IIASA

International Institute for Applied Systems Analysis • A-2361 Laxenburg • Austria
Tel: +43 2236 807 • Fax: +43 2236 71313 • E-mail: info@iiasa.ac.at • Web: www.iiasa.ac.at

INTERIM REPORT IR-98-010 /March

Multiple Objective Programming Support

Pekka Korhonen (korhonen@iiasa.ac.at)

Approved by
Gordon MacDonald (macdon@iiasa.ac.at)
Director, IIASA

Contents

Abstract

This paper gives a brief introduction into multiple objective programming support. We will overview basic concepts, formulations, and principles of solving multiple objective programming problems. To solve those problems requires the intervention of a decision-maker. That’s why behavioral assumptions play an important role in multiple objective programming. Which assumptions are made affects which kind of support is given to a decision maker. We will demonstrate how a free search type approach can be used to solve multiple objective programming problems.

Keywords: Multiple Criteria Decision Making, Value Function, Practical Application

Acknowledgments

The research was supported, in part, by grants from the Foundation of the Helsinki School of Economics.

About the Authors

Pekka Korhonen is Project Leader of the Decision Analysis and Support Project at IIASA, and also Professor of Statistics at the Department of Economics and Management Science, Helsinki School of Economics and Business Administration.

More information about the author can be found at the Web site:

http://www.iiasa.ac.at/~korhonen

Multiple Objective Programming Support[1]

Pekka Korhonen

1. Introduction

Before we can consider the concept *Multiple Objective Programming Support* (MOPS), we have to first explain the concept *Multiple Criteria Decision Making* (MCDM). Even if there is a variation of different definitions, most researchers working in the field might accept the following general definition: *Multiple Criteria Decision Making* (MCDM) refers to the solving of decision and planning problems involving multiple (generally conflicting) criteria. "*Solving"* means that a decision-maker (DM) will choose one "*reasonable*" alternative from among a set of available ones. It is also meaningful to define that the choice is irrevocable. For an MCDM-problem it is typical that no unique solution for the problem exists. Therefore to find a solution for MCDM-problems requires the intervention of a decision-maker (DM). In MCDM, the word "reasonable" is replaced by the words "*efficient/nondominated*". They will be defined later on.

Actually the above definition is a strongly simplified description of the whole (multiple criteria) decision making process. In practice, MCDM-problems are not often so well-structured, that they can be considered just as a choice problem. Before a decision problem is ready to be "*solved*", the following questions require a lot of preliminary work: How to structure the problem? How to find essential criteria? How to handle uncertainty? These questions are by no means outside the interest area of MCDM-researchers. The Outranking Method by Roy [1973] and the AHP (the Analytic Hierarchy Process) developed by Saaty [1980] are examples of the MCDM-methods, in which a lot of effort is devoted to problem structuring. Both methods are well known and widely used in practice. In both methods, the presence of multiple criteria is an essential feature, but the structuring of a problem is an even more important part of the solution process.

When the term "s*upport*" is used in connection with MCDM, we may adopt a broad perspective and refer with the term to all research associated with the relationship between the problem and the decision-maker. In this paper we take a narrower perspective and focus on a very essential supporting problem in Multiple Criteria Decision Making: How to assist a DM to find the “best” solution from among a set of available "reasonable"

[1] The paper is forthcoming as an invited article in **Floudas and Pardalos** (eds.) Encyclopedia of Optimization, Kluwer.

alternatives, when the alternatives are evaluated by using several criteria? Available alternatives are assumed to be defined explicitly or implicitly by means of a mathematical model. The term *Multiple Objective Programming* is usually used to refer to this kind of model.

The following considerations are general in the sense that usually it is not necessary to specify how the alternatives are defined. It is enough to assume that they belong to set Q. However, in Figures 1 and 2 and the numerical example we consider a multiple objective linear programming model in which all constraints and objectives are defined using linear functions.

The paper consists of seven sections. In Section 2, we give a brief introduction to some foundations of multiple objective programming. How to generate potential "reasonable" solutions for a DM's evaluation is considered in Section 3, and in Section 4, we will review general principles to solve a multiple objective programming problem. In Section 5, a multiple criteria decision support system VIG is introduced, and a numerical example is solved in Section 6. Concluding remarks are given in Section 7.

2. A Multiple Objective Programming Problem

A multiple objective programming (MOP) problem in a so-called criterion space can be defined as follows:

$$\begin{aligned} &\text{“}\mathit{max}\text{”} \quad \boldsymbol{q} \\ &\mathit{s.t.} \quad\quad \boldsymbol{q} \in \mathrm{Q}, \end{aligned} \tag{2.1}$$

where set $\mathrm{Q} \subset \Re^k$ is a so-called *feasible region* in a k-dimensional criterion space $\Re^k$. The set Q is of special interest. Most considerations in multiple objective programming are made in a criterion space.

