NOT FOR QUOTATION
WITHOUT THE PERMISSION
OF THE AUTHORS

THEORY, SOFTWARE AND TESTING EXAMPLES FOR DECISION SUPPORT SYSTEMS

A. Lewandowski
A. Wierzbicki

March 1987
WP-87-26

Working Papers are interim reports on work of the International Institute for Applied Systems Analysis and have received only limited review. Views or opinions expressed herein do not necessarily represent those of the Institute or of its National Member Organizations.

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

NOT FOR QUOTATION
WITHOUT THE PERMISSION
OF THE AUTHORS

THEORY, SOFTWARE AND TESTING EXAMPLES FOR DECISION SUPPORT SYSTEMS

A. Lewandowski
A. Wierzbicki

March 1987
WP-87-26

Working Papers are interim reports on work of the International Institute for Applied Systems Analysis and have received only limited review. Views or opinions expressed herein do not necessarily represent those of the Institute or of its National Member Organizations.

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

Foreword

Research in methodology of Decision Support Systems is one of the activities within the System and Decision Sciences Program which was initiated seven years ago and is still in the center of interests of SDS. During these years several methodological approaches and software tools have been developed; among others the DIDAS (Dynamic Interactive Decision Analysis and Support) and SCDAS (Selection Committed Decision Analysis and Support). Both methodologies gained a certain level of popularity and have been successfully applied in other IIASA programs and projects as well as in many scientific institutions.

Since development and testing the software and methodologies on real life examples requires certain - rather high - resources, it was decided to establish a rather extensive international collaboration with other scientific institutions in various NMO countries. This volume presents the result of the second phase of such a cooperation between the SDS Program and the four scientific institutions in Poland. The research performed during this stage related mostly to converting the decision support software developed during the previous phase, from the mainframe to the microcomputer, ensuring simultaneously high level of rebustness, efficiency and user friendliness. Several new theoretical developments, like new non-simplex algorithm for linear programming, new algorithms for mixed-integer programming and job shop scheduling are also described in the volume. Finally, it presents also new theoretical developments relating to supporting the processes of negotiations as well as the methodological issues on application the Decision Support Systems in industry management.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

CONTENTS

Introduction

Andrzej Lewandowski, Andrzej P. Wierzbicki

This collection of papers presents methodological reports for the contracted study agreement *'Theory, Software and Testing Examples for Decision Support Systems, Stage II'* between the International Institute for Applied Systems Analysis (IIASA), Systems and Decision Science Program, and the Polish Academy of Sciences, represented by four research institutes in Poland: the Institute of Automatic Control, Warsaw University of Technology (Part A and coordination on Polish side), the Institute of Systems Research, Polish Academy of Sciences (Part B), the Institute of Control and Systems Engineering, Academy of Mining and Metallurgy in Cracow (Part C) and the Institute of Informatics, Warsaw University (Part D). These methodological reports are augmented with more detailed manuals and software documentation in the form of separate working papers.

The papers present the results of research performed in 1986 according to the contracted study agreement, with slight modifications agreed upon in the course of research with Systems and Decision Sciences Program which coordinated the cooperation on IIASA side. Because of the need to summarize the long development of DIDAS family systems in response to many requests from various institutions collaborating with IIASA, it was agreed to prepare a comprehensive report *'Decision Support Systems of DIDAS family'* instead of reporting on further theoretical research in part A of the agreement; this theoretical research has been carried on, but will be reported in Stage III of the study. Some other minor corrections and specifications of the contracted study agreement has been agreed upon in the course of cooperation; on the whole, however, the papers presented here correspond to the scope of the study as specified in the contracted study agreement.

Therefore, the papers in this collection have diverse character, corresponding to various aspects of the theory, software and testing examples for decision support systems. All papers contained in this volume were presented at the international Task Force Meeting *'Theory, Software and Testing Examples for Decision Support Systems'*, organized upon IIASA request by the Institute of Automatic Control, Warsaw University of Technology, and the Institute of Systems Research, Polish Academy of Sciences, on December 8-9, 1986 in Warsaw. Since some of the papers are meant to be parts of self-standing software documentation, hence they might repeat, in their theoretical manuals, various explanations given in other papers of more theoretical character.

