Working Paper

Methodology and a Modular Tool for Multiple Criteria Analysis of LP Models

Marek Makowski

WP-94-102
December 1994

Methodology and a Modular Tool
for Multiple Criteria Analysis
of LP Models

Marek Makowski

WP-94-102
December 1994

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria

Telephone: +43 2236 807 □ Fax: +43 2236 71313 □ E-Mail: info@iiasa.ac.at

Foreword

For many policy making problems, the underlying processes are of a physical nature. This makes it possible to model those processes using the extensive knowledge and experience with such phenomena. However, for decision support, we also need criteria to distinguish between alternative polices and, particularly, we need means to find our way in the usually huge set of possible policies. For such purposes the method of aspiration-led multi-criteria analysis has proved to be useful.

In the present paper, it is demonstrated how this methodology can be used for linear programming models. It is also demonstrated that a standard tool can be very helpful in using this methodology. In such cases the standard tool is used as a building block in a problem specific decision support system.

Abstract

This paper provides an overview of the methodology of the multiple-criteria model analysis for decision support. In particular, different approaches to the analysis of a model using multiple-objective optimization are compared. One of the most successful methods, namely aspiration-reservation led decision support, is presented in more detail.

The implementation of a Decision Support System (DSS) has to be problem specific but reusable modular software provides substantial help in actual implementations. A DSS for regional water quality management serves as an illustration of the application of modular software tools. This paper presents one of such software tools, called LP-MULTI, especially designed and implemented to be helpful for the analysis of multiobjective problems described by linear models. The paper discusses the methodology applied to LP-MULTI and provides the necessary details of the implementation.

Key Words: decision support, multi-criteria programming, aspiration-reservation-led decision support, reference point, linear programming, applications of multi-criteria programming, regional water quality management.

Contents

Methodology and a Modular Tool for Multiple Criteria Analysis of LP Models

Marek Makowski

1 Introduction

Decision making often requires analysis of large amounts of data and complex relations. In such cases, an analysis of a mathematical model can support rational decision making. Computerized tools designed and implemented for such purposes are called Decision Support Systems (DSS). A DSS, which is typically a problem specific tool, helps in the evaluation of consequences of given decisions and advises what decision would be the best for achieving a given set of goals. In a traditional optimization approach, only one goal was used as optimized performance index and constraints were set for other goals. Such an approach has serious limitations since most real life problems are indeed multicriteria. Therefore a multiple-criteria model analysis (MCMA) is more widely used. The advantages of using a MCMA are not only because of its ability of handling several goals. The main advantage of a proper implementation of MCMA is due to the way it is used. Namely, it helps to analyze the problem rather than providing a single optimal solution.

Although a DSS must be problem specific, there exist both methodology and tools applicable for many different classes of problems. In order to spread the scope of potential applications and to increase the ability to meet specific needs of users, there is a need to modularize the architecture of DSSs. A modular DSS consists of a collection of tools rather than one closed system, thus allowing both efficient problem-specific analyses and efficient development and maintenance of the needed software. The paper describes one such tool, called LP-MULTI, especially designed and implemented to be helpful for analysis of multiobjective problems described by linear models.

The discussion below is based on the results of the methodological research and on its applications. One of the recent applications, namely a DSS for the Regional Water Quality Management Case Study for the Nitra River Basin – which will be referred to as RWQM in this paper – is summarized in Section 2. The RWQM is aimed at the design and implementation of a DSS for supporting a selection of a set of alternatives for waste water treatment plants in order to improve the water quality of a river basin, or of a larger region consisting of a number of river basins.

The remainder of this paper is organized as follows. Section 3 provides an overview of the methodological background of the model based decision analysis and support. Section 4 discusses in more detail one of the most successful approaches to decision analysis, namely the group of methods called the aspiration based decision support. The methodology applied in LP-MULTI is summarized in Section 5 and the user guide for LP-MULTI is provided in Section 6. The implementation of modular software tools is illustrated by an outline of the DSS for the Nitra Case Study presented in Section 7. The two following sections contain conclusions and references. Finally, Appendices A and B provide selected implementation details of LP-MULTI and information about availability of the software.

2 The Regional Water Quality Management Case Study

We have selected as an example a DSS for the Nitra River Basin Water Quality Management (RWQM) Case Study, which has been recently developed at IIASA. This example is well documented in [SMPK94]) and serves as a good illustration for more general problems of design and implementation of a DSS. This paper is not aimed at giving a description of the problem but rather at describing its brief characteristics enabling us to use the problem as an illustration of issues related to the design and implementation of a DSS. For this purpose the problem can be briefly characterized as follows.

We consider a river basin or a larger region composed of several basins where the water quality is extremely poor. We consider also a set of waste water treatment plants (either existing or to be possibly constructed) and, at each plant, some technology (which may be composed of a set of technologies to be selected out of a bigger given set of possible technologies) that can be implemented in order to improve the water quality in a region. The traditional optimization based approach to solving such a problem consists of looking for a set of plants and technologies whose implementation would result in maintaining prescribed water quality standards at minimum cost. However, the application of such an approach would in this case, as in many other cases, result in an infeasible solution because of the costs involved. Therefore another approach to decision support has been applied for the Nitra River Basin. Namely, a system of models has been developed for supporting a decision making process. The system is composed of simulation and single criterion dynamic programming models (cf [SMPK94]) and of an aspiration-led multiple criteria optimization model (cf [MSW95]) and it is envisaged to serve two purposes:

- as a decision-aid tool for analysts and high-level decision makers in establishing the effluent and/or ambient water quality standards and the associated appropriate economic instruments that can be enforced to control the waste water discharges.
- to aid in evaluation of alternative treatment strategies (technologies in treatment plants) and/or in selecting the most appropriate strategy based on the water quality standards and on the costs (capital investment and operational).

The DSS, composed of the models outlined above, uses a data base, which contains several sets of data such as geographical, hydrological, morphological, waste water discharges, and data related to different types of waste water treatment (costs, efficiencies).

The detailed documentation of the RWQM model used for the multiple criteria's implementation of the DSS is provided in [MSW95] and the description of the software tools used in this implementation is given in Section 7. Here we only summarize the characteristics of the underlying mathematical programming problem. The corresponding mixed integer programming problem has less (because the exact numbers depend on selected criteria and their status) than 100 binary decision variables, about 700 additional variables and about 800 logical and physical constraints. The considered criteria include three economic criteria (total annual, investment, operational and maintenance costs) and several environmental criteria (either maximal concentrations of pollutants or maximal violation of environmental standards).

3 Methodological background

The methodological background of decision making and decision support is a fast growing area of research and applications. The discussion of the related issues is far beyond

the scope of this paper. A reader interested in general methodological issues may refer to good overviews of the different concepts, provided e.g. by [And89, Kee92, KMW92, KOCZ93, LeW89, Rap89, SpW93, Thi93, Tur93, TvK85, Vin89, WeW93, Yu90]. A large bibliography on different topics related to decision support can be found e.g. in [Mak94a], which summarizes also the author's point of view on design and implementation of model based decision support.

The essence of all interactive methods for decision support is based on the commonly known observation: in typical complex situations an *a priori* specification of either attainable goals or of preferences for discrete alternatives is practically impossible. Therefore most approaches to DSS assume that a Decision Maker (DM) interactively changes goals or preferences upon analysis of feasible solutions obtained for previously specified goals.

There are many approaches to the model based decision analysis and support. In this paper we concentrate on one specific methodology that has proved to be successful in model based DSSs. This is the aspiration-led, interactive multiple-criteria model analysis (MCMA). However, before discussing the MCMA we will briefly present, in the following subsections, problems that are important not only for MCMA but also for a broader class of a model based DSS. Therefore, Section 3.1 provides an overview of different approaches to the model based DSS. Such DSS uses a mathematical programming model that corresponds to a part of the decision making process that is supported by the DSS. An approach to formulation and handling of such models is presented in Section 3.2. MCMA implies some additional requirements for a model specification. Those requirements, together with a summary of general requirements for model formulation, are discussed in Section 3.3. Finally, selected traditional ways of model analysis (excluding MCMA) are summarized in Section 3.4.

3.1 Model-based decision support

A model-based DSS relies on mathematical programming models that can adequately represent decision situations. To represent a decision situation means that the model can be used for predicting and evaluating consequences of decisions, which is a basic functionality of simulation based DSSs. In optimization based DSSs the model is also used to compute decisions that would result in attaining specified goals. A specification of a model to be used within a DSS differs from a specification of a traditional model used for simulation or for single-criterion optimization because of the way the model is used. In traditional approaches a number of constraints are added to the core of the model in order to implicitly define not only feasible but also acceptable solutions. This used to be a must for batch oriented optimization approaches but it should be avoided for a specification of a model that is to be used as a part of a DSS.

Hence in practical applications that deal with medium- and large-scale problems, it is practical to divide specification and generation of the model into two parts and the corresponding stages:

- First, a *core model* is specified and generated. This model contains only a set of constraints that correspond to logical and physical relations between variables.

- Second, during an interactive procedure a DM specifies goals and preferences, including values of objectives that he/she wants to achieve and to avoid. Such a specification usually results in the generation of additional constraints and variables, which are added to the *core model* thus forming an optimization problem.

Such an approach has several advantages over the traditional approach in which both a preferential structure of a user and logical and physical relations between variables are specified and implemented together. Some of the advantages of the two-stage approach are listed below:

- A core model defines implicitly a set of feasible solutions. Feasibility is understood in the sense of logical and physical relations that must always hold. Therefore this part of a model (once the model is verified) should not be modified during analysis of the model.

- A core model has always a non-empty set of feasible solutions. Therefore the debugging of a core model formulation is much easier (than of a traditional optimization model) and can be done via simulation.

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

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

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

- An interactive analysis of the model can be done with the help of modular tools that are not problem specific and can be used for a class of problems, e.g. LP-MULTI can be used for any LP (including MIP) model. Hence, software development is easier because one can reuse whole modules. Moreover, different methodologies and corresponding software modules for interactive analysis can be used without changing a core model formulation.

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

- There is no need to generate soft constraints in the core model. Generation of soft constraints is a sound idea but in practice the handling of a prior specification of soft constraints is cumbersome and therefore rarely used. However, one can easily handle soft constraints within multi-criteria model analysis (cf Section 6.1 for details).

3.2 Core model

In this subsection we will deal with a specification of a *core model*, which is to be used for predicting and evaluating consequences of decisions. The value of a mathematical model as a decision aid comes from its ability to adequately represent reality. Therefore, there is always a trade-off between the requested accuracy (realism) of the model and the costs (also time) of its development and providing the model with data. Hence the requested accuracy should be consistent with accuracy really needed for the model. A specification and an implementation of a model require both knowledge and experience as well as collaboration of researchers with different backgrounds with users of a model. Actual model building is still a mixture of art and science that requires knowledge and experience, including a good understanding of the problem, good knowledge of model building methodology, and understanding of solution techniques that will be used for processing the model. Good overviews of related problems illustrated by many examples are provided by Huntley and James in [HuJ90] and by Williams in [Wil90]. The process of specifying the requirements to be met by the modeling process or establishing the specifications that the modeling process must fulfill is called metamodeling and one can also examine a metamodel (through the modeling process – cf [vG91]).

A core model is typically composed of (cf e.g. [WiM92]) the following elements:

- Decision variables that represent actual decisions (alternatives, choices, options, etc.). In RWQM the decision variables are selections of technologies (which includes also the so-called *do nothing option*) of waste water treatment plants located at each of the controllable waste emission points. Each technology at each water treatment plant has a corresponding binary variable that indicates, if a given technology is selected.

- Variables defining potential criteria (objectives, goals, performance indices, outcomes), which can be used for evaluating the consequences of implementing the computed or chosen decisions. In RWQM such objectives include various costs (total annual, investment, operational) and ambient water quality indicators (concentration of different waste constituencies, violations of water quality standards) both for selected monitoring points and for the whole region.

- Various intermediate and additional variables, such as balance and/or state variables, resources, endogenous (i.e. not controllable) decisions, which are necessary (or make it easier) to formulate the constraining relations and/or ease understanding of the model formulation and of interpretation of results. In RWQM such variables include resulting (after selected treatment options) concentrations of constituencies in the discharged water and in a river at the monitoring points, cost components for each treatment plant.

- Constraints (inequalities and equations) that reflect the logical relations between all variables represented in the model. In RWQM constraints include conditions for a sum of binary variables at each plant to be equal to 1 (thus making sure that exactly one technology is selected for each plant), mass balance equations for constituencies at each considered point, non-negativity constraints for all variables.

A solution of the model is composed of all defined variables of all types (decision, criteria, additional). A solution that fulfills the constraints is called a *feasible solution*. Therefore a set of constraints of a core model indirectly determines a set of *feasible decisions* and of *feasible values of criteria*.

The vector of variables x is composed of all types of variables used for the model specification, namely: decision, outcome, criteria, state, intermediate, parametric and additional variables. Quite often one variable can be classified as being of more than one type, for example a decision variable can be also treated as an outcome variable, an intermediate variable may be also an outcome, and typically all criteria are also outcome variables. The core model defines intermediate and outcome variables by additional equations. A classification of variables is an important issue for a model specification and it is usually convenient to use different symbols for different groups of variables. Readers interested in issues of a model specification are advised to consult [Wie92a] for more details. However, for the sake of brevity, we will use in this paper a simplified notation, which is aimed at a user of a model, who usually deals with only a small fraction of all variables used in a model, namely with the decision variables and with the variables representing objectives (criteria).