Set Q may be convex/nonconvex, bounded/unbounded, precisely known or unknown, consist of finite or infinite number of alternatives, etc. When Q consists of a finite number of elements which are explicitly known in the beginning of the solution process, we have an important class of problems which may be called e.g. *(Multiple Criteria) Evaluation Problems*. Sometimes those problems are referred to as *Discrete Multiple Criteria Problems* or *Selection Problems* (for survey, see, e.g. Olson 1996).

When the number of alternatives in Q is infinite and not countable, the alternatives are usually defined using a mathematical model formulation, and the problem is called continuous. In this case we say that the alternatives are only implicitly known. This kind of problem is referred as a *Multiple Criteria Design Problem* or a *Continuous Multiple*

Criteria Problem. In this case, the set Q is not specified directly, but by means of decision variables as usually done in single optimization problems:

$$\begin{aligned} &\text{“}max\text{”} \quad \boldsymbol{q} = \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}),...,f_k(\boldsymbol{x})) \\ &s.t. \quad \boldsymbol{x} \in X, \end{aligned} \tag{2.2}$$

where $X \subset \Re^n$ is a *feasible set* and $\boldsymbol{f}: \Re^n \rightarrow \Re^k$. The space $\Re^n$ is called a *variable space* (see Fig. 1). The functions f_i, $i = 1, 2, \ldots, k$ are *objective functions*. The feasible region Q can now be written as $Q = \{\boldsymbol{q} \mid \boldsymbol{q} = \boldsymbol{f}(\boldsymbol{x}),\ \boldsymbol{x} \in X\}$.

The MOP-problem has seldom a unique solution, i.e. an optimal solution that simultaneously maximizes all objectives. Conceptually the multiple objective mathematical programming problem may be regarded as a value (utility) function maximization program:

$$\begin{aligned} &max \quad v(\boldsymbol{q}) \\ &\text{s.t.} \quad \boldsymbol{q} \in Q, \end{aligned} \tag{2.3}$$

where v is a real-valued function, which is strictly increasing in the criterion space and defined at least in the feasible region Q. It is mapping the feasible region into a one-dimensional *value* space (see,Fig. 1). Function v specifies the DM's preference structure over the feasible region. However, the key assumption in multiple objective programming is that v is unknown. Generally, if the value function is estimated explicitly, the system is considered to be in the MAUT category (see, e.g. Keeney and Raiffa, 1976) (MAUT = Multiple Attribute Utility Theory) and can then be solved without any interaction of the DM. Typically, MAUT-problems are not even classified under the MCDM-category. If the value function is implicit (assumed to exist but is otherwise unknown) or no assumption about the value function is made, the system is usually classified under MCDM (Dyer et al., 1992) or MOP.

Solutions of the MOP-problems are all those alternatives which can be the solutions of some value function $v: Q \rightarrow \Re$. Those solutions are called *efficient* or *nondominated* depending on the space where the alternatives are considered. The term nondominated is used in the criterion space and efficient in the variable space. (Some researchers use the term efficient to refer to efficient and nondominated solutions without making any difference.) Any choice from among the set of efficient (nondominated) solutions is an acceptable and “reasonable” solution, unless we have no additional information about the DM's preference structure.

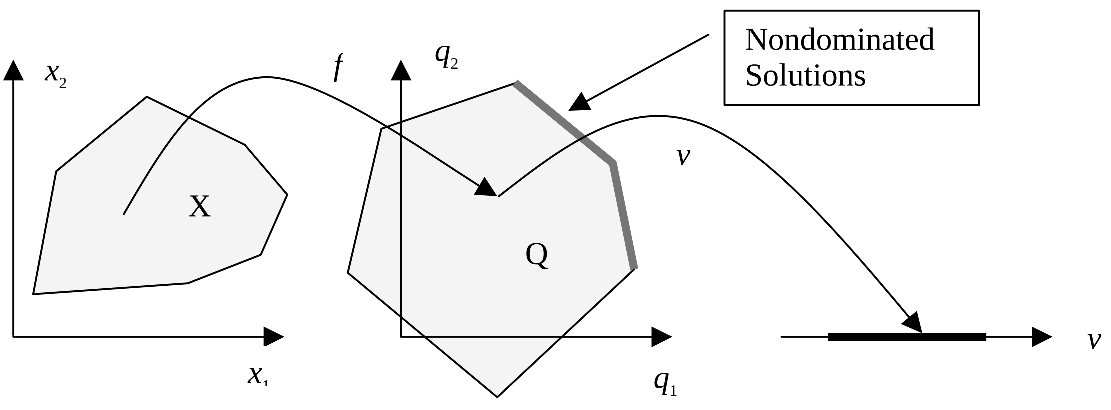


Figure 1: A Variable, Criterion and Value Space

Nondominated solutions are defined as follows:

Definition 1. In (2.1), $\boldsymbol{q}^* \in$ Q is *nondominated* iff there does not exist another $\boldsymbol{q} \in$ Q such that $\boldsymbol{q} \geq \boldsymbol{q}^*$ and $\boldsymbol{q} \neq \boldsymbol{q}^*$.

