Vetschera, Rudolf

Working Paper

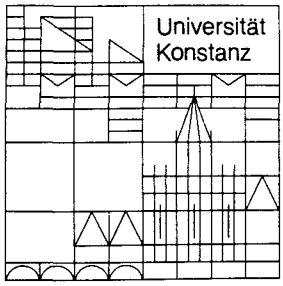
Estimating preference cones from discrete choices: Computational techniques and experiences

Diskussionsbeiträge - Serie I, No. 259

Provided in Cooperation with:
Department of Economics, University of Konstanz

Suggested Citation: Vetschera, Rudolf (1992) : Estimating preference cones from discrete choices: Computational techniques and experiences, Diskussionsbeiträge - Serie I, No. 259, Universität Konstanz, Fakultät für Wirtschaftswissenschaften und Statistik, Konstanz

This Version is available at:
https://hdl.handle.net/10419/68911



Fakultät für Wirtschaftswissenschaften und Statistik

Rudolf Vetschera

Estimating Preference Cones from Discrete Choices— Computational Techniques and Experiences

Diskussionsbeiträge

Postfach 5560
D-7750 Konstanz

Serie I — Nr. 259
März 1992

Estimating Preference Cones from Discrete Choices - Computational Techniques and Experiences

Rudolf Vetschera

Serie I - Nr. 259
März 1992

Estimating Preference Cones from Discrete Choices - Computational Techniques and Experiences *

Rudolf Vetschera

March 4, 1992

Abstract

This paper discusses the problem of estimating aspiration or reference levels in criteria space from choices between alternatives characterized by multiple criteria. Several methods for such estimations are developed and compared, taking into account both solution quality and computational efficiency. Methods based on mixed integer linear programming, which provide optimal solutions, require inacceptable computing times for application in interactive systems. An alternative method, based on a direct search algorithm, is shown to be an effective way of generating high quality estimates with small computational effort.

*Paper to be presented at the EURO/TIMS conference, Helsinki 1992

1 Introduction

Over the last 10 to 15 years, multi-criteria decision methods based on aspiration or reference levels have been established as a viable alternative to approaches based on weights and additive utility functions (c.f. Roy/Vincke, 1981; Evans, 1984; Gershon, 1984; Zanakis/Gupta, 1985; Vincke, 1986; Wierzbicki, 1986; Vanderpooten, 1989; Weidner, 1989; Habenicht, 1990; Shin/Ravindran, 1991). The importance of such approaches for multi-criteria decision making can be attributed to three factors:

- It is possible to reproduce most basic results of multi-criteria decision theory, which can be formulated using additive utility functions, via formulations based on reference outcomes (Wierzbicki, 1980).
- Reference criteria levels are formulated in decision space, while weights are dual to decision space and thus more difficult to specify (Wierzbicki, 1986).
- In combination with Tchebycheff or similar norms, approaches based on reference levels allow easy characterization of any arbitrary efficient point as an optimal solution without resorting to transformations of criteria. This is not the case for additive functions (c.f. Belkeziz/Pirlot, 1991).

If one regards aspiration or reference levels as a natural representation of preferences in multi-criteria decision problems, a logical next step is the formulation of estimation models, which determine aspiration levels consistent with observed choice behavior. However, up to now, this problem has rarely been addressed in the literature.

This lack of research is even more astonishing given that estimation procedures are an important and integral part of the literature on utility based approaches to multi-criteria decision making (e.g. Srinivasan/Shocker, 1973; Huber, 1974; Fischer, 1979; Jacquet-Lagreze/Siskos, 1982; Horsky/Rao, 1984; Klein et al., 1985; Weber, 1985; a comparison of methods is provided by Schoemaker/Waid, 1982). This contrast can be attributed to the different roles assigned to preference representations by the two schools. In utility

based approaches, it is implicitly assumed that the decision maker already has a consistent system of preferences, which only needs to be estimated to be applied to any problem at hand. Approaches based on reference levels, on the other hand, are usually based on the assumption that learning about the problem and forming preferences are the central issues in solving a decision problem. If there is no a priori fixed system of preferences, it does not make sense to try to estimate one.

While this aspect of learning and experimentation is certainly important, there is still a role for estimation techniques in reference-based decision methods. It might be easier for a decision maker, who is not yet familiar with the problem at hand, to make comparisons and indicate preferences between two alternatives, than to formulate aspiration levels for the criteria. Hence, the estimation of a preference system based on reference levels does not serve the purpose of identifying the decision maker's "true" preferences, but is an aid in formulating initial parameters for further interaction.

In this paper, we will study the problem of estimating reference levels in criteria space given choice statements between discrete alternatives. The remainder of the paper is structured as follows: Section two gives a brief review of preference representations based on preference cones, of estimation problems arising in this context and of previous work in this direction. In section three, several models for estimating reference levels from discrete choices are formulated. Section four presents results of computational experiments obtained with those models and section five summarizes the results and provides an outlook for future research.

2 Estimation Problems in Aspiration Based Decision Methods

Attempts to represent preferences in multi-criteria decision problems by means of objective levels rather than weights, date back to the concepts of goal programming, which were developed in the fifties and sixties mainly by Charnes and Cooper (Charnes et al., 1955, Charnes/Cooper, 1961). Important contributions to the theoretical foundations for this methodology were made by

Wierzbicki (1980; 1986), who proposed a representation of preferences based on cones in criteria space. This approach allows for a convenient representation of any efficient (Pareto optimal) point, even if the feasible set of the decision problem is not convex.