The core model, as defined above, is very similar to the definition of a *substantive model* proposed by Wierzbicki in [Wie92a]. The only difference is due to constraints for values of objectives, which are **not** included in a core model. This slight difference is caused by the assumption adopted in the implementation of LP-MULTI, which – in order to ensure consistency of the model analysis – does not allow any modification of the core model during the analysis.

Further on we will refer to a core model as a set of variables x and constraints that define a set X_0 of feasible solutions, i.e.

$$x \in X_0 \tag{1}$$

A properly defined core model has always a feasible solution, therefore X_0 is non-empty. Different procedures that help in analysis of the feasible solutions are discussed in the subsequent subsections.

Note, that the relation (1) is equivalent to one of the standard formulations of an LP problem, without specification of a goal function. Assume that $x \in \mathcal{R}^n$ is a vector of all variables, $A \in \mathcal{R}^{m \times n}$ is a matrix of constraining coefficients, $\underline{b} \in \mathcal{R}^m$ and $\bar{b} \in \mathcal{R}^m$ are vectors or right-hand sides, $\underline{x} \in \mathcal{R}^n$ and $\bar{x} \in \mathcal{R}^m$ are vectors of lower and upper bounds, respectively. The adopted convention assume that a corresponding i-th component of $\underline{b}$ and $\bar{b}$ are defined for four conventional types of LP constraints in the following way:
— for $=$ type: $\underline{b}_i = \bar{b}_i = rhs_i$,
— for $\leq$ type: $\underline{b}_i = -\infty, \bar{b}_i = rhs_i$,
— for $\geq$ type: $\underline{b}_i = rhs_i, \bar{b}_i = \infty$,
— for neutral constraints: $\underline{b}_i = -\infty, \bar{b}_i = \infty$,
where rhs_i is a right-hand side value of i-th constraint and ∞ is replaced by a sufficiently large number. Then a set of feasible solutions for an LP problem is defined by:

$$\underline{b} \leq Ax \leq \bar{b} \tag{2}$$

$$\underline{x} \leq x \leq \bar{x} \tag{3}$$

with the assumption that the sets of constraints (2) and bounds (3) include only logical and physical relations. Finally, by moving the constraints $x \leq \bar{x}$ to (2), by introducing so called *slack* variables to (2), and by shifting the variables by $\underline{x}$ one can obtain the most commonly known formulation of an LP problem in the form (without considering the goal function):

$$
\begin{aligned}
Ax &= b \\
x &\geq 0
\end{aligned}
\tag{4}
$$

3.3   Core model specification

Before discussing the guidelines we will briefly characterize the possibility of a presolve analysis of an LP model. It has been observed (cf. e.g. [Gon94, LMS94]) that a good presolver can substantially reduce the size of a problem and can also detect infeasibility of the problem. Therefore a presolve analysis is becoming a standard feature of LP solvers. If a presolve analysis is available as a part of a DSS for which a model is built, then some of the requirements for the model specification and generation can be softened. If a presolver is not available then a model generator should, in addition to the guidelines specified below, at least detect and suppress generation of redundant[1] constraints and variables.

We summarize here several guidelines based on the experience the author has had with applications in different areas. Those guidelines do not pertain to be complete but the experiences have shown that disregarding them results in either numerical problems or in an unnecessary complicated model generation, handling and analysis. We restrict the discussion to LP models but most of the guidelines should be also observed for non-linear models.

The following points should be considered during the specification and generation of a model that will be used for multiple criteria analysis aimed at supporting decision making:

1. The data used for the model should be stored, verified and handled separately from the model specification. The model should be generated either by a problem specific generator or by a general purpose modeling tool.
2. The model should include only substantial constraints. *Substantial* means the constraints representing all logical and physical relations between variables that should be taken into account while assessing the feasibility of a solution and physical relations between variables.
3. One should avoid "manual" scaling of the original data and of the LP matrix coefficients. The coefficients should be computed using original data. Any good LP solver scales the problem before attempting to solve it therefore one does not need to worry about generating very small or very large coefficients.
4. However, only essential matrix coefficients should be generated. This condition is important although it might be difficult to fulfill it, especially if a general purpose modeling tool is used. One should be aware that generation of non-essential small coefficients may make it impossible to scale well the matrix, which in turn usually results in numerical problems.
5. The generated bounds and right hand side values of constraints should correspond only to logical and physical relations. No additional restrictions nor constraints should be introduced in order to reflect acceptability of a solution because this will be accounted for during the model analysis. Therefore there is also no need to generate *soft* constraints.
6. All the potential criteria should be defined as outcome (or auxiliary) variables in the model.
7. The model specification should correspond only to the decision problem. For example, one should not generate additional *slack* variables in order to generate a problem in the standard LP form given by eq. (4). However, a specification of a model in a form suitable for a specialized solver (e.g. as a dynamic or stochastic problem) usually dramatically decreases the computation time.
8. One should avoid specifications of large numbers as *infinite* values. Such an approach

---

[1]This advice is justified by commonly known observations that as much as 1/3 of the constraints in some large LP models are redundant.

is sometimes used for removing the default finite bounds (specified in addition to the MPS format input files) and it is harmless for the simplex based solvers. However, it results in problems for the interior point algorithm implementations (cf [GoM95] for more details), therefore it should be avoided.

We conclude the guidelines with an additional comment on essential matrix coefficients. The problem can be illustrated by the following example. Consider the i-th row of the constraint matrix A from eq. (4) and assume that the matrix A is well scaled (i.e. the coefficients have absolute values close to one)[2] except of the i-th row. Any scaling routine can achieve good scaling of A, if all coefficients of i-th row are of the same (even very small) magnitude. However, if just one a_{ij} is several ranges of magnitude smaller than other coefficients in the i-th row, then there is no way to achieve a good scaling of A. This small a_{ij} value has a negligible impact on the value of the i-th row[3], but even one non-substantial coefficient usually causes substantial worsening of scaling of the matrix A, which in turn often results in numerical problems for a solver.

There are no easy to implement rules saying which coefficients are essential. This might be decided only by a modeler upon careful analysis of each group of constraints. One should be aware that a rule of rejecting coefficients having an absolute value smaller than a given threshold requires specification of such thresholds for each group of constraints.

3.4 Traditional model analysis for decision support

There are two groups of approaches to a model analysis:

Simulation: it is an alternative-focused method of the model analysis for which the decision variables are inputs and values of goals are outputs. Therefore simulation is oriented to examine the alternatives created by the user.

Optimization: it is a goal-oriented (value-focused) approach for which goals (objectives) are inputs and values of the decision variables are outputs. Hence, optimization helps to create alternatives.

A more detailed discussion on traditional simulation and single-criterion optimization is given in [Mak94a]. Therefore we restrain the discussion here to a few points relevant for the presentation of multiple-criteria decision analysis (MCDA).

Simulation is still an important tool for decision analysis and support. MCDA offers an easy way for implementation of an extension of this technique called *inverse simulation*, which is discussed in Section 6.2.

Traditional single-criterion optimization has one drawback important for decision support. Namely, almost all decisions are made upon analysis of several criteria. There have been a number of approaches to deal with multi-criteria problems within the framework of single-criterion optimization. The most popular approach is to select one criterion as a goal function and to impose constraints on other criteria. For example, Haimes proposed in [HaH74] ϵ *constraint approach*, which replaces (n-1) objectives into constraints with given *tolerable levels*. Such levels have an interpretation of aspirations for the criteria that have to be achieved. This hard requirement can be softened by representing requirements for the values of criteria as *soft* constraints. This approach is discussed in more

---

[2] A commonly accepted rule of thumb says that a matrix is well scaled, if the ratio of largest to smallest coefficients is smaller then 1000.

[3] This is why many modelers tend to underevaluate problems caused for solvers by generation of small (in absolute value sense) coefficients.

detail and a number of extensions of traditional single-objective optimization are summarized in [Mak94a]. The implementation of soft constraints in LP-MULTI is presented in Section 6.1.

There are two main common difficulties related to various extensions of single-criterion optimization. The first is practical: a sequential conversion of all but one criteria into constraints and changes of tolerable (desired) value of the corresponding constraints is a cumbersome procedure that is difficult to follow even by an experienced model builder. Therefore this is not a practical approach for actual decision support. The second reservation is due to the practical implementation of the sensitivity analysis, which is often a main tool for model analysis with the help of single-criterion optimization.

A sensitivity analysis that uses a dual solution of an optimization problem is recommended by many text books on applications of mathematical programming but the limitations and limited reliability of this approach are not widely recognized. However, we restrict the comments to the following two main points:

- The main limitation is due to the fact that the dual solution has a well-defined interpretation only in the neighborhood of the optimal solution. This neighborhood is not directly available from the standard output of a solver and commonly known observations show that users tend to extend the interpretation of dual solution (shadow prices for LP problems) far beyond the region in which it is valid (although a postoptimal analysis can easily provide the range for which it is valid).

- The limited reliability is due to the availability of a unique dual solution and its robustness. This problem is far beyond the scope of this paper, but the author would like to advise everyone, who uses dual solutions to read at least these two papers: Jansen et al. provide in [JdJRT93]) a good summary of the related problems and of their experiences with application; Güler et al. present in [GdHR⁺93]) a survey of degeneracy in the Interior Point methods, which by many users are considered to be free of degeneracy problems.

4 Aspiration-based decision analysis and support

The shortcomings of a single-criterion optimization as a tool for decision making support have been a main driving force for development and applications of multicriteria optimization that can better support a decision making process. The term *Multiple Criteria Decision Analysis* (MCDA) covers a wide area of methods and applications. In the following subsections we will discuss in more detail various approaches to the MCDA. Section 4.1 contains a summary of the concepts used in the multicriteria optimization. Several approaches to analysis of Pareto-efficient solutions are briefly characterized in Section 4.2. One of the most popular approaches, namely *aspiration-led multiple criteria optimization*, is discussed in detail in Section 4.3. Its extension, known as *aspiration-reservation based decision support* is summarized in Section 4.4.

The MCDA does not aim at providing *"the best"* solution but it helps in analysis of the problem. MCMA is a very useful component of DSSs and therefore it might be widely applied in different areas of applications. LP-MULTI is a modular software tool aimed at making implementation of MCMA in DSSs more easy, at least for linear programming type models. However, a proper use of LP-MULTI requires a good understanding of the underlying methodology. Therefore the related methodological issues are discussed in more detail in this paper.

4.1 Pareto efficient solutions

The key problem in any decision making is a selection of one solution $\hat{x}$, out of many feasible solutions $x \in X_0$ that are defined by the corresponding core model (cf Section 3.2). In a typical situation it is impossible to introduce an ordering among all solutions x, therefore solutions x are evaluated using a vector of selected criteria $q(x)$, where $q \in R^n$, n is a number of criteria. Criteria have usually obvious interpretations, such as costs, investments, waste concentration, income, etc. However, typically there is no way[4] to aggregate all criteria into one objective that can adequately represent a preference structure of a DM.

There are several variants for defining basic concepts of multiple criteria optimization. We recall here one of the simplest set of definitions. A reader interested in more detailed and rigid definitions may consult e.g. [SNT85, Ste86, Yu85, Wie92b]. The following definitions will be used in the subsequent discussions. Note that, in order to simplify both the discussion and the implementation, we assume that the criteria q are selected among the variables x defined in the core model and that all criteria are minimized[5].

Weakly Pareto-optimal solution: A solution $\hat{x} \in X_0$ is called a weakly Pareto-optimal solution, if there exists no other feasible solution that has better values of all criteria. Weakly Pareto-optimal solutions are usually easier to be computed. Therefore a proper method (see the explanation of eq. (9)) should be implemented to avoid computing and reporting a weakly Pareto-optimal solution as an efficient solution. This is a purely technical problem and weakly Pareto-optimal solutions have no practical meaning for a user of a properly implemented DSS.

Pareto-optimal solution: A solution $\hat{x} \in X_0$ is called a Pareto-optimal solution, if there is no other feasible solution for which one can improve the value of any criterion without worsening the value of at least one other criterion. A Pareto-optimal solution is also called an *efficient* solution (some authors call it also non-dominated solution) and it can be defined (for a minimized criterion q_i) as:

$$\neg \exists x \in X_0 \neq \hat{x} : \quad \{ q_i(x) \leq q_i(\hat{x}) \ \forall i \in [1,...,n] \ \text{and}$$
$$\exists k \in [1,...,n] : q_k(x) < q_k(\hat{x}) \ \} \tag{5}$$

Most practical in applications are properly Pareto-optimal solutions with a prior bound on trade-off coefficients (see [Wie86] for more details). Further on, a properly Pareto-optimal solution will be simply called Pareto solution.

Pareto-optimal point: Pareto-optimal point is composed of values of all criteria for a corresponding Pareto-optimal solution.

Pareto set: Pareto-optimal set (sometimes called also Pareto frontier) is composed of all Pareto-optimal points.

Utopia point: Utopia point q^U is composed of best values out of the set of all Pareto-solution for each criterion. A utopia point (often called also an ideal point) can be easily computed as a result of n single criterion optimization with each criterion at a time serving as an objective function.

---

[4]Multiattribute utility function approach assumes it is possible to construct a function that maps elements of the criteria set q into R^1 in such a way, that a larger number corresponds to the stronger preference. See e.g. [Mak94a] for the discussion (and references) about limitations of this approach.