The papers in this volume are not ordered according to contracted study agreement, nor to the order of presenting them at the Warsaw Task Force Meeting; we have chosen instead an ordering corresponding to the subjects of theory, software and applications.

1) A special character has the first paper *'Decision Support Systems of DIDAS family'*, written by Andrzej Lewandowski, Tomasz Kreglewski, Tadeusz Rogowski and Andrzej Wierzbicki, which presents a comprehensive history, methodology, theory, implementation issues and various applications of systems related to the name Dynamic Interactive Decision Analysis and Support, based upon quasisatisficing rationality

framework and reference point optimization principles.

Next four papers have mostly theoretical character:

2) The paper *'Modern Techniques for Linear Dynamic and Stochastic Programs'*, by Andrzej Ruszczynski, presents a review of modern optimization techniques for structured linear programming problems, including non-simplex algorithm and, specifically, a new regularized decomposition method for stochastic optimization problems.

3) The paper *'Theoretical Guide NOA2: a FORTRAN Package of Nondifferentiable Optimization Algorithms'*, by Krzysztof Kiwiel and Andrzej Stachurski presents theoretical background for a package of FORTRAN subroutines of nondifferentiable optimization of locally Lipschitz continuous functions.

4) The paper *'Implicit Utility Function and Pairwise Comparisons'*, by Janusz Majchrzak presents an approach to estimating the utility function of decision maker for decision support systems that process discrete alternatives.

5) The paper *'Safety Principle in Multiobjective Decision Support in the Decision Space Defined by the Availability of Resources'* by Henryk Gorecki and A.Skulimowski presents new theoretical results on decision analysis with uncertainty about constraints in the criteria space and aspirations of the decision maker.

Further seven papers report on software development and are intended as parts of software documentation.

6) The paper *'Methodological Guide to HYBRID 3.01: a Mathematical Programming Package for Multicriteria Dynamic Linear Problems'*, by Marek Makowski and Janusz Sosnowski presents detailed methodological description of two versions of HYBRID systems of DIDAS family one for mainframe computers and one for IBM-PC compatibles.

7) The paper *'IAC-DIDAS-L, a Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Linear and Dynamic Linear Models on Professional Microcomputers'* written by Tadeusz Rogowski, Jerzy Sobczyk and Andrzej Wierzbicki, presents introductory documentation and theoretical manual for two new, professional microcomputer based, versions of systems of DIDAS family (one version in FORTRAN and one in PASCAL).

8) The paper, *'A Solver for the Transshipment Problem with Facility Location'*, by Wlodzimierz Ogryczak, Krzysztof Studzinski, and Krystian Zorychta, reports on the work in the Institute of Informatics, University of Warsaw. The paper describes a solver based on branch and bound technique with novel a implementation of simplex algorithm for specially ordered network problems.

9) The paper *'A Methodological Guide to the Decision Support System DISCRET for Discrete Alternatives Problems'*, by Janusz Majchrzak presents methodological description of the DISCRET decision support system.

10) The paper *'Nonlinear Model Generator'* by Jerzy Paczynski and Tomasz Kreglewski presents introductory documentation and theoretical manual for a nonlinear model generator for decision support systems in an easy to use spreadsheet format and with a symbolic differentiation package.

11) The paper *for Multicriteria Analysis of Nonlinear Models on Professional Microcomputers'*, by Tomasz Kreglewski, Jerzy Paczynski and Andrzej Wierzbicki, presents introductory documentation and theoretical manual for new version of nonlinear DIDAS system, including spreadsheet format model definition and symbolic model differentiation.

12) The paper *'Experimental System Supporting Multiobjective Bargaining Problem - a Methodological Guide'*, by Piotr Bronisz, Lech Krus and Bozena Lopuch presents a pilot

version of a interactive decision support system in multicriteria bargaining problem.