A simple example of this approach concerns evaluating alternatives by their Tchebycheff distance to a reference level. We consider a decision problem with K objectives $k = 1, \ldots, K$. Using the Tchebycheff norm, the evaluation of an alternative represented by a criteria vector $X_n = (x_{n1}, \ldots, x_{nK})$ relative to an aspiration vector $\overline{X} = (\overline{x_1}, \ldots, \overline{x_K})$ is given by the scalarizing function (Wierzbicki, 1986):

$$s(X_n) = \min_k (x_{nk} - \overline{x_k}) \tag{1}$$

Function (1) induces a set of rectangular indifference curves in criteria space. Maximization of (1) will always result in the selection of a weakly efficient alternative and any (weakly) efficient alternative can be chosen by shifting the reference level $\overline{X}$. If the preference cone is slightly extended as in

$$s(X_n) = \min_k (x_{n,k} - \overline{x_k}) + \epsilon \sum_k (x_{n,k} - \overline{x_k}) \tag{2}$$

only strongly efficient solutions will be chosen, but not all of them can be represented.

While, as already stated above, estimation of such preference systems is not a topic widely discussed in the literature, some similar research has been conducted:

Mustafi/Xavier (1985) extended the estimation procedure for utility weights developed by Srinivasan/Shocker (1973) to estimate "threshold values". In contrast to the formulation discussed above, their "threshold values" were properties of alternatives, which were a priori classified into "acceptable" and "inacceptable" alternatives without referring to criteria values. Jaskiewicz/Slowinski (1991) used ordinal regression techniques in order to fit scalarizing functions to preference rankings derived from outranking meth-

ods. In their approach, the scalarizing function is not fitted by changing the reference point, but by rescaling the criteria by means of additional parameters. Majchrzak (1991), in an approach still under development, proposed to fit scalarizing functions by modifying the width of the preference cone, i.e. by changing parameter ϵ in (2). Similarly, Korhonen et al. (1984) used linear combinations of objectives to construct cones of different widths for eliminating alternatives in discrete MCDM problems. Since all these approaches do not consider explicitly the estimation of reference levels, we will not discuss these techniques in more detail here.

3 Approximation Models

In the most general form, the problem of estimating parameters for any kind of preference representation can be stated as: "Minimize the difference between the ranking implied by the estimated preference representation and the preference statements made by the user". For actual application, the concept of "difference", which is left open to interpretation in the above definition, must be specified.

For decision methods which provide a cardinal evaluation of alternatives, this "difference" (and the preference statements themselves) can be measured on a cardinal or ordinal scale. For estimating utility functions, it has been argued that preference statements should not only be made in the form of ordinal choices between alternatives, but also in cardinal form. Such statements can be made as holistic cardinal evaluations of alternatives, possibly as ranges to reduce the cognitive strain on the decision maker (e.g. Weber, 1985). It is also possible to obtain cardinal information via statements on the strength of preference between alternatives (e.g. Horsky/Rao, 1984; Klein et al., 1985).

Here, we want to provide a method which allows for easy specification of initial reference levels, which will later be changed during an interactive decision process. In this context, difficult questions involving significant cognitive strain on the decision maker will be counterproductive. We therefore limit the input from the decision maker to simple statements of preference between two alternatives.

Using again the sum of deviations as an objective function, the following mixed integer programming model can be used for reference point estimation problems:

$$
\begin{array}{lcll}
\text{minimize} & \sum_{i,j:iRj} d_{ij} & & \\
z_i + d_{ij} & \geq & z_j & \forall iRj \\
z_n & \leq & x_{n,1} - \overline{x_1} & \forall n \\
 & \vdots & & \\
z_n & \leq & x_{n,K} - \overline{x_K} & \forall n \\
z_n & \geq & x_{n,1} - \overline{x_1} - \lambda_{n,1} M & \forall n \\
 & \vdots & & \\
z_n & \geq & x_{n,K} - \overline{x_K} - \lambda_{n,K} M & \forall n \\
\sum_k \lambda_{nk} & \leq & K - 1 & \forall n \\
\sum_k \overline{x_k} & = & K & \\
d_{ij} & \geq & 0 & \forall i,j \\
\overline{x_k} & \geq & 0 & \forall k
\end{array}
\tag{8}
$$

This model can be solved using any standard mixed-integer or binary programming package. An example model for the GAMS system using the binary solver GAMS/ZOOM is given in the appendix.

While model (8) will generate an optimal solution to the estimation problem, the solution of a mixed integer programming model with $K \times N$ binary variables might require considerable computational effort. We will therefore develop other, approximate techniques, which require less time to solve.

The formulation of a mixed integer model was necessitated by the use of the minimum operator in function (1). This function can be interpreted as measuring the distance of the alternative in question (X_n) to the reference point using the Tchebycheff norm. Since the Tchebycheff norm is a limiting case of the ℓ_p norm for $p = \infty$, it is also possible to approximate this function by an ℓ_p norm using a large value of p (Fortuna/Krus, 1984). This leads to the following nonlinear programming formulation:

Several estimation procedures for utility functions are based on similar inputs (e.g. Srinivasan/Shocker, 1973; Jacquet-Lagreze/Siskos, 1982). These techniques proceed in two steps: first, the amount of contradiction of a preference representation with single preference statements is defined in cardinal form, and then these contradictions are aggregated to a total measure of difference, which is minimized.

The approach outlined above can be formalized as follows: Preference statements made by the user form a relation R, where iRj represents the fact that the user has stated that he/she prefers alternative X_i to alternative X_j. It should be noted that relation R need not fulfil the usual requirements for preference relations, otherwise the decision problem would already be solved. Specifically, R will not usually be complete, and need not even be transitive. However, the estimation process will not yield a perfect fit if R contains elements which actually violate transitivity, i.e. at least three elements for which iRj, jRl and lRi holds. If it contains iRj and jRl, but neither iRl nor lRi, a perfect fit is still possible.