[5]See Section 5.2 for the treatment of maximized and stabilized criteria.

Nadir point: Nadir point q^N is composed of worst values out of the set of all Pareto-solution for each criterion. Finding a nadir point is typically difficult for problems that have more than two criteria (cf e.g. [IsS87] and an example in Section 5.1).

Aspiration point: Aspiration point (sometimes called a reference point) is composed of the desired values specified by a user for each criterion. In other words, the values that a user would like to achieve for each objective. The aspiration point will be defined in this paper by $\bar{q} \in R^n$.

Reservation point: Reservation point is composed of the values still acceptable by a user for each criterion. The reservation point will be defined in this paper by $\underline{q} \in R^n$. Therefore, the pairs of aspiration and reservation levels define, for a corresponding criterion, a range of values between the desired and still acceptable levels.

Utopia and nadir (or a good approximation of a nadir) provide valuable information about ranges of values (for all efficient solutions) of each criterion. Therefore those points outline for each criterion a range for reasonable values of aspiration and reservation levels.
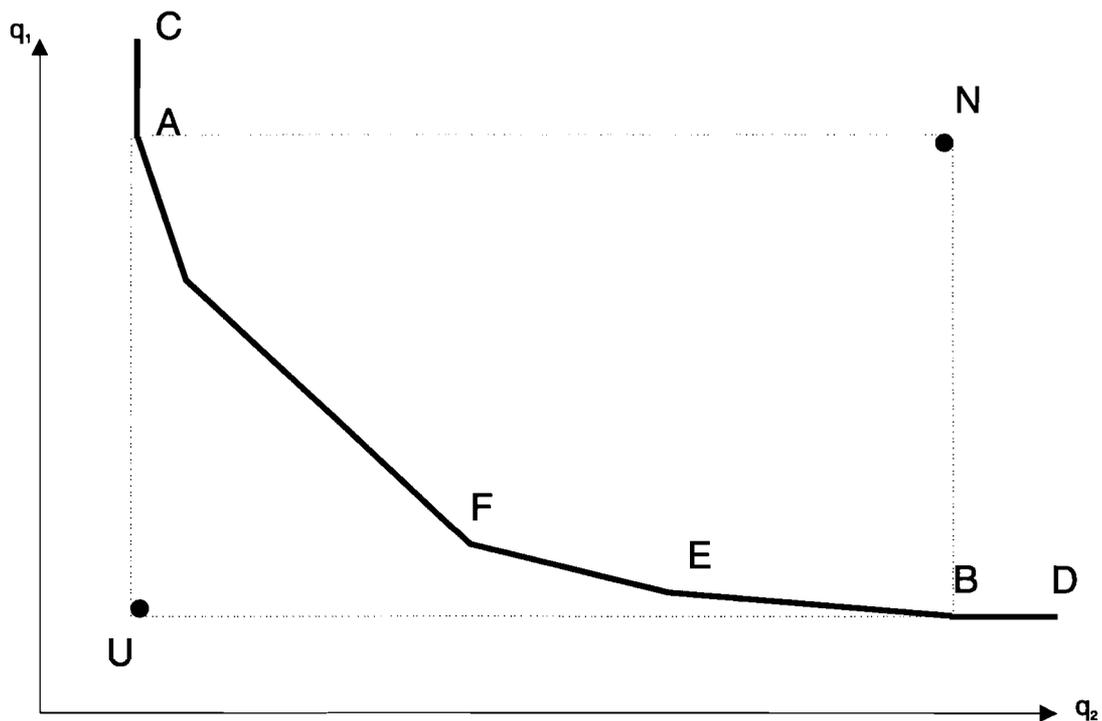

Figure 1: An illustration of basic concepts used in MCDA.

The above definitions are illustrated, for a problem with two minimized criteria (q_1 and q_2), in Figure 1. The Pareto set is contained between points **A** and **B**. Weakly Pareto points are located on the segments **AC** and **BD**, and non-properly optimal Pareto points are in the segment **BE**. Note, that the slope of segment **BE** corresponds to the trade-off coefficients (see the explanation of eq. (9) in Section 4.3) and is usually very small. If the bound on the trade-off coefficients will be increased, then the set of properly Pareto-optimal solutions will be reduced to the two segments between points **A** and **F**. The utopia and nadir points are marked by **U** and **N**, respectively.

4.2   Analysis of efficient solutions

Obviously, any rational solution should be a Pareto optimal one. The following multi-objective programming problem provides a way for computing Pareto optimal solutions:

$$\min_{x \in X_0} q(x) \tag{6}$$

where $q \in R^n$, n is a number of criteria. Although a set of Pareto solutions is a very small subset of all feasible solutions, in practical applications there is typically an infinite[6] number of efficient solutions. Hence, one needs a procedure (and corresponding tools) for generation and examination of those Pareto solutions that correspond well to the preferences of a DM and to the way in which a DM wants to express his/her preferences. The correspondence of such procedure to the needs of a DM is the key issue of MCDA support.

Before discussing in detail (in Section 5) the procedure implemented in LP-MULTI, we outline below several other approaches. All the approaches discussed assume different definitions of a *scalarizing achievement function*[7], which allows for generation of a single objective auxiliary optimization problem whose solution is also one of the solutions of the problem (6).

Theoretically, it is possible to use a *multiattribute value function* as the scalarizing function. However, there are many fundamental and technical difficulties related to the identification of the value function that adequately reflects the preferences of a DM (cf e.g. [Mak94a] for arguments and a list of references).

The oldest and simplest approach that is still quite popular assumes the scalarizing function in the form of weighted sum of criteria:

$$s(\alpha, q) = \sum_{i=1}^{n} \alpha_i q_i \tag{7}$$

where the weighting coefficients α_i have to be defined, usually indirectly, by a DM. This approach has a number of drawbacks that are discussed in more details by Wierzbicki and Makowski in [WiM92]. Here we only summarize the two main arguments. First, the scalarizing function (7) does not allow us to find all Pareto solutions. Consider the simplest case with two minimized objectives illustrated in Figure 2. For the linear case, a user can obtain only Pareto-optimal solutions corresponding to vertices **A**, **B** and **C**. For any weighting coefficients vector α with a slope smaller than the slope of the vector α^1, a solution will be in the vertex **A**. For a weighting coefficient vector that is parallel to α^1, there is no unique solution[8], and a very small increase of the slope of α will cause the solution to jump to the vertex **B**. Further increase of slope of α will not cause any changes in the Pareto solution until the slope will be greater than α^2 (which will cause another jump to the vertex **C**). This explains the experience known to everyone, who tried to use weights for analysis of multiple-criteria LP models. Namely, often a relatively large change of weights does not result in any changes of the solution but, in another

---

[6]In some problems (e.g. the RWQM problem outlined in Section 2) the number of solutions is finite but usually very large, therefore analysis of all solutions is practically impossible.

[7]Many of the discussed approaches do not use, in the corresponding original formulation, the achievement function concept. However, it is easy to formulate such functions for each approach in order to provide a consistent comparison.

[8]Therefore the corresponding problem will be degenerated and any solution from the edge **AB** is optimal. Hence, the reported solution will differ, depending not only on the applied solver but also on the parameters used for a solver, including the possibly defined starting point.
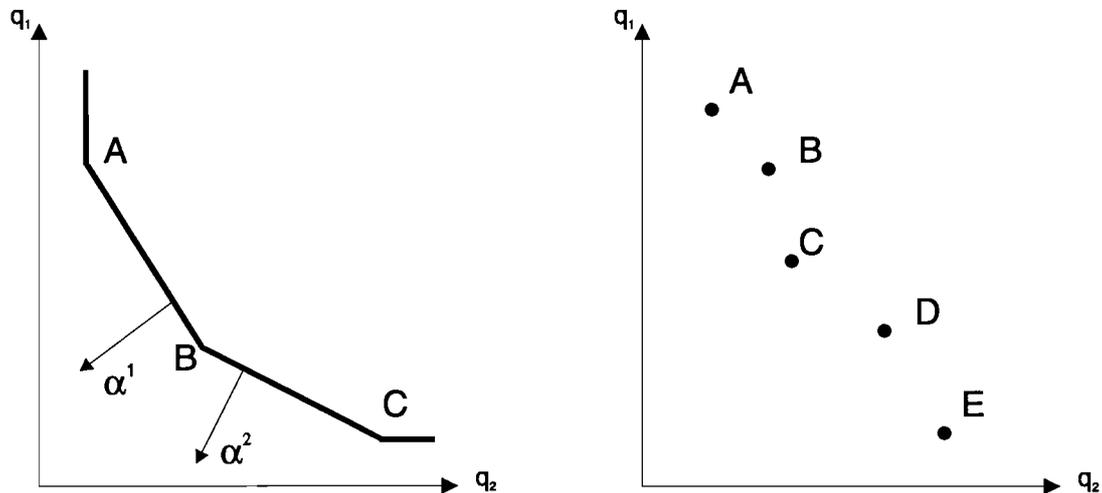
Figure 2: Limitations of selecting all Pareto solutions by scalarizing function (7): the cases of linear and discrete models.

region of Pareto set, a small modification of weights generates (for the same model) a substantially different solution. For a discrete model a surface spanned over the Pareto set (that is composed of points) may be non-convex. Therefore a number of efficient solutions will not be available (for the example depicted in Fig. 2 efficient solutions **B**, **D**), if the scalarizing function (7) is applied. Nakayama provides in [Nak94] not only similar arguments but also an example that shows that there might be no positive correlation between increasing a weight for a criterion and the corresponding improvement of the criterion value. Hence, using weights might also be counter-intuitive[9] and therefore it is hardly possible to implement the scalarizing function (7) in a way that meets expectations of a DM.

The limitations of the two approaches summarized above have led to developments of various methods based on a most natural way of expressing preferences. Namely, by a specification of the aspiration levels.

Goal Programming (GP), originally proposed by Charnes and Cooper in [ChC67], is a commonly known technique that assumes minimization of a distance between a point composed of criteria's value and a given aspiration point. The GP uses a scalarizing function:

$$s(q, \bar{q}) = \| q - \bar{q} \| \qquad (8)$$

Minimization of (8) with respect to $x \in X_0$, provides a solution having values of criteria that are in some sense closest to the goals specified as the aspiration level $\bar{q}$. This technique can be refined in various ways – by an appropriate selection of the norm $\| \cdot \|$ defining the distance that can use weighting coefficients as additional controlling parameters – but there are two disadvantages related to using the GP method:

- Minimization of (8) provides a Pareto solution, if $\bar{q}$ is not attainable. However, if $\bar{q}$ represents attainable goals then there is no way to find a Pareto solution by minimization of (8).

- Selection of the norm $\| \cdot \|$ requires definition of weighting coefficients. In the original formulation the weights are assumed to be equal to one. However, such an approach is

---

[9]The role of intuition in decision making is discussed in more detail in [Wie92c].

acceptable, if the problem is well scaled, which, in practice, is rarely the case. Therefore additional assumptions have to be made in order to define weights for the norm $\| \cdot \|$.

Wierzbicki proposed in [Wie77] an effective way for overcoming those disadvantages. The method uses, instead of the norm, a scalarizing function that remains monotone even if goals are attainable. Additionally, there are natural ways of defining weights – that have interpretation of scaling (trade-off) coefficients – used in a scalarizing function. Later this method has been elaborated and is known as the aspiration level method (also called the reference point method, cf [Wie80, Wie82]). We will use for *Aspiration Led Decision Support* methods the abbreviation ALDS. Several other extensions or similar approaches have been proposed and implemented (cf [LeW89, KoL84, LAP94, Nak94, SNT85, Sak93, SeS88, Ste86]).

The ALDS approaches discussed in Sections 4.3 can be considered as an extension of GP. The detailed comparison of the two methods is provided by Ogryczak and Lahoda in [OgL92]. All functionality of the GP can be provided by the aspiration-led method and computational complexities of both methods are comparable. Therefore the reference point approach seems to be a good replacement for the GP.

4.3 Aspiration-led decision support

The essence of the ALDS method can be summarized as follows:
1. The DM selects, out of the potential objectives, a number of variables that will serve as criteria for evaluations of feasible solutions $x \in X_0$ defined by a core model. In typical applications there are 2–7 criteria.
2. The DM specifies (with a help of an interactive tool) an aspiration level $\bar{q} = \{\bar{q}_1, \ldots, \bar{q}_n\}$.
3. The problem is transformed by a DSS into an auxiliary parametric single-objective problem. Its solution gives a Pareto-optimal point. If a specified aspiration level $\bar{q}$ is not attainable, then the Pareto-optimal point is the nearest (in the sense of a Chebyshev weighted norm) to the aspiration level. If the aspiration level is attainable, then the Pareto-optimal point is uniformly better than $\bar{q}$.
4. The DM explores various Pareto-optimal points by changing the aspiration levels $\bar{q}$. The underlying (done by a DSS) formulation of the problem is minimization of an *achievement scalarizing function* that can be interpreted as an ad-hoc non-stationary approximation of the DM's value function depending on the currently selected aspiration level.
5. The procedures described in points 2, 3 and 4 are repeated until a satisfactory solution is found.

Selection of the Pareto-optimal point depends on the definition of the achievement scalarizing function, which includes also a selected aspiration point.