Finally, further four papers are related to applications or testing examples:

13) The paper *'A Permutative Scheduling Problem with Limited Resources'* by Tomasz Rys and Wieslaw Ziembla presents a specific testing example for decision support systems with discrete scheduling alternatives.

14) The paper *'Multiobjective Evaluation of Industrial Structures - MIDA application to the Case of Chemical Industry'*, by Maciej Zebrowski presents a methodological application of decision support systems.

15) The paper *'Spatial PDA Modelling for Industrial Development with Respect to Transportation Cost'* by Maciej Skocz and Wieslaw Ziembla presents a multiobjective decision problem related to the programming of the development of a spatially distributed industrial system.

16) The paper *'Technologies Ranking and Selection in Chemical Industry - an Application of SCDAS'*, by Grzegorz Dobrowolski and Maciej Zebrowski presents a specific application of the Selection Committee Decision Analysis and Support (SCDAS) System.

These reports present the results of a collaborative study in the stage II of the contracted study agreement that corresponds to the effort of circa 10 man-years, although over 20 researchers have been involved on part-time basis in this study and the results obtained through cooperation with independently funded projects in Poland are also partially included here.

Decision Support Systems of DIDAS Family (Dynamic Interactive Decision Analysis & Support)

Andrzej Lewandowski, Tomasz Kreglewski, Tadeusz Rogowski, Andrzej P. Wierzbicki

Institute of Automatic Control, Warsaw University of Technology

ABSTRACT

This paper presents a review of methodological principles, mathematical theory, variants of implementation and various applications of decision support systems of DIDAS family, developed by the authors and many other cooperating researchers during the years 1980-1986 in cooperation with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis. The purpose of such systems is to support generation and evaluation of alternative decisions in interaction with a decision maker that might change his preferences due to learning, while examining a substantive model of a decision situation prepared by experts and analysts. The systems of DIDAS family are based on the principle of reference point optimization and the quasisatisficing framework of rational choice.

Introduction

The results reported in this paper are an outcome of a long cooperation between the System and Decision Sciences Program of the International Institute for Applied Systems Analysis (IIASA) and the Institute of Automatic Control, Warsaw University of Technology as well as many other institutions in Poland and in other countries. This cooperation concentrated on applications of mathematical optimization techniques in multiobjective decision analysis and on the development of decision support systems. Although many articles in scientific journals and papers at international conferences described specific results obtained during this cooperation (in fact, four international workshops and several working meetings were organized during these cooperation), one of the main results - the family of Dynamic Interactive Decision Analysis and Support systems - has not been until now comprehensively described. Such a description is the purpose of this paper.

1. Concepts of decision support and frameworks for rational decisions.

1.1 Concepts of decision support systems.

The concept of a decision support system, though quite widely used and developed in contemporary research, is by no means well defined. Without attempting to give a restrictive definition (since such definition in an early stage of development might limit it too strongly), we can review main functions and various types of decision support.

The main function of such systems is to support decisions made by humans, in contrast to decision automation systems that replace humans in repetitive decisions because these are either too tedious or require very fast reaction time or very high precision. In this sense, every information processing system has some functions of decision support. However, modern decision support systems concentrate on and stress the functions of helping human decision makers in achieving better decisions, following the high tech - high touch trend in the development of modern societies [1]. We can list several types of systems that serve such purposes:

- *simple managerial support systems*, such as modern data bases, electronic spreadsheet systems, etc;
- *expert and knowledge base systems* whose main functions relate to the help in recognizing a pattern of decision situation; more advanced systems of this type might involve considerable use of artificial intelligence techniques;
- *alternative evaluation and generation systems* whose main functions concentrate on the processes of choice among various decision alternatives either specified a priori or generated with help of the system, including issues of planning, of collective decision processes and issues of negotiations between many decision makers; more advanced systems of this type might involve a considerable use of mathematical programming techniques, such as optimization, game theory, decision theory, dynamic systems theory etc.