The individual contradiction with one preference statement can then be defined as:

$$d_{ij} = \max(v(X_j) - v(X_i);\ 0) \text{ for } iRj \tag{3}$$

where $v(X_n)$ represents the evaluation of alternative X_n. Function v might, in general terms, correspond to an additive utility function, a scalarizing function like (1) or (2) or any other function that generates a cardinal evaluation of alternatives. Equation (3) can be reformulated as a constraint of a mathematical programming problem as:

$$\begin{aligned} &v(X_i) + d_{ij} \geq v(X_j) \quad \forall iRj \\ &d_{ij} \geq 0 \end{aligned} \tag{4}$$

We thus obtain the following general mathematical programming problem for fitting a preference representation v:

$$\begin{aligned} &\text{minimize } F(d_{ij} : iRj) \\ &v(X_i) + d_{ij} \geq v(X_j) \ \ \forall iRj \\ &d_{ij} \geq 0 \end{aligned} \tag{5}$$

where F is some function used to aggregate the individual contradictions d_{ij}. A common choice for F in estimating utility functions is merely the sum of deviations.

Using also the sum of deviations as an objective function, we obtain an estimation model for reference points by substituting scalarizing functions (1) or (2) for the general evaluation function v in model (5). For simplicity, all further expositions will use (1), but extension to the other function (2) is straightforward. Model (5) thus becomes:

$$\begin{aligned} &\text{minimize } \sum_{i,j:iRj} d_{ij} \\ &\min_k(x_{i,k} - \overline{x_k}) + d_{ij} \geq \min_k(x_{j,k} - \overline{x_k}) \ \ \forall iRj \\ &\sum_k \overline{x_k} = K \\ &\overline{x_k} \geq 0 \\ &d_{ij} \geq 0 \end{aligned} \tag{6}$$

The second constraint in (6) provides a scaling for the reference levels. While, in contrast to additive utility functions, the goodness of fit of a scalarizing function based on the Tchebycheff norm does not depend on the scaling of the reference levels, acceptance of proposed reference levels by the user can be improved by providing "realistic" levels. Assuming that the feasible interval in all criteria is scaled to the interval $(0, 1)$, this constraint, together with the non-negativity condition on the $\overline{x_k}$'s, will in most instances lead to a reference level which is not feasible, but also not too far "beyond imagination".

Unfortunately, problem (6) is not a straightforward mathematical programming problem because of the minimum function in the constraints. In points in which the minimum difference shifts from one criterion to another, the derivative of this function does not exist.

Some mathematical programming packages, however, do allow the use of minimum functions in constraints and try to solve such problems by standard nonlinear programming techniques. The first approach to reference point estimation analyzed in this study is such a direct implementation of (6) using the GAMS system (Brooke et al, 1989) for model formulation and GAMS/MINOS as a nonlinear solver. The corresponding GAMS model is given in the appendix.

A minimum function can also be represented by a set of constraints using binary variables. The minimum of K numbers can be characterized by two properties:

- it is less than or equal to all the numbers
- it is greater than or equal to at least one of the numbers.

Using K binary variables λ_{nk}, we can represent the value $z_n = \min_k(x_{n,k} - \overline{x_k})$ by the following set of constraints:

$$
\begin{array}{lcll}
z_n & \leq & (x_{n,1} - \overline{x_1}) & \\
 & \vdots & & (a) \\
z_n & \leq & (x_{n,K} - \overline{x_K}) & \\
 & & & \\
z_n & \geq & (x_{n,1} - \overline{x_1}) - \lambda_{n,1} M & \\
 & \vdots & & (b) \\
z_n & \geq & (x_{n,K} - \overline{x_K}) - \lambda_{n,K} M & \\
 & & & \\
\sum_k \lambda_{n,k} & \leq & K - 1 & (c)
\end{array}
\tag{7}
$$

In (7), M is a suitably large constant. Part (a) of (7) states that the minimum value z_n should be smaller than or equal to all the values $(x_{n,k} - \overline{x_k})$. If the corresponding $\lambda_{n,k}$ is zero, part (b) forces z_n to be larger than or equal to $(x_{n,k} - \overline{x_k})$. If $\lambda_{n,k}$ is one, z_n need only be larger than a value that is considerably smaller than $(x_{n,k} - \overline{x_k})$, so that part of the constraint becomes ineffective. Part (c) finally guarantees that at least one $\lambda_{n,k}$ is actually zero. z_n will therefore be equal to the minimum of the $(x_{n,k} - \overline{x_k})$' s.

$$
\begin{aligned}
&\text{minimize} \sum_{i,j:iRj} d_{ij} \\
&[\textstyle\sum_k (x_{i,k} - \overline{x_k})^p]^{1/p} + d_{ij} \geq [\textstyle\sum_k (x_{j,k} - \overline{x_k})^p]^{1/p} \quad \forall iRj \\
&\textstyle\sum_k \overline{x_k} = K \\
&d_{ij} \geq 0 \\
&\overline{x_k} \geq 0
\end{aligned}
\tag{9}
$$

The solution of (9), which is only an approximation of the true optimum, can further be improved. Since the efficiency of the branch and bound technique for mixed integer programming can be enhanced by providing a good starting integer solution to the model, it is possible to combine both approaches. In the combined approach, the solution of the nonlinear programming model (9) is used as an initial guess for the mixed integer solution, which helps to reduce the branching tree and thus speed up the solution process.