Most of the ALDS methods use the scalarizing function in the form:

$$s(q, \bar{q}, w) = \max_{1 \leq i \leq n} \{w_i(q_i - \bar{q}_i)\} + \epsilon \sum_{i=1}^{n} w_i(q_i - \bar{q}_i) \qquad (9)$$

where $q(x) \in R^n$ is a vector of criteria, $\bar{q} \in R^n$ is an aspiration point, $w_i > 0$ are scaling coefficients and ϵ is a given small positive number. Minimization of (9) for $x \in X_0$ generates a properly efficient solution with trade-off coefficients less then $(1 + 1/\epsilon)$. Setting a value of ϵ is itself a trade-off between getting a too restricted set of properly Pareto solutions or a too wide set practically equivalent to weakly Pareto optimal solutions. Too small a value of ϵ results in properly optimal solutions that are, however, practically not distinguishable from weakly Pareto optimal solutions whereas too large value of ϵ

results in properly Pareto solutions with too strongly limited trade-offs (see Figure 1 for illustration). Assuming the ϵ parameter to be of a technical nature, the selection of efficient solutions is controlled by the two vector parameters: $\bar{q}$ and w.

In practice, the main controlling parameter is the aspiration point $\bar{q}$, which in all interactive methods is under the control of a user. Many implementations made for different types of problems have shown that specifications of $\bar{q}$ fit very well into a natural way of analysis of decision problems. Users of aspiration-led DSS quickly learn the range of the criteria's values that are worth examining more closely.

In case of many criteria, Nakayama recommends ([Nak94]) a procedure for an automatic trade-off between criteria, which eases the problem of the specification of aspiration levels for a large number of criteria. This procedure is useful for problems where it is not practicable to specify aspiration levels for each criterion. The procedure is based on the sensitivity analysis, for which one should consider the reservations discussed in Section 3.4. However, the problems caused by the degeneracy can easily be corrected in the case of the automatic trade-off, namely by computing a new Pareto solution using the automatic trade-off point as the aspiration point. Therefore the automatic trade-off is a robust approach also for degenerated problems.

There is a common agreement that the aspiration point is a very good controlling parameter for examining a Pareto set. Much less attention is given to the problem of defining the weighting[10] vector w. From the purely methodological point of view a selection of w might be considered not to be important, if we consider it only as a tool for examination of the whole Pareto set[11]. This might explain why some implementations (e.g. [Sak93]) assume $w_i = 1$.

In order to illustrate the role of w it might be useful to recall one of the first methods in multi-objective optimization suggested by Benayoun et al. in [BdMTL71]. This method can be interpreted as minimization of (9) with $\epsilon = 0$, and $\bar{q}$ set to the utopia point q^U. An extension of this approach is presented in more detail e.g. by Steuer in [Ste86]. Clearly, one can examine the whole Pareto set with the aspiration point fixed at the utopia point[12] by changing only w. With a weighting vector w such that

$$w \in W = \{w \in R^n | w_i > 0, \sum_{i=1}^{n} w_i = 1\} \tag{10}$$

we can generate a family of weighted Chebyshev norms for measuring the distance between the utopia point q^U and a Pareto solution q as

$$\| q - q^U \|_{\infty}^{w} = \max_{1 \leq i \leq n} \{w_i(q_i - q_i^U)\} \tag{11}$$

One can interpret w as scaling or trade-off coefficients. Since the vector w determines the selection of a norm from a family of norms (11) it therefore determines also which Pareto solution is considered to be closest to an aspiration level. Consider the example

---

[10]Note that the weights w should not be confused with the weights α required for the scalarizing function (7).

[11]Because a whole Pareto set can be examined for any $w > 0$ by changing only $\bar{q}$. Note also, that all reference points located on a ray having a direction defined by w will generate the same Pareto solution (which, for a continuous problem, is given by the intersection of this ray which the Pareto surface).

[12]More exactly, the aspiration point should be set to a slightly shifted utopia point, if one applies the definition of w in the form of (10). Replacing in this definition the condition $w_i > 0$ by $w_i \geq 0$ (as is done e.g. in [SNT85, Ste86]) makes it possible to avoid shifting the utopia point. However, in such a case eq. (11) does not define a norm (e.g. using it for measuring a distance would not allow us to distinguish different weakly Pareto-optimal solutions).
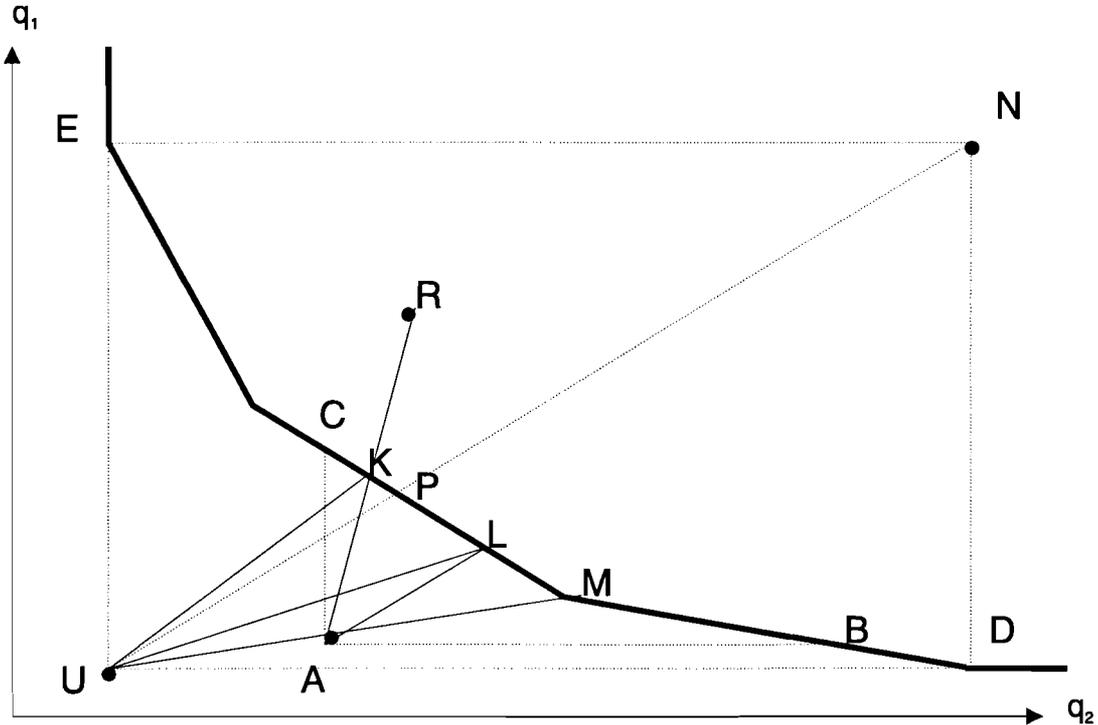
Figure 3: Interpretation of weights in a component achievement function.

in Figure 3. The slopes of half-rays UK, UL, UM are determined by the different ratios of w_1/w_2, which can be interpreted as different substitution rates between the marginal loss in criterion q_1 and the gain in q_2. For the norm defined by (11) we can obtain (by changing the ratio w_1/w_2) any Pareto point, i.e. any point on the segments between points **D** and **E**. By moving the aspiration point from the utopia point **U** (composed of q_i^U) to the point **A** (composed of $\bar{q}_i$) we limit the choice of Pareto-points to the points located between **B** and **C**. Note, that the slopes of rays from the aspiration point **A** are different than slopes of rays from the utopia point **U** to the respective point. This is consistent with the different substitution rates that result in selection of the same Pareto solutions that are closest to another reference point (aspiration point **A** instead of utopia point **U**).

A more detailed discussion on weights in a scalarizing function is beyond the scope of this paper. A reader interested in the related problems may want to consult the paper by Lootsma et al. [LAP94], who provide theoretical background and report on experiences with using weights. In this paper use of the scalarizing function (9) is compared with a scalarizing function composed of a weighted geometric mean of criteria.

We summarize the issue of weights in ALDS by listing four commonly used approaches:

- A set of weighting vectors w is generated randomly (possibly under a condition that each w has to be contained in a given cone, which is contracted in each iteration). Then this set of w is filtered and a smaller number of vectors w is used for computing corresponding Pareto solutions (cf Steuer in [Ste86]).

- Weights are calculated without using information related to a preference structure of a user. Typically, weights have in such a case mainly a scaling function and are calculated using utopia and nadir points (cf [LeW89, Nak94]). Similarly, the scaling function of weights can be implemented by setting $w_i = 1/|\bar{q}_i|$ as suggested e.g. by Korhonen in [KoL84].

- Weights are computed using preferential information, specified by a user at each iteration in a form of trade-offs between current criteria's values. An example of such an approach that uses pair-wise comparison of criteria is provided by Lootsma et al. [LAP94].
- Weights are computed using a currently specified aspiration point $\bar{q}$. In this approach a utopia point q^U is usually taken (cf [SNT85, Nak94] as the second point needed for calculating the direction. In the implementations of DSS of the DIDAS family (cf [LeW89]) a current reservation point q^R is used instead of q^U.

The Pareto points K, L, M in Figure 3 correspond to different ways of defining weights in the component scalarizing functions. Namely, for the aspiration point **A**, weights defined by utopia and nadir points would result in the point L, by utopia and aspiration points in the point M, and by the aspiration and reservation (marked by **R**) levels in the point K. This example illustrates a typical situation in which weights defined by aspiration and reservation levels provide a solution with criteria's values between the corresponding aspiration and reservation levels. Other weighting methods often provide a solution for which values of some criteria are worse than a reservation level.

Note, that the Pareto-optimal point marked by **P** in Figure 3 corresponds to the so-called compromise solution, i.e. a solution obtained for aspiration and reservation points set to the utopia and nadir points, respectively. The compromise solution is usually a starting point for the interactive analysis of a model (see Section 6.4).

As the final argument for the ALDS methodology we would like to refer to the results of an experimental investigation reported by Korhonen and Wallenius in [KoW89]. The authors compared five different interactive procedures for multiple-criteria based support for decision making. The following criteria were used for evaluation of the techniques:

- Satisfaction with the solution obtained.
- Confidence in the technique.
- Ease of understanding the technique.
- Ease of using the technique.
- Correspondence between the subject's responses and the implied search directions.
- Information provided by the technique.
- Experienced speed of convergence.

The ALDS technique was found clearly superior (the preference ranking of the techniques was identical for each of the four measures of performance). The paper contains also an interesting summary of observations regarding choice behavior related to different techniques of multiple-criteria decision support.

4.4   Aspiration-reservation based decision support

Following [OgL92] we will use for Aspiration-Reservation Based Decision Support techniques the acronym ARBDS. The ARBDS is an extension of the ALDS approach summarized in Section 4.3 and is based on the methodology proposed by Wierzbicki (cf e.g. [Wie86, Wie92c]), who formulated also general properties for the achievement scalarizing function. The commonly used (e.g. in several implementations of the DIDAS family reported in [LeW89]) form of the achievement scalarizing function is the following:

$$S(q, \bar{q}, \underline{q}) = \min_{1 \le i \le n} u_i(q_i, \bar{q}_i, \underline{q}_i) + \epsilon \sum_{i=1}^{n} u_i(q_i, \bar{q}_i, \underline{q}_i) \tag{12}$$

Maximization of the function (12) provides a properly Pareto-optimal solution with the trade-off coefficient smaller than $(1 + 1/\epsilon)$.

Component achievement functions $u_i(\cdot)$ are strictly monotone (decreasing for minimized and increasing for maximized criteria, respectively) functions of the objective vector component q_i with values

$$u_i(q_i^U, \cdot) = 1 + \bar{\beta}, \qquad u_i(\bar{q}_i, \cdot) = 1, \qquad u_i(\underline{q}_i, \cdot) = 0, \qquad u_i(q_i^R, \cdot) = -\bar{\eta} \qquad (13)$$

where $\bar{\beta}$ and $\bar{\eta}$, are given positive constants, typically equal to 0.1 and 10, respectively.
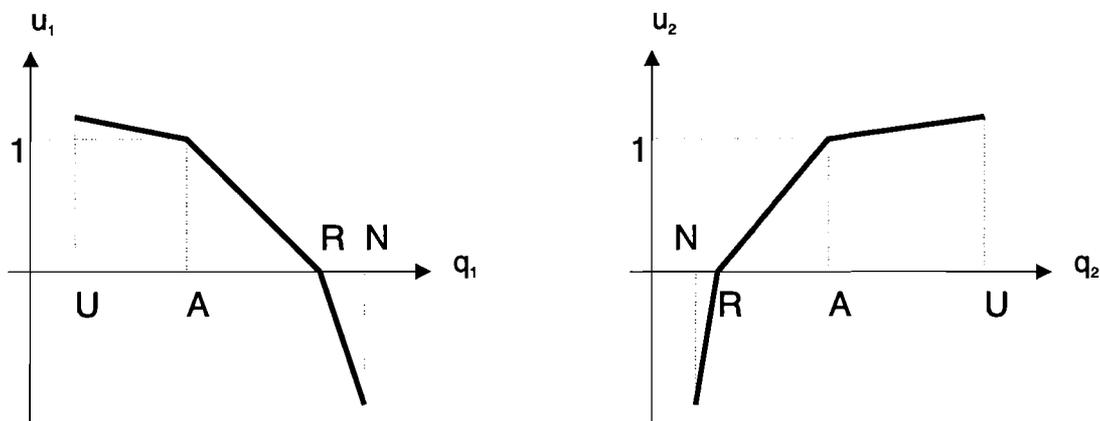

Figure 4: Piece-wise linear component achievement functions used in the achievement scalarizing function (12), q_1 and q_2 are minimized and maximized criteria, respectively. **U, A, R, N** are the utopia, aspiration, reservation and nadir values of the criteria q_1 and q_2, respectively.