Definition 2. In (2.1), $\boldsymbol{q}^* \in$ Q is *weakly nondominated* iff there does not exist another $\boldsymbol{q} \in$ Q such that $\boldsymbol{q} > \boldsymbol{q}^*$.

Correspondingly, efficient solutions are defined as follows:

Definition 3. In (2.2), $\boldsymbol{x}^* \in$ X is *efficient* iff there does not exist another $\boldsymbol{x} \in$ X such that $\boldsymbol{f}(\boldsymbol{x}) \geq \boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{f}(\boldsymbol{x}) \neq \boldsymbol{f}(\boldsymbol{x}^*)$.

Definition 4. In (2.2), $\boldsymbol{x}^* \in$ X is *weakly efficient* iff there does not exist another $\boldsymbol{x} \in$ X such that $\boldsymbol{f}(\boldsymbol{x}) > \boldsymbol{f}(\boldsymbol{x}^*)$.

The final ("best") solution $\boldsymbol{q} \in$ Q of the problem (2.1) is called *the Most Preferred Solution*. It is a solution preferred by the DM to all other solutions. At the conceptual level, we may think it is the solution maximizing an (unknown) value function in problem (2.3). How to find it? That is the problem we now proceed to consider.

Unfortunately, the above characterization of the most preferred solution is not very operational, because no system can enable the DM to simultaneously compare the final solution to all other solutions with an aim to check if it is really the most preferred or not. It is also as difficult to maximize a function we do not know. Some properties for a good system are, for example, that it makes the DM convinced that the final solution is the most preferred one, does not require too much time from the DM to find the final solution, to give reliable enough information about alternatives, etc.. Even if it is impossible to say which system provides the best support for a DM for his multiple criteria problem, all proper systems have to be able to recognize, generate and operate with nondominated solutions. To generate nondominated solutions for the DM’s evaluation is thus one key issue in multiple objective programming. In the next section, we will consider some principles.

3. Generating Nondominated Solutions

Despite many variations among different methods of generating nondominated solutions, the ultimate principle is the same in all methods: a single objective optimization problem is solved to generate a new solution or solutions. The objective function of this single objective problem may be called a *scalarizing function* according to Wierzbicki [1980]. It typically has the original objectives and a set of parameters as its arguments. The form of the scalarizing function as well as what parameters are used depends on the assumptions made concerning the DM's preference structure and behavior.

Two classes of parameters are widely used in multiple objective optimization: 1) *weighting coefficients for objective functions* and 2) *reference/ aspiration/ reservation levels for objective function values*. Based on those parameters, there exist several ways to specify a scalarizing function. An important requirement is that this function completely characterizes the set of nondominated solutions: "*for each parameter value, all solution vectors are nondominated, and for each nondominated criterion vector, there is at least one parameter value, which produces that specific criterion vector as a solution*" (see, for theoretical considerations, e.g. Wierzbicki [1986]).

3.1. A Linear Scalarizing Function

A classic method to generate nondominated solutions is to use the weighted-sums of objective functions, i.e. to use the following linear scalarizing function:

$$\max \{\lambda' \boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathrm{X}\}. \tag{3.1}$$

If $\lambda > \boldsymbol{0}$, then the solution vector $\boldsymbol{x}$ of (3.1) is efficient, but if we allow that $\lambda \geq \boldsymbol{0}$, then the solution vector is weakly-efficient. (see, e.g. Steuer [1986, p. 215 and 221]). Using the parameter set $\Lambda = \{\lambda \mid \lambda > \boldsymbol{0}\}$ in the weighted-sums linear program we can completely characterize the efficient set provided the constraint set is convex. However, Λ is an open set, which causes difficulties in a mathematical optimization problem. If we use $\mathrm{cl}(\Lambda) = \{\lambda \mid \lambda \geq \boldsymbol{0}\}$ instead, the efficiency of $\boldsymbol{x}$ cannot be guaranteed anymore. It is surely weakly-efficient, and not necessarily efficient. When the weighted-sums are used to specify a scalarizing function in multiple objective linear program (MOLP) problems, the optimal solution corresponding to nonextreme points of X is never unique. The set of optimal solutions always consists of at least one extreme point, or the solution is unbounded. In early methods, a common feature was to operate with weight vectors $\lambda \in \Re^k$, limiting considerations to efficient extreme points (see, e.g., Zionts and Wallenius [1976]).

3.2. A Chebyshev-type Scalarizing Function

Currently, the most solution methods are based on the use of a so-called Chebyshev-type scalarizing function first proposed by Wierzbicki [1980]. We will refer to this function by the term *achievement (scalarizing) function.* The achievement (scalarizing) function projects any given (feasible or infeasible) point $\boldsymbol{g} \in \Re^k$ onto the set of nondominated solutions. Point $\boldsymbol{g}$ is called a *reference point*, and its components represent the desired values of the objective functions. These values are called *aspiration levels*.