However, the solution of (9) cannot be directly used as a starting point for (8), since it might be an interior solution to the linear problem. But it is possible to "translate" (and at the same time improve) that solution into a solution of the linear programming problem underlying (8) using a linear program. The following algorithm represents the entire process:

1 Solve problem (9), denote the solution by $\overline{x}^0$.

2 From $(x_{n,k} - \overline{x_k}^0)$, determine for every alternative X_n which criterion is the limiting factor in its evaluation according to (1) and fix the variables $\lambda_{n,k}$ in (8) accordingly, i.e.:

$$
\lambda_{n,k} = \begin{cases} 0 & \text{if } x_{n,k} - \overline{x_k} = \min_k (x_{n,k} - \overline{x_k}) \\ 1 & \text{otherwise} \end{cases}
$$

3 Solve model (8) as a continuous linear programming model in variables $\overline{x_k}$. The resulting solution is an integer solution of (8), since all binary variables were previously fixed to zero or one, fulfilling also the constraints $\sum_k \lambda_{n,k} \leq K - 1$.

4 Solve model (8) as a mixed integer programming model in variables $\overline{x_k}$ and $\lambda_{n,k}$, using the solution from step 3 as a starting point.

In the GAMS system used for the computational experiments underlying this paper, it is not possible to introduce such an "incumbent" integer solution directly into the branch and bound process. However, it can be indirectly introduced as follows: let z^0 denote the objective value of the linear programming solution generated. By adding a constraint of the form

$$\sum_{i,j:iRj} d_{ij} \leq z^0 \cdot (1+\epsilon) \tag{10}$$

to the model, all nodes with larger objective values are eliminated from the search tree. Using a slightly increased value $z^0 \cdot (1+\epsilon)$ rather than z^0 will prevent the model from becoming infeasible due to round-off errors. While the branch and bound process still has to find the corresponding integer solution, this solution is usually obtained quickly in GAMS/ZOOM.

Computational experiments shown in the next section have indicated that while this heuristic considerably reduces solution times in some instances, the resulting computational effort is still rather high and the heuristic fails to reduce computation time at all in other cases. Therefore, an entirely different strategy was also studied. The constraints in (5) serve mainly as a way of defining the deviation variables d_{ij}. This definition can be incorporated into the objective function. The resulting problem is the minimization of a continuous, but nondifferentiable function under just a few constraints: the scaling and the non-negativity conditions on the $\overline{x_k}$'s. The scaling condition can be eliminated by substituting

$$\overline{x_K} = K - \sum_{k=1}^{K-1} \overline{x_k} \tag{11}$$

and the non-negativity condition can be taken into account by adding a penalty term for negative values.

For the computational experiments described in the next section, a FORTRAN routine from Späth (1974), which is based on an extension to the "direct search" algorithm of Hooke and Jeeves (1961; c.f. Bazaraa/Shetty, 1979) was used. The resulting program is also given in the appendix.

4 Computational Results[1]

4.1 Experimental Setup

The estimation techniques developed above were used in computational experiments. For each experiment, a test problem consisting of 10 alternatives and 6 criteria was randomly generated. The preference statements were specified as a complete order of all alternatives in the form $X_1 \succ X_2 \succ \ldots \succ X_{10}$. For each test problem, the following estimation problems were solved:

MIN	A direct implementation of model (6) in GAMS, using the nonlinear solver GAMS/MINOS.
MIP-10	An implementation of model (8) in GAMS, solved with GAMS/ZOOM to within 10 % of optimality (i.e. to an integer solution which is at most 10 % worse than the optimal solution).
MIP-OPT	The same model as MIP-10, but solved to optimality.
DIRECT	The direct search procedure.
MIX-10	An implementation of the three-step procedure outlined above. Model (9) in step 1 was solved using GAMS/MINOS, model (8) in steps 3 and 4 with GAMS/ZOOM. The integer solution process of model (8) was terminated within 10% of optimality.
MIX-OPT	The same as above, but solving the final mixed integer problem to full optimality.

[1]I would like to thank Werner Buser for running most of the experiments described here

All experiments were run on a personal computer with a 25MHz Intel 486 processor using GAMS/386 with its corresponding solvers.

4.2 Results: Calculation Times

Table (1) contains the solution times (T) and solution quality (Q) for ten test problems and the strategies defined above. Solution times and quality measures for strategy MIX-10 are given separately for all three stages. Since the first two stages for MIX-10 and MIX-OPT are identical, only the last stage is given for MIX-OPT. Solution quality is measured for strategy i as $z_i/z_{mip-opt}$, where z_i is the sum of deviations for the solution generated by the strategy under consideration and $z_{mip-opt}$ is the optimum value obtained by solving the mixed integer problem. Times are given in seconds.

The last two rows of table 1 provide the means and standard deviations of the respective columns.

Several points should be noted about these results:

- The mixed integer models lead in almost all cases to solution times which are not acceptable for interactive systems.
- Contrary to "common wisdom" in mixed integer programming, the time needed to prove optimality of a solution is rather short compared to the time needed to find a good integer solution for the problem. Relaxing the quality requirement, which is usually recommended for mixed integer programming, does not reduce solution times significantly.
- Also contrary to expectations, providing a bound on the solution (phase 3 of strategies MIX-10 and MIX-OPT) in many instances even increased the solution time as compared to the strategies MIP-10 and MIP-OPT, which directly solve the original mixed integer problem. In other instances, though, the combined strategy led to a significant improvement in the speed of the solution process.

