of the analyst by offering different alternatives.

The analyst is given the option to install the DSS4MOPP program on the computer or to work directly from the compact disk (CD) package.
The DSS4MOPP program is given in the appendix.

4.4 Illustrative example

The following example illustrate the way in which DSS4MOPP works. Consider the multi-objective program:

$$\begin{cases} \text{Min } [f_1(x), f_2(x), f_3(x), f_4(x), f_5(x)] \\ \text{subject to} \\ 12x_1 + 12x_2 - 15x_3 \leq 450 \\ 20x_1 + 25x_2 + 30x_3 + 35x_4 + 30x_5 \leq 600 \\ 230x_1 + 500x_5 \leq 1200 \\ 213x_3 \leq 250 \\ x_1^2 + x_2^2 + x_3^2 \leq 100 \\ x_3^2 + x_1^4 + x_1^5 \leq 200 \\ x_1^2 + x_1^2 = 100 \\ x_3^2 + x_4^2 = 100 \\ x_3^2 + x_4^2 + x_5^2 = 100 \end{cases}$$

and

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

where

$$\begin{aligned} f_1(x) &= 2x_1^2 + 2(x_2 + 5)^2 + (x_3 - 3)(x_3 - 4)(x_3 - 5,5) \\ f_2(x) &= (x_1 + 40)^2 + (x_2 - 224)^2 + (x_3 + 40)^2 \\ f_3(x) &= (x_1 - 224)^2 + (x_2 + 40)^2 + (x_3 + 40)^2 \\ f_4(x) &= (x_1 - 345)^2 + (x_2 + 40)^2 + (x_3 + 40)^2 \\ f_5(x) &= (x_1 - 224)^2 + (x_2 + 40)^2 + 300(x_3 + 40)^2 + 400(x_4 + 40)^2 + (x_5 + 400)^2 \end{aligned}$$

To start solving the problem, DSS4MOPP has to ask the analyst some questions.

In Figure 4.4, DSS4MOPP explains to the user that it has chosen the proximal bundle method. In Figure 4.5, DSS4MOPP selects the tool to use. If using it for the first time, the analyst will be requested to create a username and password. This is to enable him to protect it from unauthorised use and to retrieve it next time it is needed. In Figure 4.6, part of the multi-objective programming problem entered by the analyst is shown. The window also shows the tutorial and help options available to assist the analyst further. Figure 4.7 displays the solution obtained and the options given to the analyst. One of the options is that the analyst can generate another set of solutions if necessary.
The current solution is
$x_1 = 244626E-7$, $x_2 = 9.982534$, $x_3 = 1.459431E-6$, $x_4 = 9.155327$ and $x_5 = 1.000009$, with the objective values as
$f_1(x) = 50382.95$, $f_2(x) = 49003.48$, $f_3(x) = 54274.25$, $f_4(x) = 123123.3$ and $f_5(x) = 1659974$.