The simplest form of achievement function is:

$$s(\boldsymbol{g}, \boldsymbol{q}, \boldsymbol{w}) = \max \left[\frac{g_k - q_k}{w_k}, k \in K \right] \tag{3.2}$$

where $\boldsymbol{w} > \boldsymbol{0} \in \Re^k$ is a (given) vector of weights, $\boldsymbol{g} \in \Re^k$, and $\mathbf{q} \in Q = \{\boldsymbol{f}(\boldsymbol{x}) \mid \boldsymbol{x} \in X\}$. By minimizing $s(\boldsymbol{g}, \boldsymbol{q}, \boldsymbol{w})$ subject to $\boldsymbol{q} \in Q$, we find a weakly nondominated solution vector $\boldsymbol{q}^*$ (see, e.g. Wierzbicki [1980], [1986]). However, if the solution is unique for the problem, then $\boldsymbol{q}^*$ is nondominated. If $\boldsymbol{g} \in \Re^k$ is feasible, then $\boldsymbol{q}^* \in Q$, $\boldsymbol{q}^* \geq \boldsymbol{g}$. To guarantee that only nondominated (instead of weakly nondominated) solutions will be generated, more complicated forms for the achievement function have to be used, for example:

$$s(\boldsymbol{g}, \boldsymbol{q}, \boldsymbol{w}, \rho) = \max_{k \in K} \left[\frac{g_k - q_k}{w_k} \right] + \rho \sum_{i=1}^{k} (g_i - q_i)\}, \tag{3.3}$$

where $\rho > 0$. In practice, we cannot operate with a definition "any positive value". We have to use a pre-specified value for ρ .Another way is to use a lexicographic formulation Korhonen and Halme [1996].

The applying of the scalarizing function (3.3) is easy, because given $\boldsymbol{g} \in \Re^k$, the minimum of $s(\boldsymbol{g}, \boldsymbol{v}, \boldsymbol{w}, \rho)$ is found by solving the following LP-problem:

$$\min \quad \varepsilon + \rho \sum_{i=1}^{k} (g_i - q_i)$$

s.t. (3.4)

$$\boldsymbol{x} \in \mathbf{X}$$

$$\varepsilon \geq (g_i - q_i) / w_i \;, i = 1, 2, \; \ldots \; , \; k,$$

The problem (3.4) can be further written as:

$$min \quad \varepsilon + \rho \sum_{i=1}^{k} (g_i - q_i)$$

s.t. (3.5)

$$\boldsymbol{x} \in \mathbf{X}$$

$$\boldsymbol{q} + \varepsilon \boldsymbol{w} - \boldsymbol{z} = \boldsymbol{g}$$

$$\boldsymbol{z} \geq \boldsymbol{0}.$$

To illustrate the use of the achievement scalarizing function, consider a two-criteria problem with a feasible region having four extreme points {(0,0), (0,3), (2,3), (8,0)}, as shown in Figure 2.

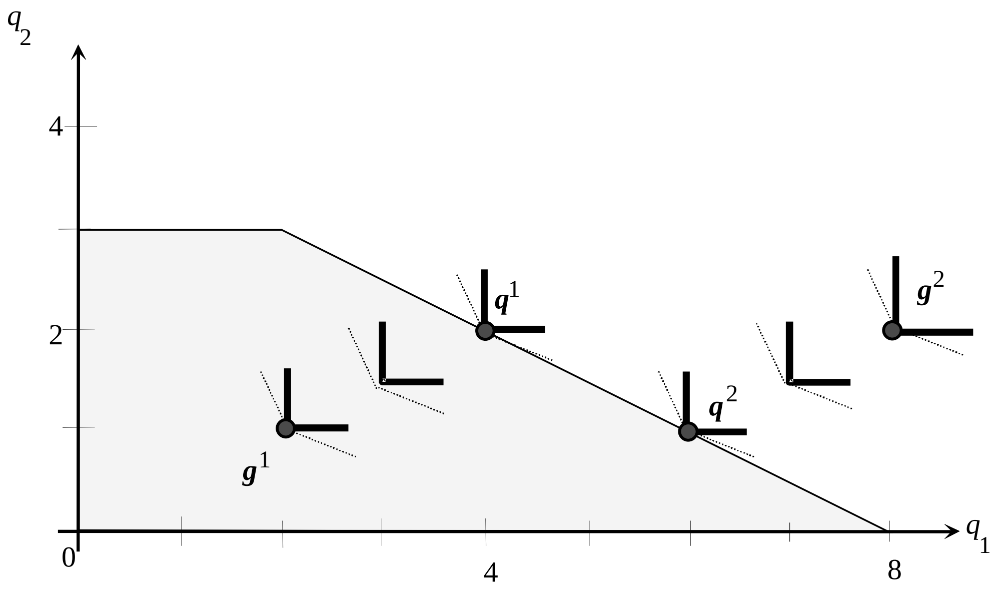


Figure 2. Illustrating the projection of a feasible and an infeasible aspiration level point onto the nondominated surface.