Some authors [2] restrict the definition of decision support systems only to the third group while requiring that a decision support system should contain a model of decision support. Although the systems described in this paper belong precisely to this category, we would like to draw the attention of the reader that it is a narrow sense of interpreting decision support systems. With this reservation, we will concentrate on decision support systems in the narrow sense. These can be further subdivided along various attributes into many classes:

- systems that support *operational planning* of repetitive type versus systems that support *strategic planning*, confronting essentially novel decision situations;
- systems that concentrate on the choice between a number of *discrete alternatives* versus systems that admit a *continuum of alternatives* and help to generate interesting or favorable alternatives among this continuum;
- systems that are essentially designed to be used by a *single decision maker ("the user")* versus systems that are designed to help *many decision makers* simultaneously;
- *specialized systems* designed to help in a very specific decision situation versus adaptable *system shells* that can be adapted to specific cases in a broader class of decision situations;
- systems that use versus such that do not use explicitly mathematical programming techniques, such as optimization, in the generation or review of alternatives;
- systems that assume (explicitly or implicitly) a specific *framework of rationality* of decisions followed by the user versus systems that try to accommodate a broader class of perceptions of rationality [3].

This last distinction was an important issue in the development of decision support systems described in this paper.

1.2 Frameworks for rational decisions.

When trying to support a human decision maker by a computerized decision support system, we must try to understand first how human decisions are made and how to help in making rational decisions. However, the rationality concept followed by the designer of the system might not be followed by the user; good decision support systems must be thus flexible, should not impose too stringent definitions of rationality and must allow for many possible perceptions of rationality by the user.

The first distinction we should make is between the *calculative* or *analytical rationality* and the *deliberative* or *holistic rationality*, the *"hard"* approach and the *"soft"* approach. The most consistent argument for the "soft" or holistic approach was given by Dreyfus [4]. He argues - and supports this argument by experimental evidence - that a decision maker is a learning individual whose way of making decisions depends on the level of expertise attained through learning. A novice needs calculative rationality; an experienced decision maker uses calculative rationality in the background, while concentrating his attention on novel aspects of a decision situation. An expert does not need calculative rationality: in a known decision situation, he arrives at best decisions immediately, by absorbing and intuitively processing all pertinent information (presumably in a parallel processing scheme, but in a way that is unknown until now). A master expert, while subconsciously making best decisions, continuously searches for *"new angles"* - for new aspects or perspectives, motivated by the disturbing feeling that not everything is understood, the feeling that culminates and ends in the *"aha"* or *heureka effect* of perceiving a new perspective. Thus, the holistic approach can be understood as *the rationality of the culture of experts.*

However, even a master expert needs calculative decision support, either in order to simulate and learn about novel decision situations, or to fill in details of the decision in a repetitive situation; novice decision makers might need calculative decision support in order to learn and become experts. These needs must be taken into account when constructing decision support systems that incorporate many elements of calculative rationality.

There are several frameworks for calculative or analytical rationality; most of these, after deeper analysis, turn out to be culturally dependent [3]. The *utility maximization framework* has been long considered as expressing an universal rationality, as the basis of decision analysis; every other framework would be termed "not quite rational". The abstractive aspects of this framework are the most developed - see, e.g., [5], [6] - and a monograph of several volumes would be needed to summarize them. Without attempting to do so, three points should be stressed here. Firstly, utility maximization framework is not universal, is culturally dependent; it can be shown to express the *rationality of a small entrepreneur facing an infinite market* [3]. Secondly, its descriptive powers are rather limited; it is a good descriptive tool for representing mass economic behavior and a very poor tool for representing individual behavior. Thirdly, it is difficult to account for various levels of expertise and to support learning within this framework.

Many types of decision support systems attempt to approximate the utility function of the user and then to suggest a decision alternative that maximizes this utility function. Most users find such decision support systems not convenient: it takes many experiments and questions to the decision maker to approximate his utility and, when the user finally learns some new information from the support system, his utility might change and the entire process must be repeated. Moreover, many users resent too detailed questions about their utility or just refuse to think in terms of utility maximization. However, a good decision support system should also support users that think in terms of utility maximization. For this purpose, the following *principle of interactive reference point maximization and learning* can be applied.