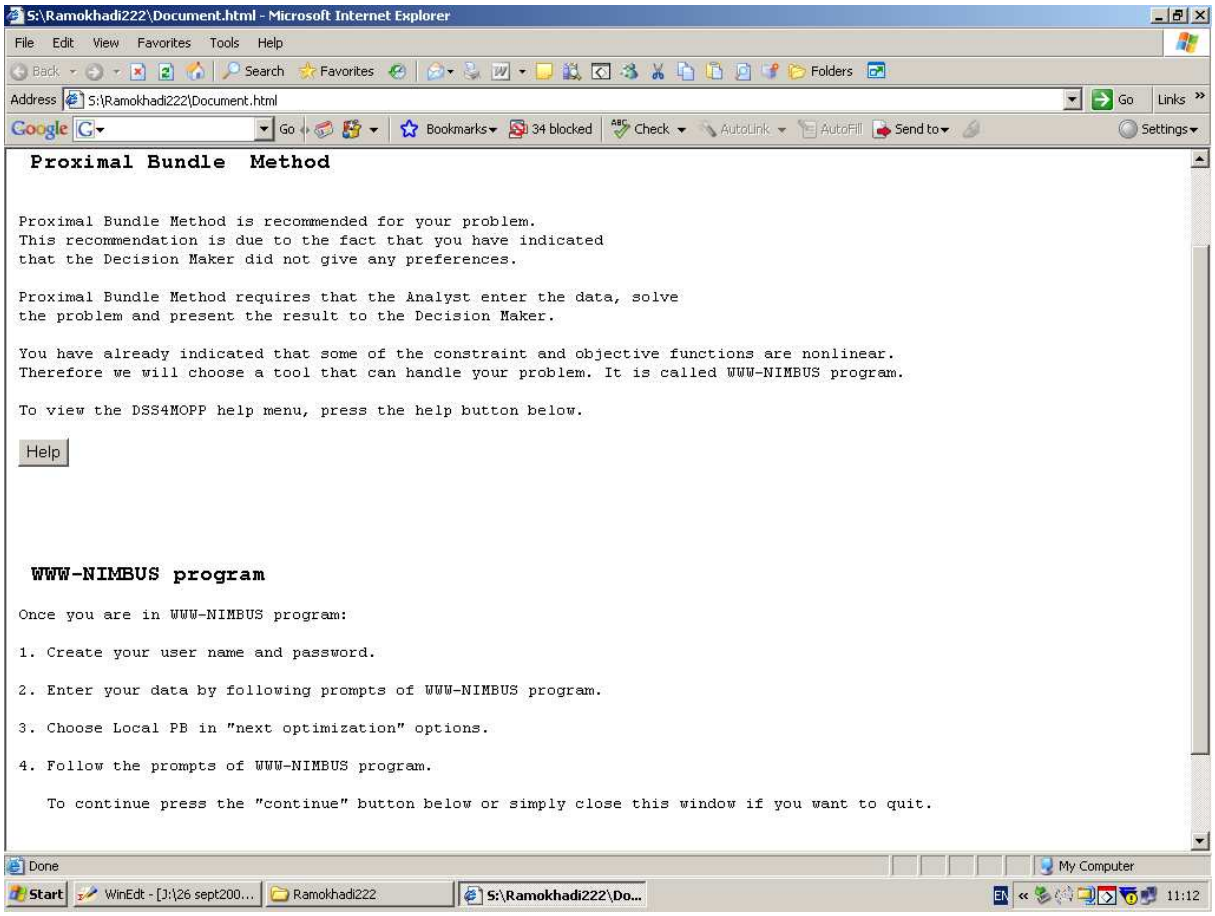


Figure 4.4: DSS4MOPP selects the proximal bundle method

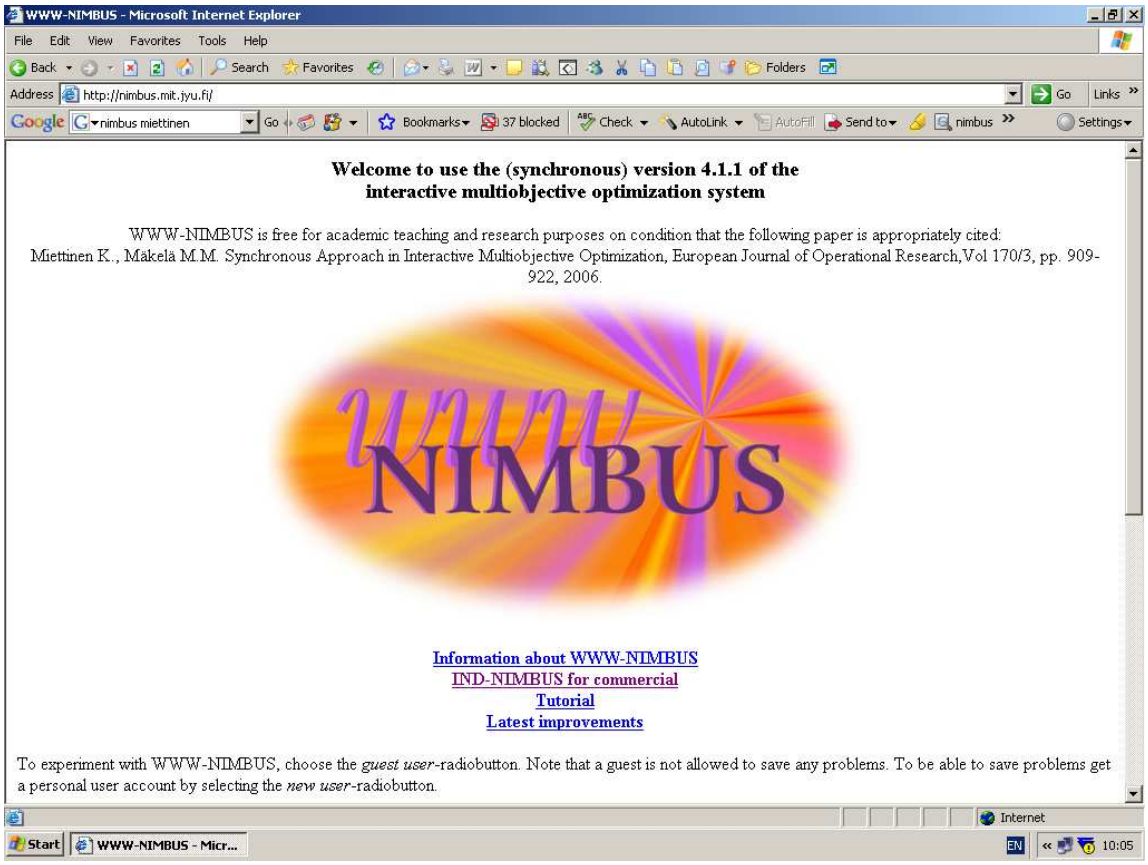