Problem	MIN		MIP-10		MIP-OPT		DIRECT	
	Q	T	Q	T	Q	T	Q	T
1	2.05	1.43	1	245	1	245	1	0.77
2	1.07	2.09	1	6190	1	6192	1.32	0.66
3	3.07	0.77	1	4884	1	4887	1	0.93
4	1.73	1.32	1	394	1	394	1.44	0.72
5	1	1.26	1	714	1	915	1	1.1
6	1.12	1.37	1	342	1	354	1	0.77
7	1	0.82	1	330	1	330	1	0.99
8	1.33	1.10	1	1137	1	1137	1	0.82
9	1.30	2.96	1.03	1276	1	1302	1	0.72
10	1	1.10	1	231	1	231	1	0.77
μ	1.47	1.42	1	1574	1	1599	1.08	0.83
σ	0.66	0.65	0.01	2142	0.00	2134	0.16	0.14

Problem.	MIX-10/1		MIX-10/2		MIX-10/3		MIX-OPT/3	
	Q	T	Q	T	Q	T	Q	T
1	2.38	3.95	2.18	0.99	1	146	1	146
2	1.39	9.88	1.27	0.71	1	16307	1	16310
3	2.14	1.81	1.30	0.77	1	1573	1	1573
4	2.52	2.09	1.06	0.98	1	167	1	167
5	1.86	6.76	1.42	0.93	1.04	1891	1	1966
6	1.78	4.56	1.31	0.76	1	1408	1	1694
7	2.37	7.03	1.87	0.82	1	526	1	526
8	1.24	3.74	1.21	0.71	time limit		time limit	
9	1.56	6.31	1.56	0.88	1	1504	1	1504
10	2.28	4.06	1.001	0.77	1.001	159	1.001	159
μ	1.95	5.02	1.42	0.83	1	2631	1	2672
σ	0.45	2.48	0.37	0.11	0.01	5175	0.00	5167

Table 1: Results for ten experiments

- All three approaches, which were not based on integer programming provided usable solutions in an acceptable time. It should be pointed out that for estimation problems in general, it is not really necessary to obtain the best fit possible, but rather to find a good approximation of data which is significantly disturbed anyway (Klein et al., 1985).
- As expected, the additional linear programming phase significantly improved the results of the nonlinear approximation, while requiring little computational effort.
- Among the three approximation approaches (MIN, DIRECT and MIX/1), the simple "Direct search" procedure in most cases outperformed the other methods. It also has the best average solution quality and the smallest variance.

4.3 Results: Perfect Fits

The results of the last subsection clearly indicate that a heuristic method must be used to solve the estimation problem in an interactive system and that the direct search approach seems to be the most promising heuristic. For use in actual application, it is also important whether a heuristic will find a perfect fit solution, i.e. a solution in which the preferences specified by the user are completely reproduced, if such a solution exists. This question was analyzed in a second set of experiments. Since direct search was especially successful in the first set of experiments, only this method was tested in the second set.

The following method was used for this test: using again randomly created test problems with 10 alternatives and 6 criteria, successively more complex preference statements were included in R until the first problem was found for which the direct search method could not find a perfect fit, i.e. a solution with $z_{direct} > 0$. Preference statements in R were generated as follows: first, unconnected preferences were generated in the order $X_1 \succ X_2, X_3 \succ X_4$ etc. After all such preference statements were included, connecting preferences were generated in the order $X_2 \succ X_3, X_4 \succ X_5$ and so on. The first problem for which the heuristic could not find a perfect fit was then solved using a

mixed integer program to verify whether a perfect fit solution existed or not. Table 2 presents the results from 20 randomly generated test problems.

Number of Problems	Results
12	The solution obtained with the mixed integer program was the same as with the heuristic
6	No perfect fit existed, but the mixed integer program found a better solution than the heuristic
1	The mixed integer program found a perfect fit
1	The mixed integer program failed to find a solution within 1 hour of calculation time

Table 2: Results for "Perfect fit"-test

The average performance of the heuristic (defined again as z_{direct}/z_{mip}) in the 6 problems in which it generated a worse solution than the mixed integer program, was 2.25; over all 18 problems of the first two categories it was 1.41. The worse relative performance compared to the results in table 1 is due to the fact that in most of the problems, the absolute value of the objective was very low, so a small absolute increase produced a large loss in relative performance.

5 Conclusions

Reference point estimation as studied in this paper is used to provide a set of initial reference levels, which will later be modified during an interactive decision procedure. The results presented in the last section clearly indicate that for this task, a heuristic method based on direct search is the best approach. This technique requires small computational effort to generate a very good solution, in many instances even the optimal solution. Furthermore, this approach can be easily implemented; the original FORTRAN procedure given in (Späth, 1974) is just 68 lines of code. These considerations have led to the use of this method in an integrated system for multiattribute decision making currently under development.

References

Bazaraa, M.; Shetty, C.M. (1979): *Nonlinear Programming, Theory and Algorithms.* J. Wiley & Sons, New York.

Belkeziz, K.; Pirlot, M. (1991): *Proper efficiency in nonconvex vector-maximization-problems.* European Journal of Operational Research 54: 74–80.

Brooke, A.; Kendrick, D.; Meeraus, A. (1989): *GAMS - A User's Guide.* The Scientific Press, San Francisco.

Charnes, A.; Cooper, W.W.; Ferguson, R.O. (1955): *Optimal Estimation of Executive Compensation by Linear Programming.* Management Science 1: 138–151.

Charnes, A.; Cooper, W.W. (1961): *Management Models and Industrial Applications of Linear Programming.* Vol. I. J. Wiley & Sons, New York.

Evans, G. (1984): *An Overview of Techniques for Solving Multiobjective Mathematical Programs.* Management Science 30: 1268–1282.

Fischer, G. (1979): *Utility Models for Multiple Objective Decisions: Do they Accurately Represent Human Preferences?* Decision Sciences 10: 451–479.

Fortuna, Z.; Krus, L. (1984): *Simulation of an Interactive Method Supporting Collective Decision Making Using a Regional Development Model.* In: Grauer, M.; Wierzbicki, A.P. (Eds.): Interactive Decision Analysis. Springer, Berlin et al.: 202–209.

Gershon, M. (1984): *The role of weights and scales in the application of multiobjective decision making.* European Journal of Operational Research 15: 244–250.