Suppose the user is an expert that can intuitively, holistically maximize his unstated utility function; assume, however, that he has not full information about the available decision alternatives, their constraints and consequences, only some approximate mental model of them. By maximizing holistically his utility on this mental model, he can specify desirable consequences of the decision; we shall call these desirable consequences a *reference point* in the outcome or objective space. The function of a good decision support system should be then not to outguess the user about his utility function, but to take the reference point as a guideline and to use more detailed information about the decision alternatives, their constraints and consequences in order to provide the user with proposals of alternatives that came close to or are even better than the reference point.

This more detailed information must be included in the decision support system in the form of a *substantive model* of the decision situation, prepared beforehand by a group of analysts (in a sense, such a model constitutes a knowledge base for the system). Upon analysing the proposals generated in the system, the utility function of the user might remain constant or change due to learning, but he certainly will know more about available decision alternatives and their consequences. Thus, he is able to specify a new reference point and to continue interaction with the system. Once he has learned enough about available alternatives and their consequences, the interactive process stops at the maximum of his unstated utility function. If the user is not a master expert and might have difficulties with holistic optimization, the system should support him first in learning about decision alternatives, then in the optimization of his utility; but the latter is a secondary function of the system and can be performed also without explicit models of utility function while using the concept of reference points.

The concept of reference point optimization has been proposed by Wierzbicki [7], [8], [9]; following this concept, the principle of interactive reference point optimization and learning was first applied by Kallio, Lewandowski and Orchard-Hays [10] and then lead to the development of an entire family of decision support systems called DIDAS. However, before describing these systems in more detail, we must discuss shortly other frameworks of calculative rationality.

A concept similar or practically equivalent to the reference point is that of *aspiration levels* proposed over twenty years ago in the *satisficing rationality* framework by Simon [11], [12] and by many others that followed the behavioral criticism of the normative decision theory based on utility maximization. This framework started with the empirical observation that people do form adaptive aspiration levels by learning and use these aspirations to guide their decisions; very often, they cease to optimize upon reaching outcomes consistent with aspirations and thus make *satisficing decisions*. However, when building a rationale for such observed behavior, this framework postulated that people cannot maximize because of three reasons: the cost of computing optimal solutions in complex situations; the uncertainty of decision outcomes that makes most complex optimizations too difficult; and the complexity of decision situations in large industrial and administrative organizations that induces the decision makers to follow some well established *decision rules* that can be behaviorally observed and often coincide with satisficing decision making. This discussion whether and in what circumstances people could optimize substantiated the term *bounded rationality* (which implies misleadingly that this is somewhat less than full rationality) applied to the satisficing behavior and drown attention away from the essential points of learning and forming aspiration levels.

Meanwhile, two of the reasons for not optimizing quoted above have lost their relevance. The development of computers and computational methods of optimization, including stochastic optimization techniques, has considerably decreased the cost and increased the possibilities of calculative optimization; moreover, the empirical research on holistic rationality indicates that expert decision makers can easily determine best solutions in very complex situations even if they do not use calculative optimization. The third reason, supported by empirical observations, remains valid: the *satisficing rationality is typical for the culture of big industrial and administrative organizations* (see also [13]). However, it can today be differently interpreted: the appropriate question seems to be *not whether people could, but whether they should maximize.*

Any intelligent man, after some quarrels with his wife, learns that maximization is not always the best norm of behavior; children learn best from conflicts among themselves that cooperative behavior is socially desirable and that they must restrict natural tendencies to maximization in certain situations. In any non-trivial game with the number of participants less than infinity, a cooperative outcome is typically much better for all participants than an outcome resulting from individual maximization. This situation is called a *social trap* and motivated much research that recently gave results of paradigm-shifting importance [14], [15]: we can speak about a perspective of *evolutionary rationality*, where people develop - through social evolution - rules of cooperative behavior that involve foregoing short-term maximization of gains.