Figure 4.5: DSS4MOPP selects the tool to solve the problem

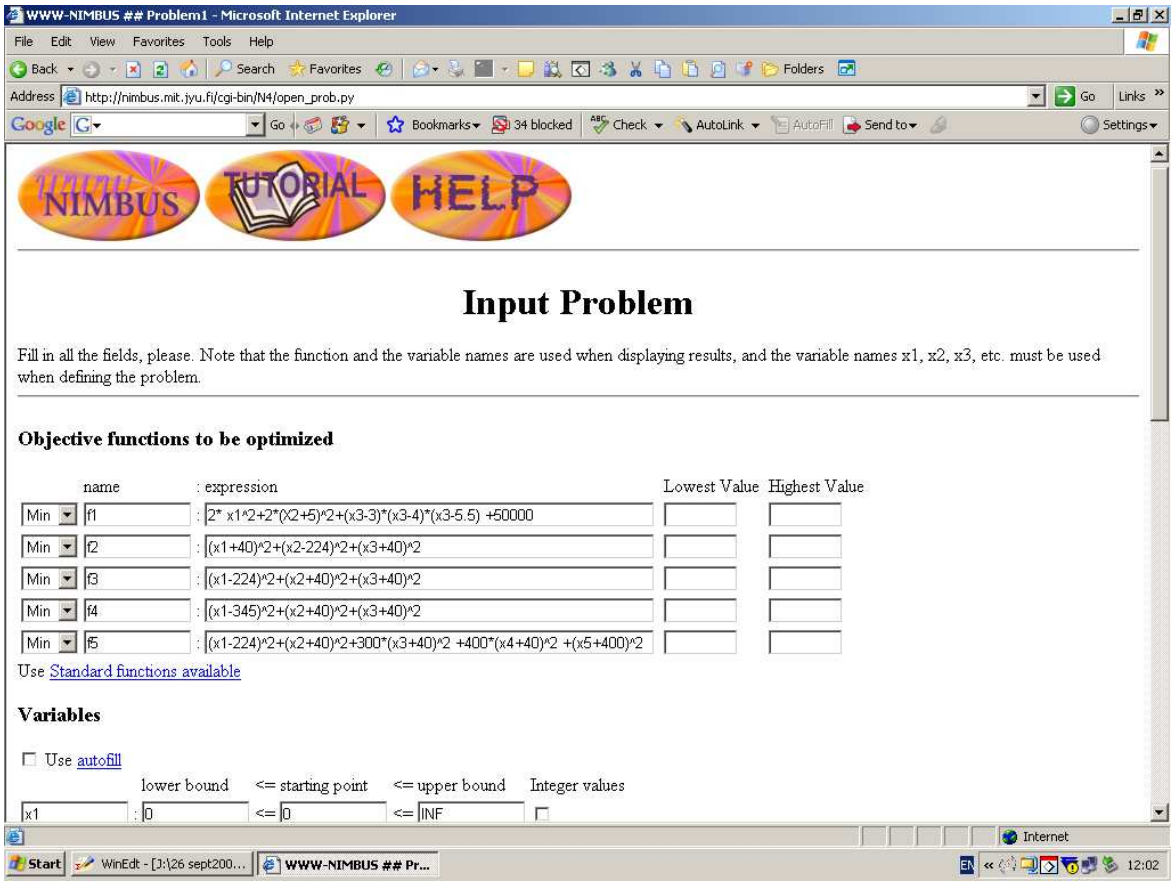
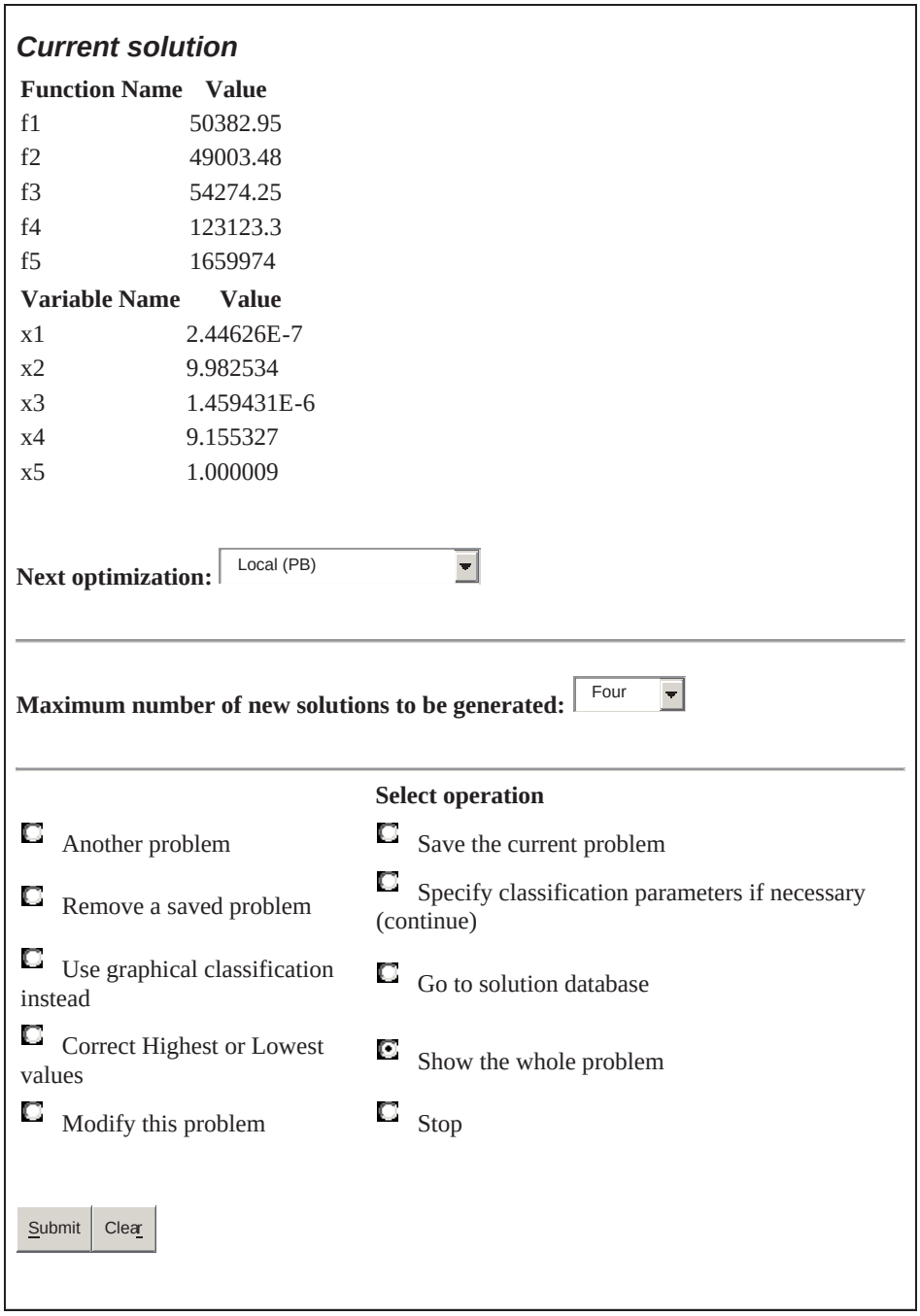