In Fig.2, the thick solid lines describe the indifference curves when ρ $=0$ in the achievement scalarizing function. The thin dotted lines stand for the case ρ > 0. Note that the line from (2,3) to (8,0) is nondominated and the line from (0,3) to (2,3) is weakly-nondominated, but dominated. Let us assume that the DM first specifies a feasible aspiration level point $\boldsymbol{g}^1$ = (2,1). Using a weight vector $\boldsymbol{w} = [2,1]^T$, the minimum value of the achievement scalarizing function (-1) is reached at a point $\boldsymbol{v}^1$ = (4,2) (cf. Figure 2). Correspondingly, if an aspiration level point is infeasible, say $\boldsymbol{g}^2$ = (8,2), then the minimum of the achievement scalarizing function (+1) is reached at point $\boldsymbol{v}^2$ = (6,1). When a feasible point dominates an aspiration level point, then the value of the achievement scalarizing function is always negative; otherwise it is nonnegative. It is zero, if an aspiration level point is weakly-nondominated.

4. Solving Multiple Objective Problems

Several dozen procedures and computer implementations have been developed during the last 25 years to address both *Multiple Criteria Evaluation* and *Design Problems*. The multiple objective decision procedures always requires the intervention of a DM at some stage in the solution process. A popular way to involve the DM in the solution process is to use an interactive approach.

The specifics of these procedures vary, but they have several common characteristics. For example, at each iteration, a solution, or a set of solutions, is generated for a DM's examination. As a result of the examination, the DM inputs information in the form of tradeoffs, pairwise comparisons, aspiration levels, etc. (see, for a more detailed discussion, Shin and Ravindran [1991]). The responses are used to generate a presumably, improved solution. The ultimate goal is to find the most preferred solution of the DM. Which search technique and termination rule is used is heavily dependent on the underlying assumptions postulated about the behavior of the DM and the way in which these assumptions are implemented. In MCDM-research there is a growing interest in the behavioral realism of such assumptions.

Based on the role that the value function (2.3) is supposed to play in the analysis, we can classify the assumptions into three categories:

1. Assume the existence of a value function v, and assess it explicitly.

2. Assume the existence of a stable value function v, but do not attempt to assess it explicitly. Make assumptions of the general functional form of the value function.

3. Do not assume the existence of a stable value function v, either explicit, or implicit.

The first assumption is adopted in multi-attribute utility or decision analysis (see, e.g. Keeney and Raiffa, 1976). Interactive software implementing such approaches on personal computers exists.

The second assumption was a basic paradigm used in interactive multiple criteria approaches in the 1970's. A classical example is the GDF-method by Geoffrion, Dyer, and Feinberg [1972]. DM's responses to specific questions were used to guide the solution process towards an "optimal" or "most preferred" solution (in theory), assuming that the DM behaves according to some specific (but unknown) underlying value function (see for surveys, e.g. Hwang and Masud [1979], Steuer [1986], Shin and Ravindran [1991], and White [1990]). Interactive software that implement such systems for a computer have often been developed by the authors of the above procedures for experimental purposes.

The approaches based on the assumption on "no stable value/utility function" typically operate with a DM's aspiration levels regarding the objectives on the feasible region. The aspiration levels are projected via minimizing so called achievement scalarizing functions (3.3) (Wierzbicki, 1980; Steuer and Choo, 1983). No specific behavioral assumptions e.g. transitivity are necessary.

In essence, this approach seeks to help the DM more or less freely to search the set of efficient solutions. Interactive software that implement such systems for a computer have been developed like ADBASE by Steuer [1986] and [1992], DIDAS by Lewandowski et al. [1989], and VIG and VIMDA by Korhonen [1987] and [1988].

For an excellent review of several interactive multiple criteria procedures, see Steuer [1986]. Other well-known books that provides a deeper background and additional references especially in the field of Multiple Objective Optimization include Cohon [1978], Haimes [1990], Hwang and Masud [1979], Ignizio [1976], Sawaragi, Nakayama, and Tanino [1985], Yu [1985], and Zeleny [1982].

Multiple Objective Linear Programming (MOLP) is the most commonly studied problem in *Multiple Criteria Decision Making* (MCDM). Most solution methods are developed for this problem.

5. An Example of a Decision Support System: VIG

Today, many systems use aspiration level projections, where the projection is performed using Chebyshev-type achievement scalarizing functions as explained above. These functions can be controlled either by varying weights (keeping aspiration levels fixed) or by varying the aspiration levels (keeping weights fixed). Instead of aspiration levels, some algorithms asks the DM to specify the reservation levels for the criteria (see, e.g. Michalowski and Szapiro [1992]).