When trying to incorporate the lessons from the perspective of evolutionary rationality into decision support systems, another question must be raised: in which situations should we stop maximizing upon reaching aspiration levels? We should stop maximizing for good additional reasons, such as avoiding social traps or conflict escalation, but if these reasons are not incorporated into the substantive model of the decision situation, the question about foregoing maximization should be answered by the decision maker, not by the decision support system. This constitutes a drawback of many decision support systems based on goal programming techniques [16], [17] that impose on the user the unmodified satisficing rationality and stop optimization upon reaching given aspirations, called goals in this case.

When trying to modify goal programming techniques and strictly satisficing rationality to account for above considerations, the *principle of ideal organization* [18] can be applied in construction of decision support systems. This principle states that a good decision support system should be similar to an ideal organization consisting of a boss (the user of the system) and the staff (the system), where the boss specifies goals (aspirations, reference points) and the staff tries to work out detailed plans how to reach these goals. If the goals are not attainable, the staff should inform the boss about this fact, but also should propose a detailed plan how to approach these goals as close as it is possible. If this goals are just attainable and cannot be improved, the staff should propose a plan how to reach them, without trying to outguess the boss about his utility function and proposing plans that lead to different goals than stated by the boss.

If, however, the goals could be improved, the staff should inform the boss about this fact and propose a plan that leads to some uniform improvement of all goals specified by the boss; if the boss wishes that some goals should not be further improved, he can always instruct the staff accordingly by stating that, for some selected objectives, the goals correspond not to maximized (or minimized) but *stabilized* variables, that is, the staff should try to keep close to the goals for stabilized objectives without trying to exceed them. By specifying all objectives as stabilized, the boss imposes strictly satisficing behavior on the staff; but the responsibility for doing so remains with him, not with the staff.

The above principle of ideal organization can be easily combined with the principle of interactive reference point maximization and learning; jointly, they can be interpreted as a broader framework for rationality, called *quasisatisficing* framework [3], [19], that incorporates lessons from the holistic and the evolutionary rationality perspectives and can support decision makers adherence either to utility maximization or satisficing. In fact, the quasisatisficing framework can also support decision makers following other perspectives of rationality, such as the *program- and goal-oriented planning and management* framework. This framework, proposed by Glushkov [20] and Pospelov and Irikov [21], represents the culture of planning, but has been independently suggested later also by representatives of other cultures [22]. In this framework, rational action or program are obtained by specifying first primary objectives, called goals, and examining later how to shift constraints on secondary objectives, called means, in order to attain the goals. In distinction to the utility maximization or satisficing frameworks, the stress here is laid on the hierarchical arrangement of objectives; but the quasisatisficing framework can also handle hierarchical objectives.

2. Quasisatisficing and achievement functions.

The main concepts of the quasisatisficing framework, beside the principle of interactive reference point optimization and learning and the principle of ideal organization, are the use of reference points (aspiration levels, goals) as parameters by which the user specifies his requirements to the decision support system (controls the generation and selection of alternatives in the system) as well as the maximization of an order-consistent achievement function as the main mechanism by which the decision support system responds to the user requirements. Achievement functions have been used also in goal programming [17], however, without the requirement of order-consistency [19]. When following the principle of interactive reference point optimization and learning, an order-consistent achievement function can be interpreted as an ad hoc approximation of the utility function of the user [23]; if the user can holistically maximize his utility and interactively change reference points, there is no need for any more precise approximation of his utility function. When following the principle of ideal organization, an order-consistent achievement function can be interpreted as a proxy for utility or achievement function of the ideal staff (the decision support system) guided by aspirations specified by the boss (the user); this function is maximized in order to obtain best response to the requirements of the boss.

Based upon above principles and starting with the system described in [10], many decision support systems have been developed with the participation or cooperation of the authors of this paper [24], [25], [26], [27], [28], [29], [30], either in IIASA, or in several Polish institutions cooperating with IIASA. The name DIDAS (Dynamic Interactive Decision Analysis and Support) has been first used by Grauer, Lewandowski and Wierzbicki in [31]. Other systems based upon such principles are now being developed for implementations on professional microcomputers; all these systems we broadly call here "systems of DIDAS family". However, also other researchers adopted or developed parallely some principles of quasisatisficing framework, represented in the works of Nakayama and Sawaragi [32], Sakawa [33], Gorecki et al. [34], Steuer et al. [35], Strubegger [36], Messner [37], Korhonen et al. [38] and others; decision support systems of such type belong to a broader family using quasisatisficing principles of rationality.