Figure 4.6: Data is entered in DSS4MOPP



Current solution

Function Name	Value
f1	50382.95
f2	49003.48
f3	54274.25
f4	123123.3
f5	1659974

Variable Name	Value
x1	2.44626E-7
x2	9.982534
x3	1.459431E-6
x4	9.155327
x5	1.000009

Next optimization: Local (PB)

Maximum number of new solutions to be generated: Four

Select operation

- ☐ Another problem
- ☐ Remove a saved problem
- ☐ Use graphical classification instead
- ☐ Correct Highest or Lowest values
- ☐ Modify this problem
- ☐ Save the current problem
- ☐ Specify classification parameters if necessary (continue)
- ☐ Go to solution database
- ◉ Show the whole problem
- ☐ Stop

Submit Clear

Figure 4.7: Printout of the solution from DSS4MOPP

Concluding remarks

Many concrete real-life situations may be cast in a mathematical programming framework. In most of these situations, one has to combine evidence from disparate sources and as a result grapple with conflicting objective functions.
Therefore, multi-objective mathematical programming is a relevant issue. Unfortunately, a multi-objective mathematical program is an ill-defined problem. As a matter of fact for this problem we do not have all ingredients needed to deal with all three analysis problems, namely the existence of a solution, the eventual unicity and the constriction of such a solution and the notion of "optimum optimorum" no longer applies.
In this dissertation we have discussed rational ways for optimum optimorum dealing with a multi-objective program having several conflicting goals. These include the notion of Pareto optimality, weak efficiency, proper efficiency, compromise solution and nadir's solution.
We have also presented several methods for finding such satisfying solutions in detail.
In the presence of such a plethora of techniques, a user may get lost. This is the reason why we have designed a Decision Support System able to help a decision maker choose the most appropriate tool for his problem. The system also singles out a satisfying solution in tune with the value system and the preferences of the decision maker.
The system, namely DSS4MOPP, it is also able to use tools from the internet like the NIMBUS program, it is able to display input controls such as text boxes, buttons and links to simplify communication with the user. Moreover, it can choose a method and a tool for solving a multi-objective programming problem and it gives the user two options. Either install it on the computer or work directly from the CD package. It can also be deployed on the internet and may be accessed anywhere provided the user has an internet browser such as Microsoft

Internet Explorer or Mozilla Firefox.
For the sake of illustration, we have carried out an example. Lines for further developments in this field include: an enrichment of the system by allowing it to help the decision maker throughout the entire decision-making process and use of language of Fuzzy sets theory to allow some leeways in the constraints satisfaction and in the multi-objective program at hand.

References

[1] T Ahn and G Grudnitski. Conceptual perspectives on key factors in DSS development: A systems approach. *Journal of Management Information Systems*, 1(2): 18–32, 1985.

[2] S Alter. A study of computer aided decision making in organizations. Massachusetts Institute of Technology, 1975.

[3] S Alter. Development patterns for decision support systems. *MIS Quarterly*, pages 33–42, September 1978.

[4] F Amador and C Romero. Redundancy in lexicographic goal programming: An empirical approach. *European Journal of Operational Research*, 41: 347–354, 1989.

[5] M Avriel. *Nonlinear programming analysis and methods.* Prentice-Hall, New Jersey, 1976.

[6] R Benayoun, J de Montgolfier, and J Tergny. Linear programming with multiple objective functions: Step method (stem). *Mathematical Programming*, 1: 366–375, 1971.

[7] J Bennett, E D Carlson, G Giddings, and P Mantey. *The design and evaluation of an interactive geo-data analysis and display system.* Amsterdam: North Holland Publishing, 1974. Information processing-74.

[8] H P Benson. Existence of efficient solutions for vector maximizationproblems. *Journal of Optimization Theory and Applications*, 26(4): 569–580, 1978.

[9] A K Bhunia and J Majumdar. Elitist genetic algorithm for assignment problem with imprecise goal. *European Journal of Operational Research*, 177: 684–692, 2007.

[10] C Botha, E Ferreira, G Geldenhuys, and H Ittman. *Selected topics in Operations Research: Quantitative Management*. UNISA, Pretoria, 1998.

[11] J T Buchanan. Multiple objective mathematical programming: A review. *New Zealand Operational Research*, 14(1): 1–27, January 1986.

[12] R Caballero, M Luque, J Molina, and F Ruiz. Mopen: A computational package for linear multi-objective and goal programming problems. *Decision Support Systems*, 41: 160–175, 2005.

[13] E D Carlson and R H Sprague. *Building effective Decision Support Systems*. N.J.: Prentice-Hall, 1982. Englewood Cliffs.

[14] A Charnes and W W Cooper. Goal programming and multi-objective optimization. *European Journal of Operational Research*, 1(1): 39–45, 1977.