An achievement scalarizing function projects one aspiration (reservation) level point at a time onto the nondominated frontier. By parametrizing the function, it is possible to project the whole vector onto the nondominated frontier as originally proposed by Korhonen and Laakso [1986]. The vector to be projected is called a *Reference Direction Vector* and the method *Reference Direction Method*, correspondingly. When a direction is projected onto the nondominated frontier, a curve traversing across the nondominated frontier is obtained. Then an interactive line search is performed along this curve. The idea enables the DM to make a continuous search on the nondominated frontier. The corresponding mathematical model is a simple modification from the original model (3.5) developed for projecting one point:

$$
\begin{aligned}
&\mathit{min} \quad \varepsilon + \rho \sum_{i=1}^{k} (g_i - q_i) \\
&\text{s.t.} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (5.1) \\
&\qquad \boldsymbol{x} \in \mathbf{X} \\
&\qquad \boldsymbol{q} + \varepsilon \boldsymbol{w} - \boldsymbol{z} = \boldsymbol{g} + t\boldsymbol{r} \\
&\qquad \boldsymbol{z} \geq \boldsymbol{0},
\end{aligned}
$$

where $t : 0 \rightarrow \infty$ and $\boldsymbol{r} \in \Re^k$ is a reference direction. In the original approach, a reference direction was specified as a vector starting from the current solution and passing through the aspiration levels. The DM was asked to give aspiration levels for criteria.

The original reference direction approach has been further developed into many directions. First, Korhonen and Wallenius [1988] improved upon the original procedure by making the specification of a reference direction dynamic. The dynamic version was called *Pareto Race*. In Pareto Race, the DM can freely move in any direction on the nondominated frontier he/she likes, and no restrictive assumptions concerning the DM's behavior are made. Furthermore, the objectives and constraints are presented in a uniform manner. Thus, their role can also be changed during the search process. The method and its implementation is called Pareto Race. The whole software package consisting of Pareto Race is called VIG.

In Pareto Race, a reference direction $\boldsymbol{r}$ is determined by the system on the basis of preference information received from the DM. By pressing number keys corresponding to the ordinal numbers of the objectives, the DM expresses which objectives he/she would like to improve and how strongly. In this way he/she implicitly specifies a reference direction. Figure 3 shows the Pareto Race interface for the search, embedded in the VIG software (Korhonen [1987]).

Thus Pareto Race is a visual, dynamic, search procedure for exploring the nondominated frontier of a multiple objective linear programming problem. The user sees the objective function values on a display in numeric form and as bar graphs, as he/she travels along the nondominated frontier. The keyboard controls include an accelerator, gears, brakes, and a steering mechanism. The search on the nondominated frontier is like driving a car. The DM can, e.g., increase/decrease the speed, make a turn and brake at any moment he/she likes.

To implement those features, Pareto Race uses certain control mechanisms, which are controlled by the following keys:

(SPACE) BAR: An “Accelerator”

Proceed in the current direction at constant speed.

F1: ”Gears (Backward)”

Increase speed in the backward direction.

F2: ”Gears (Forward)”

Increase speed in the forward direction.

F3: “Fix”

Use the current value of objective *i* as the worst acceptable value.

F4: “Relax”

Relax the “bound” determined with key F3.

F5: ”Brakes”

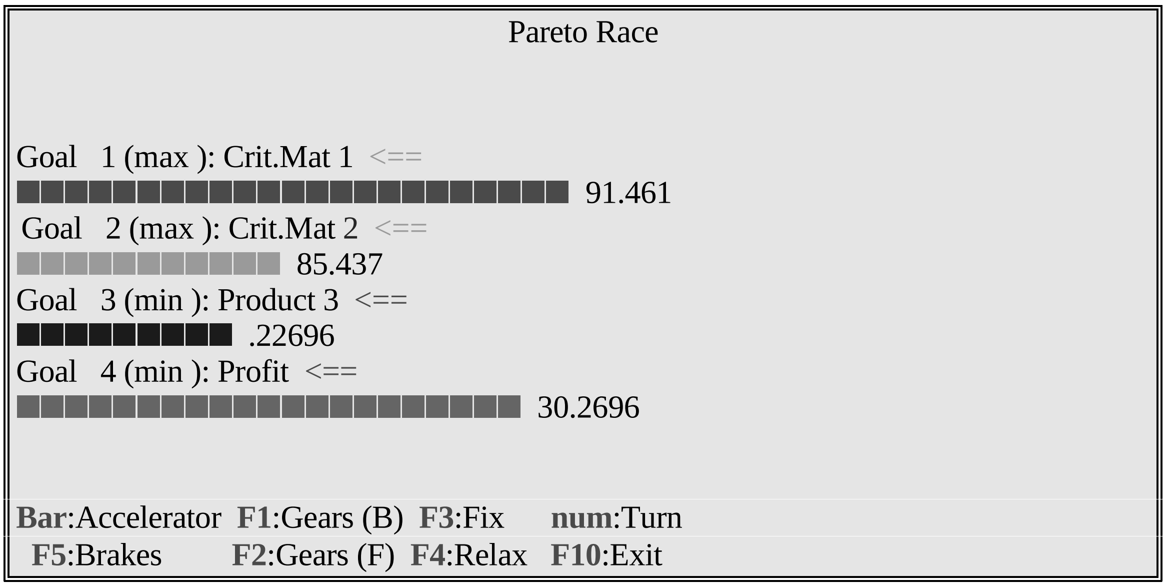
Reduce speed.

F10: “Exit”

num: ”Turn”

Change the direction of motion by increasing the component of the reference direction corresponding to the goal's ordinal number $i \in [1, k]$ pressed by DM.