The piece–wise linear component achievement functions u_i proposed by Wierzbicki in [Wie86] and illustrated in Figure 4 are defined by (14) and by (15) for minimized and maximized criteria, respectively.

$$u_i(q, \bar{q}, \underline{q}) = \begin{cases} \alpha_i w_i(\bar{q}_i - q_i) + 1, & \text{if} \quad q_i < \bar{q}_i \\ w_i(\bar{q}_i - q_i) + 1, & \text{if} \quad \bar{q}_i \le q_i \le \underline{q}_i \\ \beta_i w_i(\underline{q}_i - q_i) & \text{if} \quad \underline{q}_i < q_i \end{cases} \qquad (14)$$

$$u_i(q, \bar{q}, \underline{q}) = \begin{cases} \alpha_i w_i(\underline{q}_i - q_i) & \text{if} \quad q_i < \underline{q}_i \\ w_i(\bar{q}_i - q_i) + 1, & \text{if} \quad \underline{q}_i \le q_i \le \bar{q}_i \\ \beta w_i(\bar{q}_i - q_i) + 1, & \text{if} \quad \bar{q}_i < q_i \end{cases} \qquad (15)$$

where $w_i = 1/(\underline{q}_i - \bar{q}_i)$, and α_i, β_i $(i = 1, 2, \ldots, n)$ are given parameters. The parameters α_i and β_i are set in such a way that u_i takes the values defined by (13).

The ARBDS method outlined above can be also interpreted in terms of fuzzy sets as an extension of *interactive fuzzy multi-objective programming* as proposed by Seo and Sakawa in [SeS88, Sak93]. In this approach the membership function is not elicited at an initial iteration but a user is allowed to interactively change it upon analysis of obtained solutions. This approach assumes the classical form of the membership function originally proposed by Zadeh in [Zad65]. However, in order to properly handle – within the framework of the component achievement function – criteria's values worse than a reservation level, and better than an aspiration level, it is necessary to admit values of a membership function that are negative or greater than one. Such an extension of the membership function has been proposed by Granat and Wierzbicki in [GrW94]. Note

that then the component achievement functions u_i defined by (14,15) and illustrated in Figure 4 can be interpreted in a natural way as the extended-valued membership functions $\mu(q)$. A method of constructing order-consistent component achievement scalarizing functions based on membership functions describing the satisfaction of the user with the attainment of separate objectives is discussed in more detail by Granat and Wierzbicki in [GrW94]. Therefore such an interpretation of the extended-valued membership function makes it possible to combine the extensions of the two methods: *Aspiration Reservation Based Multiple-Criteria Optimization* and *Fuzzy Multi-objective Linear Programming* into a uniform approach described in Section 5.1.

5 LP-MULTI: modular tool for MCDA

LP-MULTI has been designed and implemented as a modular tool that makes it easier to apply multiple-criteria model analysis to problem specific DSS. The following subsections provide a summary of methodology and implementation of the prototype of LP-MULTI.

5.1 Methodology applied in LP-MULTI

The approach implemented in LP-MULTI is based on the ARBDS approach summarized in Section 4.4. The basic difference is due to the definition of the component achievement function, which replaces the functions defined by (14,15). Another, more technical modification is applied to the achievement scalarizing function (12). We briefly summarize first that latter modification and will discuss afterwards the justification for another form of the component achievement function in more details.

During the analysis of the problem it is often useful to temporarily disregard some of the criteria. A criterion for which a user does not wish to define the corresponding component scalarizing function is called in LP-MULTI *an inactive criterion*. Inactive criteria are also handy for computing a good approximation of a nadir point. However, a complete disregarding of a criterion from the achievement scalarizing function may result in both numerical problems (caused by a degenerated problem) and in a random value of the criterion (which may be unnecessarily bad[13]). Therefore, the following form of the achievement scalarizing function is implemented in LP-MULTI in order to facilitate a proper handling of inactive criteria:

$$S(q, \bar{q}, \underline{q}) = \min_{1 \leq i \leq n} \gamma_i u_i(q_i, \bar{q}_i, \underline{q}_i) + \epsilon \sum_{i=1}^{n} (\gamma_i u_i(q_i, \bar{q}_i, \underline{q}_i) + (1. - \gamma_i) s_i q_i) \qquad (16)$$

where γ_i is equal to 1 or 0, for active and non-active criteria, respectively, and the scaling coefficients s_i are defined by:

$$s_i = \frac{\text{sign}(q_i^U - q_i^N)}{\max(1, |q_i^U - q_i^N|)} \qquad (17)$$

where sign(x) is a function that returns 1 for non-negative numbers and –1 otherwise.

A justification for another form of the component achievement function requires a more detailed discussion: The Fuzzy Multi-objective Linear Programming often uses a piece-wise linear membership function composed of several segments (cf e.g. [Sak93]). In particular, such a function can be used as an approximation of a non-linear membership

---

[13]Which in turn would result in a bad approximation of a nadir point, cf Section 6.

function. This observation motivated us to extend the commonly used ARBDS method by allowing in LP-MULTI a specification of additional points between aspiration and reservation values. Users, who feel themselves comfortable with the fuzzy multi-objective programming method specify those points by specification of a piece-wise linear membership function. However, experience has shown that such points can be also easily specified without an explicit interpretation in terms of fuzzy sets. We have observed that specification of such additional points is especially useful for problems with more than three criteria, when typically a user selects 2-3 criteria to be of primary[14] importance. In such situations a user is often not certain about crisp values for aspiration and/or reservation levels for criteria that are considered of secondary importance and therefore specification of ranges (instead of precise values) is much easier. The ranges can be represented in a natural way by additional segments of piece-wise linear function u_i.
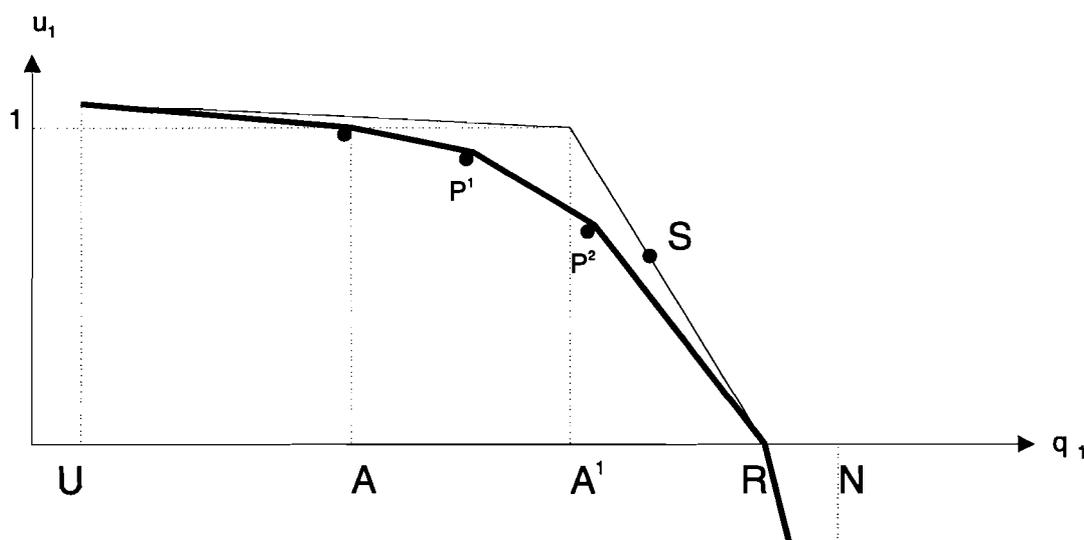

Figure 5: Interactive specification of a piece-wise linear component achievement function for a minimized criterion. Last solution **S** lies on the old function (marked by the thin line) defined by the aspiration level $\mathbf{A}^1$ and the reservation value **R**. A user selected (by clicking a mouse) a new aspiration level **A** and two new additional points ($\mathbf{P}^1$, $\mathbf{P}^2$) between aspiration and reservation points. The segments between the utopia point **U** and the aspiration point, and between the reservation point and the nadir point **N**, are generated by FT-TOOL.

Therefore the functions u_i defined by (14,15) are replaced in LP-MULTI by a piece-wise linear, strictly monotone functions that may have more than three segments. Such functions are specified indirectly by the user with the help of an interactive tool (see Figure 5 for an illustration). The interaction for the procedure outlined below has been implemented in the FT-TOOL (cf [GrM95] for the description). Upon analysis of a previously computed Pareto solution, the user specifies new aspiration and reservation levels and, optionally additional points. More precisely, the functions u_i are defined as the result of the following procedure:

---

[14]Subconscious classification of criteria into two groups changes during the interaction. Typically a user changes substantially aspiration and/or reservation levels for one, two or three criteria until he/she finds a solution satisfactory with regard to the selected criteria. Once this is achieved other criteria are considered more closely.

- A user specifies with the help of an interactive tool $\bar{q}_i$ and $\underline{q}_i$ (aspiration and reservation values) for each criterion. Optionally, the user may specify additional points between $\bar{q}_i$ and $\underline{q}_i$.
- The values of $u_i(\bar{q})$ and $u_i(\underline{q})$ are set by FT-TOOL to 1.0 and 0.0, respectively. Values of u_i for the optionally specified points are read directly from the screen. Therefore the user specifies, for i-th criterion, a set of pairs $\{q_{ji}, u(q_{ji})\}, j = 1, ..., p_i$, where $p_i \geq 2$ is a number of points specified for i-th criterion.
- The points specified by a user, together with the utopia point q^U and approximation of nadir point q^N (which are computed before an interactive analysis of a problem starts) are used for the definition of a piece-wise linear functions u_i.

For the example illustrated in Figure 5, a new Pareto-optimal solution will be located on the new function u_1, and (if the component achievement functions for other criteria have not been changed) will have a better (smaller) value for the criterion corresponding to the function u_1.

Hence, the piece-wise linear function u_i is defined in LP-MULTI by segments u_{ji}:

$$u_{ji} = \alpha_{ji} q_i + \beta_{ji}, \qquad q_{ji} \leq q_i \leq q_{j+1,i} \qquad j = 1, \ldots, p_i \qquad (18)$$

where p_i is a number of segments for i-th criterion,

$$\alpha_{ji} = \frac{u_{j+1,i} - u_{ji}}{q_{j+1,i} - q_{ji}} \qquad (19)$$

$$\beta_{ji} = u_{ji} - \alpha_{ji} q_{ji} \qquad (20)$$

Concavity of the piece-wise linear function $u(q)$ defined by (18) can be assured by a condition:

$$\alpha_{1i} \geq \alpha_{2i} \geq \ldots \geq \alpha_{p_i i} \qquad (21)$$

This assumption corresponds well to the nature of the problem since one accepts small changes of u_i when a criterion value is better or close to an aspiration level and the speed of such change should increase along with moving towards a reservation level and should increase even faster between reservation and nadir points. Such features are consistent with the commonly known properties of the membership function used in applications based on the fuzzy set approach. It is obviously rational to deal with a strictly concave function by dropping out points j for which $\alpha_{j-1} = \alpha_j$. Therefore we can assume that:

$$\alpha_{1i} > \alpha_{2i} > \ldots > \alpha_{p_i i} \qquad (22)$$

Note, that α_{ji} are weighting (scaling, trade-off) coefficients discussed in more details in Section 4.2. However, the implementation in LP-MULTI allows for weights that are different also for the criteria's values between aspiration and reservation levels, thus giving an additional tool for better reflecting the preferences of a DM by the corresponding component achievement function. We provide justification for the choice made for the implementation of the calculation of weights in LP-MULTI, without trying to argue that this is always the best choice. This choice is based on the experiences we have with different applications. The main arguments are as follows:

- LP-MULTI generates (in the interactive phase) only one Pareto solution for each selection of $\bar{q}$.
- Our experiences show that it is better to use directly only one control for the selection of a Pareto solution. Using directly both controls (aspiration point and weights) often creates confusion even for a developer of a DSS. Therefore such an approach would

hardly be acceptable for a DM. Controlling through selection of aspiration/reservation levels is much more intuitive and easier to be understood and used than any way of elicitation of user preferences necessary for controlling the selection by weights. Problems related to the implementation of the latter approach are well-illustrated e.g. in a survey of experiences provided by Lootsma et. al in [LAP94].