[15] R Chelouah and P Siarry. A hybrid method combining continuous tabu search and nelder-mead simplex algorithms for the global optimization of multiminima functions. *European Journal of Operational Research*, 161: 636–654, 2005.

[16] I D Chivers. *Introduction to programming with Fortran: With coverage of Fortran 90, 95, 2003*. Springer, London, 2006.

[17] J F Courtney and G L Sanders. A field study of organizational factors influencing DSS success. *MIS Quarterly*, 9(1): 77–93, March 1985.

[18] P Csaki, T Rapsak, P Turchanyi, and M Vermes. R and D for group decision aid in Hungary by WINGDSS: A Microsoft Windows based Group Decision Support System. *Decision Support Systems*, 14: 205–217, 1995.

[19] N O Da Cunha and E Polak. Constrained minimization under vector-valued criteria in finite dimensional spaces. *Journal of Mathematical Analysis and Applications*, 19(1): 103–124, 1967.

[20] K Curley and L Gremillion. The role of the champion in DSS implementation. *Information and Management*, 6(4): 203–209, 1983.

[21] G B Dantzig. *A complementary algorithm for an optimal capital path with invariant proportions*. International Institute for Applied Systems Analysis, 1973.

[22] K Deb. *Multi-objective optimization using evolutionary algorithms.* John Wiley and Sons, New York, 2001.

[23] B L Dos-Santos and C W Holsapple. A framework for designing adaptive decision support system interfaces. *Decision Support Systems*, 5: 1–11, 1989.

[24] K A Dowsland. *Advanced topics in computer science series:Modern heuristic techniques for combinatorial problems.* McGraw-Hill, first edition, 1995.

[25] M Ehrgott. *Multicriteria optimization.* Springer, Auckland, second edition, 2005.

[26] G W Evans. An overview of techniques for solving multi-objective mathematical programs. *Management Science*, 30(11): 1268–1282, 1984.

[27] L R Gardiner and R E Steuer. Unified interactive multiple objective programming. *European Journal of Operational Research*, 74: 371–406, 1994.

[28] A M Geoffrion. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, 22: 619–630, 1968.

[29] M Gershon. The role of weights and scales in the application of multi-objective decision making. *European Journal of Operational Research*, 15: 244–250, 1984.

[30] L Gremillion and P Pyburn. Justifying decision support and office automation systems. *Journal of Management Information Systems*, 2(1): 5–17, 1985.

[31] Y Y Haimes, L S Lasdon, and D A Wismer. On a bicriterion formulation of the problems of intergrated system identificationand system optimization. *IEEE Transactions on Systems,Man, and Cybernetics*, 1(3): 296–297, 1971.

[32] M I Henig and Z Ritz. Multiplicative decision rules for multi-objective decision problems. *European Journal of Operational Research*, 26: 134–141, 1986.

[33] C L Hwang and A S M Masud. *Multiple objective decision making: Methods and applications: A state-of-the-art survey, Lecture notes in economics and mathematical systems*, volume 164. Springer-Verlag, Berlin, Heidelberg, 1979.

[34] N Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4(4): 373–395, 1984.

[35] G Kasper. The effect of user-developed DSS applications on forecasting decision-making performance in an experimental setting. *Journal of Management Information Systems*, 2(2): 26–39, 1985.

[36] P Keen. Decision support systems: Translating analytic techniques into useful tools. *Sloan Management Review*, 22(3): 33–44, 1980.

[37] P Keen and G Wagner. DSS: An executive mind-support system. *Datamation*, 25(11): 117–122, November 1979.

[38] P G Keen and M J Meador. Setting priorities for DSS development. *MIS Quarterly*, 8(2): 117–129, June 1984.

[39] P G W Keen and M S S Morton. *Decision support systems: An organizational perspective*. Reading Mass,Addison-Wesley, 1978.

[40] K C Kiwiel. A descent method for nonsmooth convex multiobjective minimization. *Large Scale Systems*, 8: 119–129, 1985.

[41] K C Kiwiel. Proximity control in bundle methods for methods for convex nondifferentiable minimization. *Mathematical Programming*, 46: 105–122, 1990.

[42] B Liu. *Introduction to uncertain programming*. Tsinghua University, 2006.

[43] D G Luenberger. *Introduction to linear and nonlinear programming*. Addison-Wesley Publishing Company, Menlo-Park, 1973.

[44] M K Luhandjula. Fuzzy stochastic linear programming: Survey and future research directions. *European Journal of Operational Research*, 174: 1353–1367, 2006.

[45] R Mann and H Watson. A contingency model for user involvement in DSS development. *MIS Quarterly*, 8(1): 27–38, March 1984.

[46] R Marti. *Metaheuristic procedures for training neural networks*. Springer, New York, 2006.

[47] A McCosh and M Scott-Morton. *Management decision support systems*. John Wiley and Sons, New York, 1978.

[48] K M Miettinen. *Nonlinear multi-objective optimization*. Kluwer Academic Publishers, 101 Philip Drive, Assinipi Park, Norwell, Massachusetts 02061 USA, first edition, 1999.

[49] K M Miettinen and M M Makela. Interactive bundle-based method for nondifferentiable multi-objective optimization: NIMBUS. *Optimization*, 34: 231–246, 1995.

[50] K M Miettinen and M M Makela. Optimization system www-nimbus. *Laboratory of Scientific Computing*, 9, 1998. University of Jyvaskyla, Department of Mathematics, Finland.