Habenicht, W. (1990): *Neuere Entwicklungen auf dem Gebiet der Vektoroptimierung - ein Überblick.* Arbeitspapier 1/90, Institut für Betriebswirtschaftslehre - Lehrstuhl für Industriebetriebslehre, Universität Hohenheim.

Hooke, R.; Jeeves, T.A. (1961): *"Direct Search" Solution of Numerical and Statistical Problems.* Journal of the ACM 8: 212–229.

Horsky, D.; Rao, M.A. (1984): *Estimation of Attribute Weights from Preference Comparisons.* Management Science 30: 801–822.

Huber, G. (1974): *Methods for Quantifying Subjective Probabilities and Multi-Attribute Utilities.* Decision Sciences 5: 430–458.

Jacquet-Lagreze, E.; Siskos, J. (1982): *Assessing a set of additive utility functions for multicriteria decision-making, the UTA method.* European Journal of Operational Research 10: 151–164.

Jaskiewicz, A.; Slowinski, R. (1991): *An Interactive Method for Multiple Objective Nonlinear Programming Based on Outranking and Ordinal Regression.* Paper presented at the IIASA Workshop on User-Oriented Methodology and Techniques of Decision Analysis and Support, Serock near Warsaw.

Klein, G.; Moskowitz, H.; Mahesh, S.; Ravindran A. (1985): *Assessment of Multiattribute Measurable Value and Utility Functions via Mathematical Programming.* Decision Sciences 16: 309–324.

Korhonen, P.; Wallenius, J.; Zionts, S. (1984): *Solving the Discrete Multiple Criteria Problem Using Convex Cones.* Management Science 30: 1336–1345.

Majchrzak, J. (1991): *Pairwise Comparisons in Decision Support for Multicriteria Choice Problems.* Paper presented at the IIASA Workshop on User-Oriented Methodology and Techniques of Decision Analysis and Support, Serock near Warsaw.

Mustafi, C.K.; Xavier, M.J. (1985): *Mixed-Integer Linear Programming Formulation of a Multi-Attribute Threshold Model of Choice.* Journal of the Operational Research Society 36: 935–942.

Roy, B.; Vincke, P. (1981): *Multicriteria analysis: survey and new directions.* European Journal of Operational Research 8: 207–218.

Schoemaker, P.; Waid, C. (1982): *An Experimental Comparison of Different Approaches to Determining Weights in Additive Utility Models.* Management Science 28: 182–196.

Shin, W.; Ravindran, A. (1991): *Interactive Multiple Objective Optimization: Survey I: Continuous Case.* Computers and Operations Research 18: 97–114.

Späth, H. (1974): *Algorithmen für multivariable Ausgleichsmodelle.* Oldenbourg, München.

Srinivasan, V.; Shocker, A. (1973): *Estimating the Weights for Multiple Attributes in a Composite Criterion Using Pairwise Judgements.* Psychometrika 38: 473–493.

Vanderpooten, D. (1989): *The Use of Preference Information in Multiple Criteria Interactive Procedures.* In: Lockett, A.G.; Islei, G. (Eds.): Improving Decision Making in Organisations. Springer, Berlin et al:. 390–399.

Vincke, P. (1986): *Analysis of multicriteria decision aid in Europe.* European Journal of Operational Research 25: 160–168.

Weber, M. (1985): *A Method of Multiattribute Decision Making with Incomplete Information.* Management Science 31: 1365–1371.

Weidner, P. (1989): *Problems in Models and Methods of Vector Optimization.* Wiss. Schriftenreihe der TU Karl-Marx-Stadt 5: 47–57.

Wierzbicki, A. (1980): *The Use of Reference Objectives in Multiobjective Optimization.* In: Fandel, G.; Gal, T. (Eds.): Multiple Criteria Decision Making - Theory and Application. Springer, Berlin et al.: 468–486.

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

Zanakis, S.; Gupta, S. (1985): *A Categorized Bibliographic Survey of Goal Programming.* Omega 13: 211–222.

Appendix

GAMS model using the minimum operator

```
*****************************************************************
* Reference point estimation
* Method: DNLP
*****************************************************************

* Specification of Options
OPTIONS ITERLIM=100000;
$OFFSYMLIST OFFSYMXREF

* Starting value for random number generator
OPTIONS SEED = 555555;

* Options to limit output
OPTION LIMROW=0, LIMCOL=0;
OPTION SOLPRINT=OFF

* The following sets define the dimensions of the model
SETS
   N  Alternatives  /1*10/
   K  Criteria      /1*6/
   I(N)             /1*9/;

TABLE
 X(n,k) Performance of alternatives
   1
1  0;

* Randomly generate an N*K decision matrix
x(n,k) = UNIFORM(0,1);
DISPLAY X;

FREE VARIABLES
   z   Objective function;
POSITIVE VARIABLES
  xq(k)   Reference Point
  d(i)    Deviations;

* Provide a feasible starting value
xq.l(k) = 1;

* Definition of the model rows
EQUATIONS
  ZF      Objective function
  Pr(i)   Preference statements
  Summe   Sum of xq's;
ZF..
   Z =E= SUM(i,d(i));
Pr(i)..
   SMIN(k, x(i,k)-xq(k)) + d(i) =G=
       SMIN(k, x(i+1,k)-xq(k));
Summe..
   SUM(k, xq(k)) =E= 6;

* Definition and solution of the model
MODEL nmod /ALL/;
SOLVE nmod USING DNLP MINIMIZING Z;

* Output of results
DISPLAY XQ.L;
PARAMETER
   s(n)   Evaluation of alternatives
   dd(n)  Deviations
   sd     Sum of deviations;
s(n)=SMIN(k,x(n,k)-xq.l(k));
dd(n)=0.0;
dd(n+1)$(s(n+1) GT s(n)) = s(n+1)-s(n);
sd=SUM(n, dd(n));
DISPLAY s,dd,sd;
```