• We do not recommend the use of a nadir point information. For problems with more than two criteria finding a nadir point maybe a difficult problem (cf e.g. [IsS87]). This can be illustrated for example by the summary of the RWQM problem (cf Section 2). The pay-off matrix (presented in Table 1) shows that the approximation of the nadir

Criterion	Criteria value					
minimized	TAC	IC	OMRC	DOmin	BODmax	NH4max
TAC	1.55	0.0	1.55	0.14	25.8	4.71
IC	6.06	0.0	6.06	3.60	11.6	3.84
OMRC	1.55	0.0	1.55	0.14	25.8	4.71
DOmin	14.4	34.3	8.45	5.38	9.81	1.70
BODmax	14.4	32.7	8.45	5.38	9.81	1.70
NH4max	14.4	29.9	8.45	5.38	9.81	1.70
Utopia	1.55	0.0	1.55	5.38	9.81	1.70
Nadir	14.4	50.3	8.69	0.14	25.8	4.71

Table 1: The pay-off table for the RWQM problem: 6 criteria are examined.

point for the criterion *IC* found during the analysis is much better than the worst value found during selfish optimizations. Moreover, this value is much smaller than a true nadir value (which is larger than 80). The row labeled "Utopia" summarizes the criteria's values obtained from selfish solutions. The row labeled "Nadir" summarizes the worst values of criteria that were obtained in any part of the analysis reported in [MSW95]. This explains why the nadir point values for criteria IC and OMRC are worse than the worst values from selfish optimizations. The analysis presented does not contain a case with a worst value for criterion IC.

• Definition of weights by (23) better reflects trade-offs (implied by a pair $\bar{q}, \underline{q}$) than weights computed using a pair $\bar{q}, q^U$, especially in situations, when the selected aspiration point component for a criterion is much closer to the corresponding nadir than to the utopia value or if the difference between aspiration and reservation is relatively small. For example, the most interesting region for examination of the problem summarized in Table 1 is for the relatively small range of criteria TAC, INV and BODmax (for aspiration-reservation pairs [8, 10] and [10,15], [10,12], respectively) and a relatively large range for criterion NH4max (with aspiration value about 3).

In particular, if the user specifies only aspiration and reservation points, then the component achievement function (18) has the same form as (14,15) and the corresponding weights are defined by:

$$w_i = 1/|\bar{q}_i - \underline{q}_i| \qquad (23)$$

The component achievement functions defined by (18) provide indirectly scaling coefficients that are controlled by a user in a way that is consistent with the specification of his/her preferences. Therefore the consistency of scaling coefficients for different criteria (including differences in magnitudes of the criteria's values) is assured by LP-MULTI.

5.2 Types of criteria

The user of LP-MULTI selects (during the initialization of LP-MULTI) for each criterion its type, which must be either minimization of maximization. During the interactive analysis of the problem each criterion can be, possibly temporary, stabilized. In order to simplify the presentation we have assumed so far that all criteria are minimized. In this subsection we will deal with the other two types of criteria.

It is easy to specify the component achievement function (18) in such a way that it handles both minimized and maximized criteria. Since the types of criteria are specified during the initialization of the problem, this condition can be met by an appropriate organization of the interaction. The aspiration and reservation (and possibly also points between them) can only be defined in such a way that the resulting α_{ji} defined by (19) are negative for criteria that are minimized and are positive for maximized criteria. Hence the resulting piece-wise linear function $u_i(q_i)$ is decreasing, if a criterion is minimized and is increasing, if a criterion is maximized but in both cases the function is strictly concave.

Let us consider i-th criterion that is stabilized, which means that a user wants a value of this criterion to be close to a given target value $\tilde{q}_i$. A stabilized criterion q_i is replaced by an auxiliary criterion (which is minimized) where $\tilde{q}_i$ is a given target value, which should attain the stabilized criterion q_i. This replacement is done by the LP-MULTI and is hidden from a user (cf Appendix A.3 for details). We discuss here only the meaning of utopia, aspiration, reservation and nadir points for the auxiliary criterion because those points have different values and interpretation than the corresponding points of the stabilized criterion. The utopia component for the auxiliary criterion is equal to 0, while the nadir point component is equal to $\max(||\tilde{q} - q_i^U||, ||\tilde{q} - q_i^N||)$, where q_i^U, q_i^N are utopia and nadir points of the criterion q_i, respectively. Both aspiration and reservation levels have to be specified for the auxiliary criterion by an interactive tool. The FT-TOOL does this in

Figure 6: Stabilization of a criterion implemented in FT-TOOL. The thin line corresponds to the trapezoidal function initially generated by FT-TOOL. The solid line represents the users' modifications.

the following way (cf Figure 6 for illustration): the shape of the component achievement function (18) is changed from a strictly monotone (for criteria that are minimized or maximized) to a symmetric trapezoidal for stabilized criteria. The meaning of aspiration and reservation levels for stabilized criteria correspond to the meaning of such criterion,

which is minimization of a deviation of q_i from a given target value $\tilde{q}_i$, marked in Figure 6 by **T**. Therefore the aspiration level corresponds to the desired range of deviation from the currently defined target value **T** (segments TA on Figure 6) and the reservation level represents the maximum tolerable deviation (segments TR). The segments TA might be empty but the segments TR must have some minimum length (cf Appendix A.1 for implementation assumptions).

Finally we would like to point out, that the component achievement function (18) for a stabilized criterion with the corresponding aspiration level equal to zero has a similar form to the membership function used by Nakayama in [Nak94] for the illustration of relations between fuzzy mathematical programming and multi-objective programming.

The two basic types of criteria provide a framework for definition of more complex criteria. For example, for dynamic problems it is typical to deal with trajectories. In such cases one can easily define auxiliary variables which have interpretation of a deviation from a given trajectory. Depending on the application either a trajectory should be followed, or only surplus (or deficit) should be minimized. The corresponding auxilary variables can be defined as follows:

$$\max_{t \in T} \| x_t - \bar{x}_t \| \tag{24}$$

$$\max_{t \in T}(x_t - \bar{x}_t) \tag{25}$$

$$\min_{t \in T}(x_t - \bar{x}_t) \tag{26}$$

where T is set of indices and $\bar{x}_t$ is a given reference trajectory. Such variables can be used as criteria: minimized for the first two cases and maximized for the last case.

6   A user guide for LP-MULTI

We summarize in this section information that is useful for efficient use of LP-MULTI. The selection of information is aimed at a user who wants to understand how the multiple-criteria model analysis described in previous sections and implemented in LP-MULTI can be used for examination of a particular model. Therefore the implementation details (that might be interesting for a system analyst and for readers, who implement other approaches to ARBDS) have been moved to Appendix A.

The two useful techniques of the model analysis, namely, *inverse simulation* and *soft constraints* can be easily implemented with help of LP-MULTI, provided that simple provisions are made during the specification of the core model. The corresponding requirements are discussed in Section 6.1 and 6.2, respectively. Section 6.3 provides a summary of the preparatory stage for interactive analysis. Finally Section 6.4 contains an overview of the interactive analysis of a model supported by LP-MULTI.

6.1   Soft constraints

Typical advice for traditional single-criterion optimization that requires treatment of all but one goals as constraints is to specify two types of constraints, so-called hard and soft constraints, which correspond to *must* and *should* types of conditions, respectively. In this way hard requirements for goals (that are usually not attainable) can be softened by representing requirements for the values of goals as *soft* constraints. Implementation of this approach in the framework of single-criterion optimization is discussed in more detail in [Mak94a]. A properly specified core model for multiple-criteria analysis does not

require *soft* constraints because all criteria can be treated in a uniform way. However, users familiar with implementations of soft constraints may want to examine the following interpretation of soft constraints within the framework of ARBDS.

For the sake of simplification let us consider in detail only one type of soft constraint, namely *an upper bound* type:

$$x_i - x_i^S \leq \bar{x} \tag{27}$$

where an additional variable x_i^S represents a surplus, i.e. a violation of the original constraint $x_i \leq \bar{x}$. Typically, the variables x_i^S enter goal function as penalty terms which are equivalent to application of the scalarizing function (7), where the corresponding weight w_i is the same as the penalty term coefficient. Therefore, soft constraints have drawbacks similar to those discussed for the scalarizing function (7). Hence, it is easier and more efficient to treat directly the variable x_i as a minimized criterion, instead of generating a soft constraint in form (27). Note, that for such criterion the value $\bar{x}$ can serve as an aspiration level and a maximum tolerable surplus can be used as a reservation level. Thus, a DM can use an easy and intuitive way (based on ARBDS) for examination of trade-offs between goals.

For the general type of constraints (2) one can define an additional variable as

$$x_j^S - A_j x = 0 \tag{28}$$

where A_j is the j-th row of the matrix A, and the additional variable x_j^S can be either stabilized or minimized or maximized, depending on the meaning the j-th constraint.

Nakayama reports in [Nak94] the following approach to be very effective in practice. For the component achievement function (14) one can consider instead of the corresponding constraint (39) a constraint in the form:

$$\beta_i x - w_i(q_i - \bar{q}_i) \leq 0 \tag{29}$$

where x is the auxiliary variable (cf Section A.3 for details) and the user controlled parameter β_i allows for the treatment of the i-th criterion as:
- a criterion in the sense presented in Section 4.3 (for $\beta_i = 1$),
- a hard constraint for the criterion value $\bar{q}$ (for $\beta_i = 0$).

Selecting values of $\beta_i \in (0,1)$ helps to consider intermediate (between a hard constraint and no constraint for a criterion value) interpretations of the corresponding criterion. A comparison of the effectiveness of such an approach with the approach based on the piece-wise component achievement function (18) is an open question.

6.2 Inverse simulation

So-called inverse simulation (cf [Wie92a]) is a very useful technique for examining decisions in the two typical situations, when:
- specification of a set of feasible decisions is not easy,
- it is likely that the values of decision variables that a DM would like to examine will be infeasible.

Wierzbicki proposed in [Wie92a] a general form of optimization problem for the inverse simulation, namely:

$$\hat{x} \in \text{Arg} \min_{x \in X_0} (\rho \parallel q - \bar{q} \parallel + (1 - \rho) \parallel x - \bar{x} \parallel) \tag{30}$$

where the parameter $\rho \in [0,1]$ controls the trade-off between the desired values of goals and the specified values of decisions denoted by $\bar{q}$ and $\bar{x}$, respectively.

A pure inverse simulation takes into account only given values of decisions, therefore $\rho = 0$ and the corresponding optimization problem can be written as:

$$\hat{x} \in \text{Arg} \min_{x \in X_0} \| x - \bar{x} \| \tag{31}$$

Both forms of the inverse simulation can be easily implemented as a single-criterion optimization problem (cf e.g. [Mak94a] for details). However, it is often desirable to consider a deviation from a given set of decisions as one of the criteria within a uniform framework of multiple-criteria model analysis. If a number of the considered decision variables is small, then the easiest way is to use the selected variables as stabilized criteria (cf Section 5.2). Otherwise, one can add to the corresponding core model additional variables $x_i^+ \geq 0$, $x_i^- \geq 0$ defined by additional constraints in the following way:

$$x_i + x_i^+ + x_i^- = \bar{x}_i \qquad i \in I \tag{32}$$

where I is a set of indices of variables to be considered in the inverse simulation. An additional variable x^S, that fulfills the following two conditions:

$$x_i^+ - x^S \leq 0. \qquad i \in I \tag{33}$$

$$x_i^- - x^S \leq 0. \qquad i \in I \tag{34}$$

can later be used as a minimized criterion. In a similar way one may define separate criteria for groups of variables whose indices belong to K sets $I_k, k \in [1, K]$.

6.3 Preparatory stage

The preparatory stage of the model analysis by LP-MULTI is done automatically (i.e. without interaction with a user). The preparatory stage is done only, if initialization is requested[15] from the LP-MULTI. In a typical application the preparatory stage is done by a developer of a DSS but a user may want also to start the analysis from scratch. Therefore we summarize the functions performed during the preparatory stage.

The starting point of multiple-criteria analysis of a problem is specification of the core model (cf Section 3.2). The formulation of the core model should be provided in the LP-DIT format[16]. In order to prevent the consistency of the analysis, LP-MULTI treats a modification of the core model specification as a fatal error. Therefore, if there is a reason for modification of the core model during the analysis, then the analysis should be reinitialized using the modified core model.

Specification of the criteria is done during the initialization. The specification includes for each criterion:
- name of the criterion (maximum 6 characters long).
- type of the criterion (MIN for minimization, MAX for maximization).
- name of variable that define the criterion.

Specification is read from a free format ASCII file, therefore names should be composed of printable characters (but cannot contain blanks, which serve as the token's separator). Each criterion has to be specified on a separate line.

LP-MULTI first checks the consistency of the criteria definition. Assuming that n criteria are correctly specified LP-MULTI will perform:

---

[15]This is implementation dependent and is typically done by specification of a flag in the command line arguments of the program or by selection of an option from a menu.

[16]See the description of LP-DIT in [Mak94b]. LP-DIT provides also a utility program for conversion of models specified in the MPS format to the binary format used by LP-DIT.

- n selfish optimizations (i.e. single criterion optimizations for each criterion separately). This substage will provide the utopia point.
- n maximizations of the achievement scalarizing function (18) with only one criterion active for each optimization. This substage results in an approximation of the nadir point.
- Computation of so-called compromised solution. This is a Pareto solution for a problem with the aspiration levels set to the utopia point and the reservation point equal to the approximation of the nadir point.

The preparatory stage is invoked only if initialization is explicitly requested. Therefore it is skipped for a typical interactive session.

6.4 Interactive analysis

The preparatory stage (cf Section 6.3) consists of processing the core model definition and of criteria specification followed by $2n + 1$ optimizations runs. As a result the utopia and nadir points[17] and the compromise solution are calculated.

The interactive analysis is done by FT-TOOL documented in [GrM95]. It can be done also with another modular tool supporting similar functionality (such a tool may provide additional, problem specific, analysis of a solution). Therefore we provide here only an outline of this part of the interactive analysis that is common for all applications of ARBDS.