[51] K M Miettinen and M M Makela. Synchronous approach in interactive multi-objective optimization. *European Journal of Operational Research*, 170(3): 909–922, 2006.

[52] S Mishra. Weighting method for bi-level linear fractional programming problems. *European Journal of Operational Research*, 2006. Article in press.

[53] MRB. *CPLEX 11*, October 2007. www.gams.com/dd/docs/solvers/cplex.pdf.

[54] W Ogryczak, K Studzinski, and K Zorychta. Dinas: A computer-assisted analysis system for multi-objective transshipment problems with facility location. *Computers Operations Research*, 19(7): 637–647, 1992.

[55] Dash Optimization. *Xpress-Optimizer Reference Manual*. Dash Optimization Ltd, 15 edition, 2002.

[56] B Ramesh and K Sengupta. Multimedia in a design rationale decision support system. *Decision Support Systems*, 15: 181–196, 1995.

[57] B Render and R M Stair. *Quantitative Analysis for Management*. Prentice Hall, sixth edition, 1997.

[58] L Scharage. *Optimization modeling with LINGO*. LINDO Systems Inc, Chicago, Illinois, sixth edition, 2006.

[59] Helios Software Solutions. *Textpad*. http://wapedia.mobi/en/textpad, 1992.

[60] R Sprague. A framework for the development of decision support systems. *MIS Quarterly*, 4(4): 1–26, December 1980.

[61] R H Sprague and H J Watson. *Decision support systems: Putting theory into practice*. Prentice-Hall, New Jersey, second edition, 1989.

[62] J Sutton. Evaluation of a decision support system: A case study with the office products division of ibm. *IBM research report*, 1978.

[63] R J Thierauf. *Decision support systems for effective planning and control: A case study approach*. Prentice-Hall, Englewood Cliffs, N.J. 07632, June 1982.

[64] D Vanderpooten. *Stochastic versus fuzzy approaches to multi-objective mathematical programming under uncertainty*, chapter Multi-objective programming: Basic concepts and approaches, pages 7–22. Kluwer Academic Publishers, Netherlands, 1990.

[65] G Wagner. Optimizing decision support systems. *Datamation*, 26(5): 209–214, May 1980.

[66] W L Winston. *Operations research: Applications and algorithms*. International Thomson Publishing, California, third edition, 1994.