An example of the Pareto Race screen is given in Fig. 3 below. The screen is associated with the numerical example described in the next section.



```
Pareto Race

Goal   1 (max ): Crit.Mat 1  <==
                                         91.461
Goal   2 (max ): Crit.Mat 2  <==
                    85.437
Goal   3 (min ): Product 3  <==
                 .22696
Goal   4 (min ): Profit  <==
                                      30.2696

Bar:Accelerator  F1:Gears (B)  F3:Fix      num:Turn
  F5:Brakes        F2:Gears (F)  F4:Relax   F10:Exit
```

Figure 3: An Example of the Pareto Race Screen

Pareto Race does not specify restrictive behavioral assumptions for a DM. He/she is free to make a search on the nondominated surface, until he/she believes that the solution found is his/her most preferred one.

Pareto Race is only suitable for solving moderate size problems. When the size of the problem becomes large, computing time makes the interactive mode inconvenient. To solve large-scale problems Korhonen, Wallenius and Zionts [1992] proposed a method based on Pareto Race. An (interactive) free search is performed to find the most preferred direction. Based on the direction, an nondominated curve can be generated in a batch mode if desired.

6. Numerical Illustrations

For illustrative purposes, we will consider the following production planning problem, where a decision maker (DM) tries to find the “best” product-mix for three products: Product 1, Product 2, and Product 3. The production of these products requires the use of one machine (Mach. Hours), man-power (Man Hours), and two critical materials (Crit. Mat 1 and Crit. Mat 2). Selling the products results in profit (Profit). Assume that the DM describes his/her decision problem as follows:

"*Of course, I would like to make as much profit as possible. Because it is difficult and quite expensive to obtain critical materials, I would like to use them as little as possible, but never more than I have presently in storage (96 units of each). Only one machine is used to produce the products. It operates without any problems for at least 9 hours. The length of the regular working day is 10 hours. People are willing to work overtime which is costly and they are tired the next day. Therefore, if possible, I would like to avoid it. Finally, product 3 is very important to a major customer, and I cannot totally exclude it from the production plan.*"

The traditional single objective programming considers the problem as a profit maximization problem. The other "requirements" are taken as constraints. The multiple objective programming takes a "softer" perspective. We may, for instance, consider the problem as a four objective problem. The DM would like to make as much profit as possible, but simultaneously, he/she would like to use those two critical materials as little as possible, and in addition to maximize the use of product 3. Machine hours and man hours are considered as constraints, but during the search process the role of constraints and objectives may also be changed, if necessary.

We assume that the problem can be modeled as an MOLP-model. The coefficient matrix of the problem is given in Table 1.

Table 1: The coefficient matrix of the production planning problem

	Product 1	**Product 2**	**Product 3**
Mach.Hours	1.5	1	1.6
Man Hours	1	2	1
Crit.Mat 1	9	19.5	7.5
Crit.Mat 2	7	20	9
Profit	4	5	3

Thus, we have the following multiple objective linear programming model:

$$\begin{array}{lcrcrcrcl}
\textbf{Crit.Mat 1} & : & 9P_1 & + & 19.5\,P_2 & + & 7.5\,P_3 & \rightarrow & \text{min} \\
\textbf{Crit.Mat 2} & : & 7P_1 & + & 20\,P_2 & + & 9\,P_3 & \rightarrow & \text{min} \\
\textbf{Profit} & : & 4P_1 & + & 5\,P_2 & + & 3\,P_3 & \rightarrow & \text{max} \\
\textbf{Product 3} & : & & & & & P_3 & \rightarrow & \text{max}
\end{array}$$

Subject to:

$$\begin{array}{lcrcrcrcl}
\textbf{Mach.Hours} & : & 1.5P_1 & + & P_2 & + & 1.6\,P_3 & \leq & 9 \\
\textbf{Man Hours} & : & P_1 & + & 2\,P_2 & + & P_3 & \leq & 10
\end{array}$$

The problem has no unique solution. Using the Pareto Race (see, Fig. 3.) or any other software developed for multiple objective programming enables a DM to search nondominated solutions. Which solution he/she will choose as a final one depends entirely on his/her own preferences. Actually, all sample solutions except solution II are somehow consistent with his/her statement above. In solution II, product 3 is excluded from the production plan.

Table 2: A Sample of Solutions for the Multiple Criteria Problem

	I	II	III	IV
Objectives:				
Crit.Mat1	91.46	94.50	93.79	90.00
Crit.Mat2	85.44	88.00	89.15	84.62
Profit	30.27	31.00	30.42	29.82
Product3	0.23	0.00	0.50	0.44
Constraints:				
Mach.Hours	9.00	9.00	9.00	9.00
Man.Hours	9.73	10.00	10.00	9.62
Decision Variables:				
Product1	3.88	4.00	3.45	3.71
Product2	2.81	3.00	3.03	2.74
Product3	0.23	0.00	0.50	0.44

7. Conclusion

In this article, we have provided an overview on Multiple Objective Programming Support. The emphasis was how to find the most preferred alternative from among a set of reasonable (nondominated) alternatives. This kind of the approach is unique for the Multiple Criteria Decision Making. We have left other features like structuring the problem, finding relevant criteria etc. beyond this presentation. They are important, but also relevant in the considerations of any decision support system.