GAMS model to solve the mixed integer problem

```
*****************************************************************
* Reference point estimation
* Method: Mixed Integer / Optimum
*****************************************************************

* Specification of Options
OPTIONS ITERLIM=10000000, RESLIM=60000, OPTCR=0;
$OFFSYMLIST OFFSYMXREF

* Starting value for random number generator
* (different values were used for different experiments
OPTIONS SEED = 555555;

* Options to limit output
OPTION LIMROW=0, LIMCOL=0;
OPTION SOLPRINT=OFF

* The following sets define the dimensions of the model
SETS
   N  Alternatives  /1*10/
   K  Criteria      /1*6/
   I(N)             /1*9/;

TABLE
 X(n,k) Performance of alternatives
   1
1  0;

* Randomly generate an N*K decision matrix
x(n,k) = UNIFORM(0,1);
DISPLAY X;

* Variables
SCALAR
  M /10.0/     The Big M constant;
FREE VARIABLES
   z           Objective value
   y(n)        Minima;
POSITIVE VARIABLES
  xq(k)        Reference Point
  d(n)         Deviations;
BINARY VARIABLES
  l(n,k)       Switches;

* Definition of the model rows
EQUATIONS
  ZF           Objective function
  P1(n)        Preference statements
  DefY1(n,k)   Minimum operations, part 1
  DefY2(n,k)   Minimum operations, part 2
  LS(n)        Sum of lambdas
  Summe        Sum of xq's;

ZF..
   Z =E= SUM(i,d(i));
P1(n)$(ORD(n) LT 10)..
   y(n) + d(n) =G= y(n+1);
DefY1(n,k)..
   y(n) =L= x(n,k) - xq(k);
DefY2(n,k)..
   y(n) =G= x(n,k) - l(n,k)*M - xq(k);
LS(n)..
   SUM(k, l(n,k)) =E= 5;
Summe..
   SUM(k, xq(k)) =E= 6;

* Definition and solution of the model
MODEL TEST /ALL/;
TEST.OPTFILE=1;
SOLVE TEST USING MIP MINIMIZING Z;
```

```
* Output of results
DISPLAY XQ.L;
PARAMETER
   s(n)   Evaluation of alternatives
   dd(n)  Deviations
   sd     Sum of deviations;
s(n)=SMIN(k,x(n,k)-xq.l(k));
dd(n)=0.0;
dd(n+1)$(s(n+1) GT s(n)) = s(n+1)-s(n);
sd=SUM(n, dd(n));
DISPLAY s,dd,sd;
```

GAMS model for the combined strategy

```
*****************************************************************
* Reference point estimation
* Method: Combined
*****************************************************************

* Specification of Options
OPTIONS ITERLIM=10000000, RESLIM=60000, OPTCR=0.1;
$OFFSYMLIST OFFSYMXREF

* Starting value for random number generator
* (different values were used for different experiments
OPTIONS SEED = 555555;

* Options to limit output
OPTION LIMROW=0, LIMCOL=0;
OPTION SOLPRINT=OFF

* The following sets define the dimensions of the model
SETS
   N  Alternatives  /1*10/
   K  Criteria      /1*6/
   I(N)             /1*9/;

TABLE
   X(n,k) Performance of alternatives
   1
1  0;

* Randomly generate an N*K decision matrix
x(n,k) = UNIFORM(0,1);
DISPLAY X;
```

```
* The nonlinear approximation model of step 1
SCALAR
  q /5/   Exponent of approximation;
FREE VARIABLES
   z      Objective function;
POSITIVE VARIABLES
  xq(k)   Reference Point
  d(i)    Deviations;

* Provide a feasible starting value
xq.l(k) = 1.0;

* Definition of the model rows
EQUATIONS
  ZF      Objective function
  Pr1(i)   Preference statements
  Summe   Sum of xq's;
ZF..
   Z =E= SUM(i,d(i));
Pr1(i)..
   (SUM(k, (xq(k)-x(i,k))**q))**(1/q) - d(i) =l=
       (SUM(k, (xq(k)-x(i+1,k))**q))**(1/q);
Summe..
   SUM(k, xq(k)) =E= 6.0;

MODEL Step1 /ZF, Pr1 ,Summe/;
SOLVE Step1 USING NLP MINIMIZING Z;

* Display results from step 1
PARAMETER
   s(n)   Evaluation of alternatives
   dd(n)  Deviations
   sd     Sum of deviations;
s(n)=SMIN(k,x(n,k)-xq.l(k));
dd(n)=0.0;
dd(n+1)$(s(n+1) GT s(n)) = s(n+1)-s(n);
sd=SUM(n, dd(n));

DISPLAY "=====> Ergebnis Phase 1:";
DISPLAY xq.l;
DISPLAY s,dd,sd;

* Additional variables used in steps 3 and 4
SCALAR
  M /10.0/;
FREE VARIABLES
   y(n)      Minima;
BINARY VARIABLES
  l(n,k)     Switches;

* Step 2: fix values of lambda according to the first solution
l.fx(n,k) = 1;
PARAMETER
   Mini(n)   Minimal values
   xq1(k)    First solution;

Mini(n)=SMIN(k,x(n,k)-xq.l(k));
xq1(k)=xq.l(k);
l.fx(n,k)$((x(n,k)-xq.l(k)) EQ Mini(n)) = 0;

* Definition of the model used in steps 3 and 4
EQUATIONS
  P1(n)      Preference statements
  DefY1(n,k) Minimum operations, part 1
  DefY2(n,k) Minimum operations, part 2
  LS(n)      Sum of lambdas;
P1(n)$(ORD(n) LT 10)..
   y(n) + d(n) =G= y(n+1);
DefY1(n,k)..
   y(n) =L= x(n,k) - xq(k);
DefY2(n,k)..
   y(n) =G= x(n,k) - l(n,k)*M - xq(k);
```

```
LS(n)..
   SUM(k, l(n,k)) =E= 5;

* Model of step 3
MODEL Step3 /ZF, P1, DefY1, DefY2, LS, Summe/;
SOLVE Step3 USING MIP MINIMIZING Z;

* Display results from step 3
s(n)=SMIN(k,x(n,k)-xq.l(k));
dd(n)=0.0;
dd(n+1)$(s(n+1) GT s(n)) = s(n+1)-s(n);
sd=SUM(n, dd(n));

DISPLAY "=====> Ergebnis Phase 2:";
DISPLAY xq.l;
DISPLAY s,dd,sd;

* Define upper bound on objective for step 4
PARAMETER
   Z1   Objective value from step 3;
Z1=z.l;
EQUATIONS
   ZLim  Limit on objective;
ZLim..
   Z =L= Z1*1.01;

* Make binary variables switchable again
l.lo(n,k)=0;
l.up(n,k)=1;

MODEL Step4 /ZF, P1, DefY1, DefY2, LS, Summe, ZLim/;
Step4.OPTFILE=1;
SOLVE Step4 USING MIP MINIMIZING Z;

* Display results from last step
s(n)=SMIN(k,x(n,k)-xq.l(k));
dd(n)=0.0;
dd(n+1)$(s(n+1) GT s(n)) = s(n+1)-s(n);
sd=SUM(n, dd(n));

DISPLAY "=====> Ergebnis Phase 3:";
DISPLAY xq.l;
DISPLAY s,dd,sd;
```