Appendix

```
<!DOCTYPE HTML PUBLIC "   -//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>DSS4MOPP</TITLE>
<link rel="stylesheet" type="text/css" href="style.css" />
<META NAME="Generator" CONTENT="TextPad 4.6">
<META NAME="Author" CONTENT="?">
<META NAME="Keywords"    CONTENT="?">
<META NAME="Description" CONTENT="?">
<script>


function chooseApplication(f){

//'width=somenumber,height=somenumber,left=somenumber,top=somenumber,toolbar=y
es/no,scrollbars=yes/no,location=yes/no');
 // window.alert(f.txt8.value+ ' '+f.txt9   .value );
  //condition 1
  var value1 = form1.txt7.value;
  var value2 = form1.txt8.value;
  var value3 = form1.txt9.value;
  var value4 = form1.txt10.value;
  var value5 = form1.txt11.value;

  var bool1 = form1.txt7.disabled;
  var bool2 = form1.txt8.di   sabled;
  var bool3 = form1.txt9.disabled;
  var bool4 = form1.txt10.disabled;
  var bool5 = form1.txt11.disabled;


//condition 1  --LINGO/MOPEN

if(value1 == 'no' && bool2 == true && bool3 == true &&  value4 == 'yes' &&
  bool5 == false){
  window.open( '. \\goalProgramming.html',toolbar='no','scrollbars=yes,
width=900,height=600'); //'this' ,'scrollbars=yes, width=50,height=50');
}

//condition 2

else if(value1 == 'no' && bool2 == true && value3 == 'no' && value4 == 'no' &&
value5 == 'yes'   )
  {
   window.open( '. \\Lexicographic.html',toolbar='no','scrollbars=yes,
width=900,height=600'); //'this' ,'scrollbars=yes, width=50,height=50');
```

```
//condition 3

else if(value1 == 'no' && value2== 'no' && value3 == 'no' && value4 == 'no' &&
value5 == 'no')
  {
   window.open( '. \\compromise.html',toolbar='no','scrollbars=yes,
width=900,height=600'); //'this' ,'scrollbars=yes, width=50,height=50');
   }


//condition 5

else if(value1 == 'no' && bool2 == true && value3 == 'yes' && value4 == 'no' &&
value5 == 'no' )
  {
   window.open( '. \\weighting.html',toolbar='no','scrollbars=yes, width=900,height=600');
//'this' ,'scrollbars=yes, width=50,height=   50');
    }

//condition 7

else if( value1 =='no' &&  value2 == 'yes' && bool3 == true && bool4 == true &&
bool5== true )
  {
    window.open( '. \\ReferencePoint.html',toolbar='no','scrollbars=yes,
width=900,height=600'); //'this' ,'scrollbars=yes, width=    30,height=30');
  }

//---------------------------------   NIMBUS
//condition  A

else if( value1 =='yes' &&  bool2 == true && bool3 == true && value1 == 'no' &&
value1 == 'no' )
  {

    window.open( '. \\proximal.html',toolbar='no','scrollbars=yes, width=90    0,height=600');
//'this' ,'scrollbars=yes, width=50,height=50');
  }

//condition B

else if( value1 =='yes' &&  bool2 == true && bool3 == true && value1 == 'yes' &&
value1 == 'yes' )
```

```
    {
  window.open( '. \\goalNimbus.html',toolbar='no','scrollbars=yes,
width=900,height=600'); //'this' ,'scrollbars=yes, width=50,height=50');
  }
//condition C

else if( value1 ='yes' &&  bool2 == true && bool3 == true && value1 == 'yes' &&
value1 ==  'no')
  {

    window.open( '. \\nimbus2.html',toolbar='no','scrollbars=yes, width=900,height=600');
//'this' ,'scrollbars=yes, width=50,height=50');
   }


//condition D

 else if( value1 ='yes' &&  bool2 == true && value3 == 'no' && value4 == 'no' &&
value5 == 'no')
  {

       window.open( '. \\proximal.html',toolbar='no','scrollbars=yes,
width=900,height=600'); //'this' ,'scrollbars=yes, width=50,height=50');
    }


//condition E

 else if( value1 ='yes' &&  bool2 == true && value3 == 'yes' && value4      == 'no' &&
value5 == 'no')
  {


      window.open( '. \\nimbus4.html',toolbar='no','scrollbars=yes, width=900,height=600');
//'this' ,'scrollbars=yes, width=50,height=50');
    }

//condition F
else if( value1 ='yes' &&  value2 == 'yes' && bool3 == true     && bool4 == true &&
bool5== true )
  {


      window.open( '. \\nimbus5.html',toolbar='no','scrollbars=yes, width=900,height=600');
//'this' ,'scrollbars=yes, width=50,height=50');
    }
```

```
else{

 window.open( '. \\default.html',toolbar='no','scrollbars=yes, width=900,height=600');
//'this' ,'scrollbars=yes, width=50,height=50');

  }
}



function savefile( f ) {
 f = f.elements;  //  reduce overhead

 var w = window.frames.w;
 if( !w ) {
  w = document.createElement( 'iframe' );
  w.id = 'w';
  w.style.display = 'none';
  document.body.insertBefore( w );
  w = window.frames.w;
  if( !w ) {
   w = window.open( '', '_temp', 'width=100,height=100' );
   if( !w ) {
    window.alert( 'Sorry, co  uld not create file.' ); return false;
   }
  }
 }

 var d = w.document;

 d.open('text/plain', 'replace' );
 d.charset ="utf -8";
  d.writeln( "1. At least one objective or constraint function from MOPP nonlinear?
"+f.txt7.value );
  d.writeln( "2. Decision Maker willing to work together with analyst throughout the
solution process? " +f.txt8.value );
  d.writeln( "3. Decision Maker willing to choose solution from a set of  all possible
solutions? " + f.txt9.value );
  d.writeln( "4.  Decision Maker have specific targets for each objective? " + f.txt10.value
);
 d.writeln( "5. Decision Maker regard some objectives as more important than others? " +
f.txt11.value );
```

```
 d.write(" ;");
 d.close();


 if( d.execCommand( 'SaveAs', null,'myData.txt') ){
  window.alert( ' Your data has been saved.' );
    //window.open( '. \\lingo.html', "this",toolbar='no'); //check how to make the next page
   chooseApplication(f);

 } else {
  window.alert( 'Your data was not saved.' );
 }
 w.close();
 return false;  //  don't submit the form
}



//-----------------------------------------------------------------
/* Mopen (for reference point method) and Nimbus
   If Yes for question  6: Decision Maker prefer to work together with
   analyst throughout the solution process? is answered YES then
   deactivate  questions  7  -9.
 */


/*  Comment


*/

function checkOne(){
  var value1 = form1.txt7.value;
  var value2 = form1.txt10.value;
  var value3 = form1.txt11.value;

          if(value1 == 'no' && value2 =='yes' && value3 != 'irrelevant') {
                    form1.txt8.disabled= 1;
               form1.txt9.disabled= 1;
        return true;

    }
   else{
              form1.txt8.disabled= 0;
              form1.txt9.disabled= 0;
   }
```

```
 }

//-----------------------------------------

function checkTwo(){
  var value1 = form1.txt7.value;
  var value2 = form1.txt8.value;

          if(value1 == 'no' && value2 =='yes') {