Since the maximization of an order-consistent achievement function is a specific feature of systems of DIDAS family, we review here shortly the theory of such functions.

We consider first the basic case where the vector of decisions $x \in R^n$, the vector of objectives or outcomes of decisions $q \in R^p$, and the substantive model of decision situation has the form of a set of admissible decisions $X_0 \subset R^n$ - assumed to be compact - together with an outcome mapping, that is, a vector-valued objective function $f: X_0 \rightarrow R^p$ - assumed to be continuous, hence the set of attainable outcomes $Q_0 = f(X_0)$ be also compact; further modifications of this basic case will be considered later. If the decision maker wants to maximize all outcomes, then the partial ordering of the outcome space is implied by the *positive cone* $D = R^p_+$ - which means that the inequality $q' \geq q'' \Leftrightarrow q' - q'' \in D$ is understood in the sense of simple inequalities for each component of vectors q', q''.

However, the cone $D = R^p_+$ has nonempty interior; a more general case is when the decision maker would like to maximize only first p' outcomes, minimize next outcomes from $p'+1$ until p'', while the last outcomes from $p''+1$ until p are to be kept close to some given aspiration levels, that is, maximized below these levels and minimized above these levels; such objectives or outcomes are called *(softly) stabilized.* In this case, we redefine the positive cone to the form

$$D = \{q \in R^p:\ q_i \geq 0,\ i=1,..p';\ q_i \leq 0,\ i=p'+1,..p'';\ q_i = 0,\ i=p''+1,..p\} \tag{1}$$

This cone D does not have an interior if $p'' < p$. Since the cone D is closed and the set Q_0 is compact, there exist *D-efficient* (D-optimal) elements of Q_0 , see [18]. These are such elements $\hat{q} \in Q_0$ that $Q_0 \cap (\hat{q} + \tilde{D}) = 0$ where $\tilde{D} = D \setminus \{0\}$; if $p' = p$ and $D = R^p_+$, then D-efficient elements are called also Pareto-optimal (in other words - such that no outcome can be improved without deteriorating some other outcome). The corresponding decisions $\hat{x} \in X_0$ such that $\hat{q} = \hat{f}(x)$ are called D-efficient or Pareto-optimal as well. Although the decision maker is usually interested both in efficient decisions and outcomes, for theoretical considerations it is sufficient to analyse only the set of all D-efficient outcomes

$$\hat{Q}_0 = \{\hat{q} \in Q_0 : Q_0 \cap (\hat{q} + \tilde{D}) = 0\}, \quad \tilde{D} = D \setminus \{0\} \tag{2}$$

Several other concepts of efficiency are also important. The *weakly D-efficient* elements belong to the set

$$\hat{Q}_0^w = \{\hat{q} \in Q_0:\ Q_0 \cap (\hat{q} + intD) = \phi\} \tag{3}$$

In other words, these are such elements that cannot be improved in all outcomes jointly . Although important for theoretical considerations, weakly D-efficient elements are not useful in practical decision support, since there might be too many of them: if $p'' < p$ and the interior of D is empty, then all elements of Q_0 are weakly D-efficient. Another concept is that of ***properly D-efficient*** elements; these are such D-efficient elements that have bounded trade-off coefficients that indicate how much one of the objectives must be deteriorated in order to improve another one by a unit (for various almost equivalent definitions of such elements see [39]). In applications, it is more useful to further restrict the concept of proper efficiency and consider only such outcomes that have trade-off coefficients bounded by some a priori number. This corresponds to the concept of ***properly D-efficient elements with (a priori) bound ϵ*** or ***D_ϵ -efficient elements*** that belong to the set