Conventionally, in the context of the interactive model analysis, the term *iteration* is used for a sequence composed of:

Analysis of a Pareto solution: The last computed Pareto solution (the compromise solution for a first, after initialization, iteration) is presented to a DM. Typically, for each criterion, at least values of the last[18] computed Pareto solution, aspiration, reservation, as well as utopia and nadir points are displayed. Additionally, problem specific data may also be processed and displayed by problem specific tools. For example, for the RWQM DSS the additional information includes graphs of different types of waste concentration profiles (maximum, over the set of monitoring points, values are treated as criteria). Interactive examination of a value of any variable defined in the core model is often desired by a user and therefore should be also available.

Optional change of the status of criteria: The user may change the status of any criteria in the following way: Any active criterion may be stabilized or declared as non-active. For a non-active criterion the component achievement function is not defined by the user (cf Section A.3 for the implementation details). A non-active criterion may become active. The status of criteria is easily recognized for the FT-TOOL implementation: active criteria have increasing or decreasing (for maximized and minimized criteria, respectively) component achievement function u_i, a trapezoidal function u_i for stabilized criteria and no component achievement function for non-active criteria.

---

[17]It is difficult to compute a true nadir point for many problems with more than two criteria and in practice only an approximation of nadir point is available. For the sake of brevity we will call the current approximation of nadir simply a nadir.

[18]Many implementations display also values of previously computed points and provide a possibility of management of solutions, cf [GrM95] for more details.

Specification of preferences: The user specifies his/her preferences in the form of component achievement function u_i for each active criterion. This is equivalent to specification of an extended-valued membership function in terms of fuzzy multicriteria-programming. In the simplest approach, which is typical for first iterations, the user specifies only aspiration and reservation levels. The previously selected aspiration and reservation points are used as the starting point for the specification. Therefore, the user may only modify selected points while keeping the remaining aspiration and reservation levels unchanged (see Figure 5 on page 20). More advanced users may specify piece-wise linear component achievement functions. Up to this point of the iteration a user may freely switch between the analysis of solutions, change of criteria status and the specification of his/her preferences. At this point the user may decide to break the analysis (which can be continued during another session). The remaining part of the iteration is performed automatically, once the user confirms his/her preferences for the current iteration.

Generation of an optimization problem: LP-MULTI converts the current formulation of the multiple-criteria problem into the corresponding single-criterion problem, which is generated in the LP-DIT format. Before the conversion a verification of the component achievement functions is made. This verification is done in order to avoid numerical problems and it rarely changes the specified preferences. Nevertheless the user is asked to accept the modifications made by LP-MULTI and/or FT-TOOL, if any modifications are necessary (cf Section A.1 for details). The conversion of the multiple-criterion problem into the equivalent parametric single-criterion optimization problem that is implemented in LP-MULTI is documented in Section A.3. Note, that the corresponding single-criterion problem has always a feasible and bounded solution provided that such solutions exist for all single-criterion optimizations (for each criterion selected, in row, to be the objective function) performed for calculation of the utopia point. This is typically a case for a properly formulated core model and a selection of criteria that correspond to real-life problems.

Solution of an optimization problem: A modular solver is used for solving the corresponding single-criterion optimization problem. The solution is generated in the LP-DIT binary format, therefore it is easy to retrieve any part of the solution both for a next iteration within LP-MULTI, and for additional, problem specific analysis.

LP-MULTI only supports decision analysis and it is exclusively up to the user's discretion how the preferences are specified and how the solutions are evaluated. Therefore, we limit our suggestions to only one. Namely, it is strongly recommended to continue the analysis until the selected solution has the values of all criteria between aspiration and reservation levels specified by the user. There is a strong justification for this advice. Namely, the component achievement scalarizing functions u_i are defined by the user only for the criteria values between aspiration and reservation levels. For the values of criteria between utopia and aspiration, and between reservation and nadir points, the functions u_i are defined by LP-MULTI. This definition (cf Section 5) usually reflects well typical preference structure for criteria values outside the range between aspiration and reservation levels. Nevertheless, the user should have full control of the specification of the function that reflects his/her preferences in the region that determines the corresponding Pareto solution. Moreover, it is usually easy to adjust the aspiration and/or reservation level in such a way that the next solution is between those points.

We would also like to point out, that the approximation of nadir is updated for every

optimal solution analized during the interaction. For problems with more than two criteria calculation of a true nadir point is usually difficult. This is one of the reasons why the nadir point in LP-MULTI has only informative meaning. However, the user should not be surprised, if he/she notices that a value of a nadir point component worsens during the interaction (cf Section 5 for illustration and references).

Current implementations of GUI (Graphical User Interface) make it possible to design and implement the user interface in such a way, that during the specification of preferences (expressed in the form of the component achievement functions) the user can analyze (in other windows on the screen, possibly using also another computer) current and previous solutions in different forms, including graphs and tables.

7 Implementation of a DSS for the Nitra case study

In order to illustrate usage of modular software tools for implementation of a DSS, we summarize below tools that were used for the DSS for RWQM (cf Section 2). The RWQM is only one component of the software that is developed for the Nitra case study (cf [SMPK94] for a documentation of the case study). We outline here the RWQM structure as an illustration of an application of reusable modular software tools. This DSS is composed of the following elements:

Problem generator – generates a core model that corresponds to the model specification documented in [MSW95] and outlined in Section 2.

LP-DIT – Data Interchange Tool for Linear Programming Problems (cf [Mak94b] for details) is a prototype implementation for handling data that define a MIP or LP problem. LP-DIT provides an easy and efficient way for the definition and modification of MIP problems, as well as the interchange of data between a problem generator, a solver, and software modules that serve for problem modification and solution analysis.

FT – Fuzzy Tool is a prototype implementation of an interactive tool for specification of user preferences in terms of fuzzy sets (cf [GrM95] for details).

LP-MULTI – Modular tool documented in this paper. It currently uses LP-DIT for data handling and FT-TOOL for interaction with a user.

MOMIP – Modular Optimizer for Mixed Integer Programming (cf [OgZ94] for details). It also uses LP-DIT for data (both problem specification and solution) handling.

Note, that the problem generator is the only software module that is specific for a RWQM model. Other software tools can be applied in the development of other DSS.

This approach has several important advantages that, for the sake of brevity, will not be discussed fully here. Instead, we summarize only the functional structure of the software.

Data handling: The data used in the model (cf [MSW95] for details) is the output from the simulation model documented in [SMPK94] and has been combined in one free-format ASCII file. The data file is composed of several segments containing groups of related data and a description of data items. The organization of the data file is flexible and provides adequate documentation so that its organization is easy to modify.

Problem generation: A problem-specific model generator (subsequently referred to as *the generator*) has been implemented. The generator generates in LP-DIT format a *core* model, according to the assumptions described in Section 3.2.

Multicriteria problem analysis: The core model is used by LP-MULTI for the generation of a multicriteria problem. First, the utopia point, an approximation of the nadir point, and a compromise solution are automatically computed. After this stage is completed, the interactive phase is started. In this phase the FT-TOOL allows for an interactive analysis of solutions and the selection of new aspiration and reservation levels. A user can also change the status of a criterion and optionally specify preferences in terms of fuzzy sets. The solutions are stored, and a summary of solutions is logged, so that it is easy to continue analysis during another session and to produce a report based on a set of selected solutions.

Solution of multi-criteria problem: LP-MULTI converts the multicriteria problem and generates a corresponding MIP problem in the LP-DIT format. Then it calls the MOMIP solver. The currently examined model has about 800 rows and 800 variables (including 90 binary variables), and it typically takes less than one minute to solve it on the Sun Workstation.

Reporting: Tools for examining complete results are currently very simple. One can obviously examine complete solutions (i.e. values of all variables and constraints). Additionally, a simple tool has been developed for plotting the resulting ambient concentrations along a river for each constituent. However, all the information needed for the interaction is available with the help of FT-TOOL and of standard Unix tools.

8 Conclusion

LP-MULTI has been implemented so far only for LP and MIP problems. However, it can be used also in DSS that deal with non-linear problems, provided that a linear part of the corresponding core model will be generated in the LP-DIT format. This might be a practical solution because for many problems a linear part contains majority of constraints.

Current implementation of LP-MULTI forces the scalarizing function (18) to be strictly concave. This requirement will be relaxed in a future implementation by following the approach proposed by Inuiguchi in [IIK90].

A prototype of the LP-MULTI has been implemented and tested for the Nitra case study (cf Section 7). The current version of LP-MULTI is the result of several applications made for different problems. However, it is still a prototype and therefore criticism and suggestions (for both methodological background and the implementation of LP-MULTI) will be appreciated. The author will try to incorporate the suggestions into the distributable version of LP-MULTI, which should be ready by Spring 1995.

References

[And89]   S. Andriole, *Handbook of Decision Support Systems*, TAB Professional and Reference Books, Blue Ridge Summit, PA, 1989.

[BdMTL71] R. Benayoun, J. de Mongolfier, J. Tergny and O. Larichev, *Linear programming with multiple objective functions: STEP method (STEM)*, Mathematical Programming **1** (1971) 366–375.

[ChC67] A. Charnes and W. Cooper, *Management Models and Industrial Applications of Linear Programming*, J. Wiley & Sons, New York, London, 1967.

[GdHR$^+$93] O. Güler, D. den Hertog, C. Roos, T. Terlaky and T. Tsuchiya, *Degeneracy in interior point methods for linear programming: a survey*, Annals of Operations Research **46** (1993) 107–138.

[GoM95] J. Gondzio and M. Makowski, *Solving a class of LP problems with primal-dual logarithmic barrier method*, European Journal of Operational Research **80**, no. 1 (1995) 184–192.

[Gon94] J. Gondzio, *Presolve analysis of linear programs prior to applying the interior point method*, Technical Report 1994.3, Department of Management Studies, University of Geneva, Geneva, Switzerland, 1994.

[GrM95] J. Granat and M. Makowski, *FT, A tool for interactive specification of user preferences in terms of fuzzy sets*, Working Paper WP-95-xx, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1995. (forthcoming).

[GrW94] J. Granat and A. P. Wierzbicki, *Interactive specification of DSS user preferences in terms of fuzzy sets*, Working Paper WP-94-29, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[HaH74] Y. Haimes and W. Hall, *Multiobjectives in water resource systems analysis: the surrogate trade-off method*, Water Resources Research **10** (1974) 615–624.

[HuJ90] I. Huntley and D. James, eds., *Mathematical Modelling, A Source Book of Case Studies*, Oxford University Press, Oxford, New York, Tokyo, 1990.

[IIK90] M. Inuiguchi, H. Ichimashi and Y. Kume, *A solution algorithm for fuzzy linear programming with piecewise linear membership function*, Fuzzy Sets and Systems **34** (1990) 15–31.

[IsS87] H. Isermann and R. E. Steuer, *Computational experience concerning payoff tables and minimum criterion values over the efficient set*, European J. Oper. Res. **33** (1987).

[JdJRT93] B. Jansen, J. de Jong, C. Ross and T. Terlaky, *Sensitivity analysis in Linear Programming: Just be careful !*, Technical Report AMER.93.022, Shell Internationale Research, Maatschappij B.V., The Netherlands, 1993.

[Kee92] R. Keeney, *Value Focused Thinking, A Path to Creative Decisionmaking*, Harvard University Press, Harvard, 1992.

[KMW92] P. Korhonen, H. Moskowitz and J. Wallenius, *Multiple criteria decision support – a review*, European J. Oper. Res. **63** (1992) 361–375.

[KOCZ93] G. Klein, J. Orasanu, R. Calderwood and C. Zsambok, eds., *Decision Making in Action: Models and Methods*, Ablex Publishing Co., Norwood, NJ, USA, 1993.

[KoL84] P. Korhonen and J. Laakso, *A visual interactive method for solving the multiple-criteria problem*, in Interactive Decision Analysis, M. Grauer and A. Wierzbicki, eds., Lecture Notes in Economics and Mathematical Systems, vol. 229, Springer Verlag, Berlin, New York, 1984, pp. 146–153.

[KoW89] P. Korhonen and J. Wallenius, *Observations regarding choice behaviour in interactive multiple criteria decision-making environments: An experimental investigation*, in Methodology and Software for Interactive Decision Support, A. Lewandowski and I. Stanchev, eds., Lecture Notes in Economics and Mathematical Systems, vol. 337, Springer Verlag, Berlin, New York, 1989, pp. 163–170.

[LAP94] F. Lootsma, T. Athan and P. Papalambros, *Controlling the search for a compromise solution in multi-objective optimization*, Report 94-12, Department of Mechanical Engineering & Applied Mechanics, The University of Michigan, Ann Aborn, Michigan, USA, 1994.

[LeW89] A. Lewandowski and A. Wierzbicki, eds., *Aspiration Based Decision Support Systems: Theory, Software and Applications*, Lecture Notes in Economics and Mathematical Systems, vol. 331, Springer Verlag, Berlin, New York, 1989.

[LMS94] I. Lustig, R. Marsten and D. Shanno, *Interior point methods for linear programming: Computational state of art*, ORSA Journal on Computing **6**, no. 1 (1994) 1–14.

[Mak94a] M. Makowski, *Design and implementation of model-based decision support systems*, Working Paper WP-94-86, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[Mak94b] _____, *LP-DIT, Data Interchange Tool for Linear Programming Problems, (version 1.20)*, Working Paper WP-94-36, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[MSW95] M. Makowski, L. Somlyódy and D. Watkins, *Multiple criteria analysis for regional water quality management: A Nitra river basin case study*, Working Paper WP-95-xx, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1995. (forthcoming).