References

Cohon, J. (1978), Multiobjective Programming and Planning, Academic Press, New York.

Dyer, J., Fishburn, P., Steuer, R., Wallenius, J., and Zionts, S. (1992), "Multiple Criteria Decision Making, Multiattribute Utility Theory - The Next Ten Years", Management Science 38, 645-654.

Geoffrion, A., Dyer, J., and Feinberg, A. (1972). "An Interactive Approach for Multi-Criterion Optimization, with an Application to the Operation of an Academic Department", Management Science 19, pp. 357-368.

Haimes, Y., Tarvainen, K., Shima, T., and Thadathil, J. (1990), Hierarchical Multiobjective Analysis of Large-Scale Systems, New York, Hemisphere Publishing Company.

Hwang, C. and Masud, **A.** (1979), Multiple Objective Decision Making - Methods and Applications: A State-of-the-Art Survey, Springer, Berlin.

Ignizio, J. (1976), Goal Programming and Extensions, D. C. Heath, Lexington, Mass.

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

Korhonen, P. (1987), "VIG - A Visual Interactive Support System for Multiple Criteria Decision Making", Belgian J. of OR, Statistics and Computer Science 27, 3-15.

Korhonen, P. (1988), "A Visual Reference Direction Approach to Solving Discrete Multiple Criteria Problems", European Journal of Operational Research 34, 152-159.

Korhonen, P. and Halme, M. (1996): "Using Lexicographic Parametric Programming for Searching a Nondominated Set in Multiple Objective Linear Programming", Journal of Multi-Criteria Decision Analysis, Vol. 5, N:o 4, pp. 291-300.

Korhonen, P., and Laakso, J. (1986). "A Visual Interactive Method for Solving the Multiple Criteria Problem", European Journal of Operational Research 24, pp. 277-287.

Korhonen, P. and Wallenius, J. (1988), "A Pareto Race", Naval Research Logistics 35, 615-623.

Korhonen, P., Wallenius, J. and Zionts, S. (1992): "A Computer Graphics-Based Decision Support System for Multiple Objective Linear Programming", European Journal of Operational Research, Vol. 60, N:o 3, pp. 280-286.

Lewandowski, A., Kreglewski, T., Rogowski, T., and Wierzbicki, A. (1989), "Decision Support Systems of DIDAS Family (Dynamic Interactive Decision Analysis & Support)", In A. Lewandowski and A. Wierzbicki (Eds.), Aspiration Based Decision Support Systems, Springer, Berlin, 21-47.

Michalowski, W. and Szapiro, T. (1992), "A Bi-Reference Procedure for Interactive Multiple Criteria Programming", Operations Research 40, pp. 247-258.

Olson, D. (1996), Decision Aids for Selection Problems, Springer Series in Operations Research, New York.

Roy, B. (1973), "How outranking relation helps multiple criteria decision making", in: J. Cochrane and M. Zeleny (eds.), Multiple Criteria Decision Making, University of South Carolina Press, Columbia, SC, 179-201.

Saaty, T. (1980), The Analytic Hierarchy Process, McGraw-Hill, New York.

Sawaragi, Y., Nakayama, H. and Tanino, T. (1985), Theory of Multiobjective Optimization, Academic Press, New York.

Shin, W. and Ravindran, A. (1991), "Interactive Multiple Objective Optimization: Survey I - Continuous Case", Computers Ops Res. 18, 97-114.

Steuer, R.E. (1986), Multiple Criteria Optimization: Theory, Computation, and Application, Wiley, New York.

Steuer, R. (1992), "Manual for the ADBASE Multiple Objective Linear Programming Package", Department of Management Science, University of Georgia, Athens, Georgia, USA.

Steuer, R. and Choo, E.-U. (1983), "An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming", Mathematical Programming 26, 326-344.

White, D. (1990), "A Bibliography on the Applications of Mathematical Programming Multiple-Objective Methods", J. Opl Res. Soc. 41, 669-691.

Wierzbicki, A. (1980), “The Use of Reference Objectives in Multiobjective Optimization”, in G. Fandel and T. Gal (Eds.), Multiple Objective Decision Making, Theory and Application, Springer-Verlag, New York.

Wierzbicki, A. (1986), “On the Completeness and Constructiveness of Parametric Characterizations to Vector Optimization Problems”, OR Spektrum 8, 73-87.

Yu, P.-L. (1985), Multiple Criteria Decision Making: Concepts, Techniques, and Extensions, Plenum, New York.

Zeleny, M. (1982), Multiple Criteria Decision Making, McGraw-Hill, New York.

Zionts, S. and Wallenius, J. (1976). ”An Interactive Programming Method for Solving the Multiple Criteria Problem”, Management Science 22, 652-663.