                    form1.txt9.disabled= 1;
              form1.txt10.disabled= 1;
          form1.txt11.disabled= 1;
       return true;
    }
    else{
                    form1.txt9.disabled= 0;
              form1.txt10.disabled= 0;
          form1.txt11.disabled= 0;
    }


 }

 //----------------------------------------

// condition 4 and 5

function checkThree(){
  var value1 = form1.txt7.value;
  var value2 = form1.txt9.value;
  var value3 = form1.txt10.value;
  var value4 = form1.txt11.value;

          if(value1 == 'no' && value2 !='irrelevant' && value3 == 'no'
          && value4 != 'irrelevant' && value2 != value4 ) {

       form1.txt8.disabled = 1;
                    return true;

    }

   else{
                    form1.txt8.disabled= 0;
                    }
```

```
 }
//-----------------------------------------
//condition 7.8 and 9

function checkFour(){
  var value1 = form1.txt7.value;
  var value2 = form1.txt10.value;
  var value3 = form1.txt11.value;

          if(value1 == 'yes' && (value2 =='yes' ||value3 =='no') &&     value2 !='irrelevant'
            && value3 !='irrelevant') {


                  form1.txt8.disabled= 1;
              form1.txt9.disabled= 1;

          return true;
    }
   else{

              form1.txt8.disabled= 0;
          form1.txt9.disabled= 0;


    }
 }

 //---------------------------
//condition 10 and 11    -must ask Moeti  --clashes with conditions 7,8 and 9 above

function checkFive(){
  var value1 = form1.txt7.value;
  var value2 = form1.txt9.value;
  var value3 = form1.txt10.value;
  var value4 = form1.txt11.value;

          if(value1 == 'yes' &&   value2 !='irrelevant' && value3 == 'no' && value4 == 'no')
{
       form1.txt8.disabled= 1;

       return true;

    }

   else{
```

```
                    form1.txt8.disabled= 0;

    }

 }
//----------------------------------------


function checkSix(){
  var value1 = form1.txt7.value;
  var value2 = form1.txt8.value;

  if(value1 == 'yes' && value2 =='yes') {
        form1.txt9.disabled= 1;
           form1.txt10.disabled= 1;
              form1.txt11.disabled= 1;


        return true;

    }

   else{

        form1.txt9.disabled= 0;
           form1.txt10.disabled= 0;
              form1.txt11.disabled= 0;

    }

 }

//----------------------------------------
function targetsAndObjectImportance(){
           if(checkOne()){}
           else if(checkTwo()){}
           else if(checkThree()){}
           else if(checkFive()){}
           else if(checkFour()){}
           else if(checkSix()){}
           else{
                    form1.txt7.disabled= 0;

              form1.txt8.disabled= 0;
        form1.txt 9.disabled= 0;
```

```
   form1.txt10.disabled= 0;
        form1.txt11.disabled= 0;

        }

}


//---------------------------------


</script>


</HEAD>

<BODY BGCOLOR="white" TEXT="#000000" LINK="#FF0000" VLINK="#800000"
ALINK="#FF00FF" BACKGROUND="?">

<table>
<TABLE  ALIGN="center" BORDER=0 CELLSPACING=0 CELLPADDING=0
WIDTH="100%">
<TR><IMG SRC = "gfx  \logo.gif" BORDER = 0></TR>
<TR ALIGN="MIDDLE" VALIGN="middle">
<TH><H1>DSS4MOPP</H1></TH>
</TR>

<TR><TD>


Welcome to the Decision Support System (DSS) for Multi    -objecti ve Programming
Problems (MOPP).
The aim is to help you to choose the appropriate method and to solve your MOPP.


Your comments may be sent to Moeti Ramokgadi at <a
href='mailto:0732052145@mtnloaded.co.za'>ramokmj@unisa.ac.za </a>



</td>
<tr><td>
</tr></td>

</TR>
</table>
```

```
<form  name ='form1' action="#" onsubmit="return savefile(this);">


<!-- first two questions have been removed from the list of questions. The naming of the questions
  have not been changed so the new question has name: txt3 instead of the more natural txt1 -->


<fieldset COLOR="black"><legend>Select options</legend>

<table ALIGN="center" border='1'>
<tr><td>


1. At least one objective or constraint function from MO    PP nonlinear?

</td><td><select name="txt7" title=" " onBlur='targetsAndObjectImportance()' >
      <option value="yes">Yes</option>
  <option value="no">No</option>
      <option value="irrelevant" selected="selected">Irrelevant</option>
   </select></td> </tr>
<tr><td>


2.          Is the Decision Maker willing to work together with the Analyst throughout the solution process?
</td></td><td ALIGN="center">
 <!-- div id = 'div1' style='visibility:hidden'> irrelevant </div     -->
 <select name="txt8" title=" " onBlur    ="targetsAndObjectImportance()" >
      <option value="yes">Yes</option>
      <option value="no">No</option>
      <option value="irrelevant" selected="selected">Irrelevant</option>
  </select>
</td></tr>
<tr><td>

3. Is the Decision Maker willing to have    as many options as possible to choose from ?
</td></td><td ALIGN="center">
<!--div id = 'div2' style='visibility:hidden'> irrelevant </div>     -->
  <select name="txt9" title=" " onBlur='targetsAndObjectImportance()' >
      <option value="yes">Yes</option>
      <option value="no">No</option>
```

```
      <option value="irrelevant" selected="selected">Irrelevant</option>
   </select>
</td></tr>
<tr><td>
4.        Does the Decision Maker have specific targets for each objective?
</td><td ALIGN="center">
  <select name="txt10" title=" " onBlur='targetsAndObjectImportance()' >
     <option value="yes">Yes</option>
  <option value="no">No</option>
     <option value="irrelevant" selected="selected">Irrelevant</option>
   </select>
</td></tr>


<tr><td>

5. Does the Decision Maker regard some objectives as more important than others?
</td><td ALIGN="center">
  <select name="txt11" title=" " onBlur='targetsAndObjectImportance()'>
     <option value="yes">Yes</option>
     <option value="no">No</option>
     <option value="irrelevant" selected="selected">Irrelevant</option>
  </select>
</td></tr>

<tr><td>

<!--
 10. Are you able to offer aspiration levels for each objective ? Yes/No
</td><td ALIGN="center">
  <select name="txt12" title=" ">
     <option value="yes">Yes</option>
     <option value="no" selected="selected">No</option>
  </select>
<br/></td></tr>
-->




</table>

<table ALIGN="center">

<tr>
```

```
<td ALIGN="center">
<input  type="submit" class="key" value="Continue " title=" Save      - Alt+S "
accesskey="s"/></td></tr>
</table>

</fieldset>
</form>
<p align = "center"><font size = "2">@copyright Ramokgadi 2008, all rights
reserved</font></p>

</BODY>
</HTML>
```