[Nak94] H. Nakayama, *Aspiration level approach to interactive multi-objective programming and its applications*, Working Paper WP-94-112, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[OgL92] W. Ogryczak and S. Lahoda, *Aspiration/reservation-based decision support — a step beyond goal programming*, Journal of Multi-Criteria Decision Analysis **1**, no. 2 (1992) 101–117.

[OgZ94] W. Ogryczak and K. Zorychta, *Modular optimizer for mixed integer programming*, MOMIP *version 2.1*, Working Paper WP-94-35, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[Rap89] A. Rapoport, *Decision Theory and Decision Behaviour, Normative and Descriptive Approaches*, Theory and Decision Library, Mathematical and Statical Methods, vol. 15, Kluwer Academic Publishers, Dordrecht, Boston, London, 1989.

[Sak93] M. Sakawa, *Fuzzy Sets and Interactive Multiobjective Optimization*, Plenum Press, New York, London, 1993.

[SeS88] F. Seo and M. Sakawa, *Multiple Criteria Decision Analysis in Regional Planning: Concepts, Methods and Applications*, D. Reidel Publishing Company, Dordrecht, 1988.

[SMPK94] L. Somlyódy, I. Masliev, P. Petrovic and M. Kularathna, *Water quality management in the Nitra River Basin*, Collaborative Paper CP-94-02, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1994.

[SNT85]   Y. Sawaragi, H. Nakayama and T. Tanino, *Theory of Multiobjective Optimization*, Academic Press, New York, 1985.

[SpW93]   R. Sprague and H. Watson, eds., *Decision Support Systems: Putting Theory into Practice*, Prentice Hall, Englewood Cliffs, N.J., 1993.

[Ste86]   R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, and Application*, J. Wiley & Sons, New York, 1986.

[Thi93]   R. J. Thierauf, *Creative Computer Software for Strategic Thinking and Decision Making – A Guide for Senior Management and MIS Professionals*, Quorum Books, London, 1993.

[Tur93]   E. Turban, *Decision Support and Expert Systems: Management Support Systems*, Macmillan Publishing Company, New York, 1993.

[TvK85]   A. Tversky and D. Kahneman, *The framing of decisions and philosophy of choice*, in Behavioral Decision Making, G. Wright, ed., Plenum, New York, 1985, pp. 25–42.

[vG91]   J. P. van Gigch, *System Design Modeling and Metamodeling*, Plenum Press, New York, London, 1991.

[Vin89]   P. Vincke, *Multicriteria Decision-aid*, J. Wiley & Sons, Chichester, New York, 1989.

[WeW93]   J. Wessels and A. Wierzbicki, eds., *User-Oriented Methodology and Techniques of Decision Analysis and Support*, Lecture Notes in Economics and Mathematical Systems, vol. 397, Springer Verlag, Berlin, New York, 1993.

[Wie77]   A. Wierzbicki, *Basic properties of scalarizing functionals for multiobjective optimization*, Mathematische Operationsforschung und Statistik, s. Optimization **8** (1977) 55–60.

[Wie80]   ———, *The use of reference objectives in multiobjective optimization*, in Multiple Criteria Decision Making, Theory and Applications, G. Fandel and T. Gal, eds., Lecture Notes in Economics and Mathematical Systems, vol. 177, Springer Verlag, Berlin, New York, 1980, pp. 468–486.

[Wie82]   ———, *A mathematical basis for satisficing decision making*, Mathematical Modelling **13** (1982) 5–29.

[Wie86]   ———, *On the completeness and constructiveness of parametric characterizations to vector optimization problems*, OR Spectrum **8** (1986) 73–87.

[Wie92a] ———, *Multi-objective modeling and simulation for decision support*, Working Paper WP-92-80, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1992.

[Wie92b] ———, *Multiple criteria games: Theory and applications*, Working Paper WP-92-79, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1992.

[Wie92c] ———, *The role of intuition and creativity in decision making*, Working Paper WP-92-78, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1992.

[Wil90]   H. P. Williams, *Model Building in Mathematical Programming*, J. Wiley & Sons, Chichester, New York, 1990.

[WiM92]   A. Wierzbicki and M. Makowski, *Multi-objective optimization in negotiation support*, Working Paper WP-92-07, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1992.

[Yu85]   P. L. Yu, *Multiple-Criteria Decision Making: Concepts, Techniques, and Extensions*, Plenum Press, New York, London, 1985.

[Yu90]   _____, *Forming Winning Strategies, An Integrated Theory of Habitual Domains*, Springer Verlag, Berlin, New York, 1990.

[Zad65]   L. Zadeh, *Fuzzy sets*, Information and Control **8** (1965) 338–353.

A   Implementation details of LP-MULTI

We summarize in this Appendix a number of implementation details that are usually hidden from a user of a DSS. However, those details (and underlying assumptions) might be interesting for readers, who want to analyze one or more of the following methodological and technical problems:

- Hidden conversions implemented to avoid numerical problems and the default tolerances.
- The conversion implemented for handling stabilized criteria.
- The conversion of the multiple-criteria problem to an equivalent single-criterion problem and the formulation of the corresponding mathematical programming problem.

    FT-TOOL supports the interaction with the user in the current implementation of LP-MULTI. Therefore a reader interested in the related topics should consult [GrM95].

A.1   Hidden conversions and tolerances

In order to avoid numerical problems it is sometimes necessary to slightly modify the specification of the problem provided by the user. Such situations do not occur often and the modifications usually do not change the problem specification in a substantial way. Nevertheless, the user is asked for acceptance of the changes made by LP-MULTI and/or FT-TOOL in every case when a modification is made.

The first type of modification occurs, if the user specifies a component achievement function (18) that is not a strictly concave function. In such a case the points (between aspiration and reservation levels) that cause the function to be non-concave are removed and the modified function is displayed. The user either accepts the changes or may modify the function again.

The second type of conversion occurs, if the core model practically fixes a value of a criterion. In such a situation the following condition holds:

$$|q_i^U - q_i^N| \leq \eta \max\{|q_i^U|, |q_i^N|\} \tag{35}$$

where η is a given value (cf below) and the i-th criterion is converted into a non-active criterion. Note that this is also an indication, that this criterion is most probably wrongly selected.

The third type of modification is applied, if the user specifies one of the following pairs of points that differ less than a given threshold value δ_i defined for i-th criterion by:

$$\delta_i = \eta |q_i^U - q_i^N| \tag{36}$$

The following actions are taken by LP-MULTI depending on which pair of points is too close:

utopia and aspiration: the utopia point is removed from the set of points that define the piece-wise linear function (18).

aspiration and reservation: the criterion is converted into a stabilized criterion with the target value $\tilde{q}_i$ equal to the value of the aspiration level $\bar{q}_i$ (cf Section A.2 for more details).

reservation and nadir: the reservation point is removed from the set of points that define the piece-wise linear function (18).

reservation range for stabilized criterion: the reservation range (represented by the segment TR in Figure 6) that represents the maximum tolerable deviation from the reference value is increased to $\eta|q_i^U - q_i^N|$.

The following tolerances are set in the current implementation of LP-MULTI:

- The ϵ parameter used in eq. (18) for preventing computation of weakly Pareto solutions is set to 10^{-4}. Some authors suggest smaller values (e.g. Nakayama in [Nak94] reports implementations with $\epsilon = 10^{-6}$). Our experience shows that smaller values of ϵ create two types of problems. First, the scaling of the resulting optimization problem is difficult. Assuming that the core model is well scaled, a very small value of ϵ deteriorates the scaling of the resulting LP problem. Second, a too small value of ϵ often results in coefficient value in the corresponding LP problem that is smaller than the optimality tolerances implemented in a solver. This in turn results in a sub-optimal solution reported as the optimal solution, which means that a weakly Pareto solution can be reported contrary to the expectations of the user.

- The threshold parameter η used in eqs. (35) and (36) is set to 0.01. This is due to the observation that users rarely want to distinguish points in the pairs listed above, if the points differ by less than 1% of the value represented by one of them.

A.2 Stabilized criteria

Let us consider i-th criterion that is stabilized, which means that a user wants a value of this criterion to be close to a given target value $\tilde{q}$. A stabilized criterion q_i is replaced by an auxiliary criterion (which is minimized) q_i^a defined as:

$$q_i^a - q_i^+ - q_i^- = 0 \tag{37}$$

and the two auxiliary non-negative variables q_i^+, q_i^- fulfill the following constraint:

$$q_i + q_i^+ - q_i^- = \tilde{q}_i \tag{38}$$

The corresponding utopia, aspiration, reservation and nadir points components of i-th criterion are defined by LP-MULTI in the way described in Section 5.2.

A.3 Conversion of MCLP to LP

In order to achieve maximization of the scalarizing achievement function $\mathcal{S}(q, \bar{q}, \underline{q})$ defined by (16) using the piece-wise linear component achievement functions u_i defined by (18) we introduce auxiliary variables x, y_i, $i \in [0, n]$, x is unconstrained, $y_i \geq 0$. The following additional constraints are defined:

$$x + y_i - u_{ji} \leq 0, \qquad j \in [1, p_i], \qquad i \in [0, n] \tag{39}$$

where the segments u_{ji} are defined by (18). Finally we replace maximization of (16) by minimization of the auxiliary LP goal function in the form:

$$-x - \epsilon \sum_{i=1}^{n} (\gamma_i y_i + (1 - \gamma_i) s_i q_i) \tag{40}$$

where γ_i is equal to 1 or 0, for active criteria and non-active criteria, respectively, and the scaling coefficients s_i are defined by (17). Note, that in the actual implementation variables y_i, the component achievement functions u_{ji} and the corresponding constraints (39) are not generated for non-active criteria.

Observe, that the auxiliary variables enter only constraints (39). Therefore at least one y_i is equal to zero. Hence, an optimal value of the auxiliary variable x has an important interpretation, namely when $x > 0$ then all active criteria have values better than a reservation level, $x > 1$ indicates that all criteria's values are better than an aspiration level while a negative x shows that at least one criterion has a value worse than a reservation level. The value of the component achievement function (18) for the i-th criterion is equal to $x + y_i$. Note that scaling of each criterion is based on the component achievement functions (18) that are defined by the user for criteria's values between the corresponding aspiration and reservation levels. This justifies the advice given in Section 6 to continue an interactive analysis of the problem as long as the resulting value of x fulfills the condition $0 < x < 1$.

A.4   Names used for auxiliary rows and columns

Names of auxiliary variables (rows and columns) are used for conversion of the multicriteria problem into a single-criterion LP and are not important for a user who deals rather with names of criteria and of variables specified for the core model. However, we provide the rules for generating auxiliary names for those users who would like to analyze a full solution of the corresponding single-criterion problem as provided by a solver.

The following conventions are adopted while generating the auxiliary LP (or MIP) problem:

- LP-MULTI checks, if the LP problem defined in LP-DIT format has a goal function. If the goal function is already defined and if the problem is specified for minimization, then elements specified in (40) are generated to the corresponding objective row. Otherwise a new goal function named mc_goal is generated in the form defined by (40).
- The auxiliary variable x has the name <mc_aux
- The auxiliary variables y_i have names y^name, where the string name is replaced in the generated names by the i-th criterion name.
- The auxiliary variables u_{ji} have names i^name, where the string name is replaced in the generated names by the i-th criterion name and the index i (corresponding to a number of a segment of the piece-wise linear function u_i) is represented in the generated names by digits 0 through 9 and letters starting from A. In the current implementation the number of segments must be smaller than 14.
- Auxiliary row names are generated for each criterion as i^name, where $i \in [0, p_i]$ and name is a criterion name, p_i is a number of points that define the i-th component achievement function. The index i is defined in the same way as for auxiliary variables.
- The correspondence between names of auxiliary rows and indices i is as follows:
  (0) – for the equation (39)
  (1 through n) – for the equations (18)
  (MAX_PTS) – for the equation (37)
  (MAX_PTS + 1) – for the equation (38)
- The correspondence between names of auxiliary variables and indices i is as follows:
  (MAX_PTS) – for the variable q_i^a
  (MAX_PTS+1) – for the variable q_i^+
  (MAX_PTS+2) – for the variable q_i^-

B Availability of software

Currently LP-MULTI works together with a prototype of the FT-TOOL (cf [GrM95]), which supports interactive specification of preferences in terms of fuzzy sets. Therefore the use of LP-MULTI is currently restricted to IIASA and should be done in cooperation with the author. The portable and distributable version of FT-TOOL is still under development and it should be ready for distribution by Spring 1995.

A distributable beta version of the LP-MULTI will then be available upon request by anonymous ftp. Currently SunOS and Solaris versions (compiled with Gnu C ver. 2.6.3 and linkable with both Gnu C++ and SunPro++ ver. 4.0 compilers) and a version for MS-DOS for Borland C++ ver. 4.0 are being developed.

The distributable versions of the software will include also a Postscript file with the updated version of this Working Paper, which will continue to serve as a documentation of the software. Users are kindly requested to print this file and to make sure, that the version of the documentation corresponds to the version of the software.

LP-MULTI will be available free of charge for non-commercial research and educational purposes. Please contact the author (by e-mail: `marek@iiasa.ac.at`) for more information.