FORTRAN program for the direct search method

```
C FORTRAN Program for reference point estimation
C using the direct search heuristic

      REAL XD(10,6)
      REAL X(6),SX,H,R,DT,SS
      INTEGER N,ITMAX,DS
      INTEGER*2 IHR1,IHR2,IMIN1,IMIN2,ISEC1,ISEC2,I1001,I1002
      COMMON /DATEN/ XD,DS
      EXTERNAL S
C
      OPEN(1,FILE='DATEN')
      DO 10 i=1,10
         READ(1,1010) (XD(I,J),J=1,6)
10       CONTINUE
C
      DO 20 I=1,10
         WRITE(*,1010) (XD(I,J),J=1,6)
20       CONTINUE
      N=5
      DO 30 I=1,6
         X(I)=1.0
30       CONTINUE
      SX=1.0
      H=0.5
      R=0.9
      HMIN=0.000001
      ITMAX=10000
      DS=0
      CALL GETTIM(IHR1,IMIN1,ISEC1,I1001)
      CALL SEARCH (N,X,SX,H,R,HMIN,S,ITMAX)
      SS=0.0
      DO 40 I=1,5
         SS=SS+X(I)
40       CONTINUE
      X(6)=6.0-SS
      CALL GETTIM(IHR2,IMIN2,ISEC2,I1002)
      DT=FLOAT(ISEC2-ISEC1)+0.01*FLOAT(I1002-I1001)
      DT=DT+60.0*FLOAT(IMIN2-IMIN1)+3600.0*FLOAT(IHR2-IHR1)
      WRITE(*,*) 'Optimalwert von S=',SX
      WRITE(*,*) 'An der Stelle x=...'
      WRITE(*,*) (X(i),I=1,6)
      WRITE(*,*) 'Nach ',ITMAX,' Iterationen'
      WRITE(*,*) 'Zeit: ',DT,' Sekunden'
      STOP

1010  FORMAT(6F12.10)
      END

      FUNCTION S(X)
C   Function to be minimized
      REAL X(6)
      REAL SD,Y(10),H,XMIN, XX(6), SS
      REAL XD(10,6)
      INTEGER DF
      COMMON /DATEN/ XD,DF

      SS=0.0
      DO 5 I=1,5
         XX(I)=X(I)
         SS=SS+XX(I)
 5       CONTINUE
      XX(6)=6.0-SS
      IF (DF.GT.1) WRITE(*,*) 'SS=',SS
      DO 20 I=1,10
         XMIN=XD(I,1)-XX(1)
         DO 10 J=2,6
            H=XD(I,J)-XX(J)
            IF (DF.GT.1) WRITE(*,*) 'Wert(',I,',',J,')=',h
            IF (H.LT.XMIN) XMIN=H
10          CONTINUE
```

```
         Y(I)=XMIN
20       CONTINUE
      SD=0.0
      DO 30  I=1,9
         IF(Y(I).LT.Y(I+1)) SD=SD+(Y(I+1)-Y(I))
30       CONTINUE
      DO 40 I=1,6
         IF (XX(I).LT.0.01) SD=SD+100.0*(0.01-XX(I))
40       CONTINUE
      S=SD
      IF (DF.GT.0) THEN
         WRITE(*,*) 'Aufruf mit Werten...'
         WRITE(*,*) (X(i),I=1,6)
         WRITE(*,*) 'Minima:'
         WRITE(*,*) (Y(i),I=1,10)
         WRITE(*,*) 'Ergebnis ',sd
         ENDIF
      RETURN
      END
```

Subroutine SEARCH is published in Späth (1974), pp.109-110. It is defined as

```
    SUBROUTINE SEARCH (N,X,SX,H,R,HMIN,S,ITMAX)
```

and has the parameters

N	Number of variables
X	Vector of variables
SX	Optimal value of the objective funciton
H	Initial step size
R	Step size multiplier used to reduce the step size after each major iteration
HMIN	Minimum step size (termination criterion)
S	Function to be evaluated
ITMAX	Maximum number of function calls (termination criterion)