$$\hat{Q}_0^\epsilon = \{\hat{q} \in Q_0:\ Q_0 \cap (\hat{q} + \tilde{D}_\epsilon) = 0\}, \tag{4}$$

$$\tilde{D}_\epsilon = \{q \in R^p : dist(q,D) \leq \epsilon\ ||q||\} \setminus \{\phi\}$$

where $\epsilon > 0$ is a given number [18]. D_ϵ -efficient elements have trade-off coefficients bounded approximately by ϵ and $1/\epsilon$. For computational and practical purposes, an efficient outcome with trade-off coefficients very close to zero or to infinity cannot be distinguished from weakly efficient outcomes; hence, we shall concentrate in the sequel on properly efficient elements with bound ϵ.

When trying to characterize mathematically various types of efficiency with help of achievement functions, two basic concepts are needed: this of *monotonicity*, essential for sufficient conditions of efficiency, and that of *separation of sets*, essential for necessary conditions of efficiency. The role of monotonicity in vector optimization is explained by the following basic theorem [19]:

Theorem 1. Let a function $r:Q_0 \rightarrow R^1$ be strongly monotone, that is, let $q' > q''$ (equivalent to $q' \in q'' + \tilde{D}$) imply $r(q') > r(q'')$. Then each maximal point of this function is efficient. Let this function be strictly monotone, that is, let $q' \gg q''$ (equivalent to $q' \in q'' + intD$) imply $r(q') > r(q'')$. Then each maximal point of this function is weakly efficient. Let this function be ϵ-strongly monotone, that is, let $q' \in q'' + \tilde{D}_\epsilon$ imply $r(q') > r(q'')$. Then each maximal point of this function is properly efficient with bound ϵ.

The second concept, that of separation of sets, is often used when deriving necessary conditions of scalar or vector optimality. We say that a function $r:R^p \rightarrow R^1$ *strongly separates two disjoint sets* Q_1 and Q_2 in R^p, if there is such $\beta \in R^1$ that $r(q) \leq \beta$ for all $q \in Q_1$ and $r(q) > \beta$ for all $q \in Q_2$. Since the definition of efficiency (2) requires that the sets Q_0 and $q + \tilde{D}$ are disjoint (similarly for the definitions (3) or (4)), they could be separated by a function. If Q_0 is convex, these sets can be separated by a linear function. If Q_0 is not convex, the sets Q_0 and $\hat{q} + \tilde{D}$ could be still separated at an efficient point $\hat{q}$, but we need for this a nonlinear function with level sets $\{q \in R^p \colon r(q) \geq \beta\}$ which would closely approximate the cone $\hat{q} + \tilde{D}$. There might be many such functions; their desirable properties are summarized in the definitions of *order-consistent achievement functions* [19] of two types: *order-representing functions* (which, however, characterize weak efficiency and will not be considered here) and *order- approximating functions*. The latter type is defined as follows:

Let A denote a subset of R^p, containing $\hat{Q}_0$ but not otherwise restricted, and let $\bar{q} \in A$ denote reference points or aspiration levels that might be attainable or not (we assume that the decision maker cannot a priori be certain whether $\bar{q} \in Q_0$ or $\bar{q} \notin Q_0$). Order-approximating achievement functions are such continuous functions $s:Q_0 \times A \rightarrow R^1$ that $s(q,\bar{q})$ is strongly monotone (see Theorem 1) as a function of $\bar{q} \in Q_0$ for any $q \in A$ and, moreover, possesses the following property of order approximation:

$$\bar{q} + D_{\bar{\epsilon}} \subset \{q \in R^p \colon s(q,\bar{q}) \geq 0\} \subset \bar{q} + D_\epsilon \tag{5}$$

with some small $\epsilon \geq \bar{\epsilon} \geq 0$; together with the continuity requirement, the requirement (5) implies that $s(q,\bar{q}) = 0$ for all $q = \bar{q}$.

If $p' = p$ and $D = R^p_+$, then a simple example of an order-approximating function is:

$$s(q,\bar{q}) = \min_{1 \leq i \leq p} \alpha_i (q_i - \bar{q}_i) + \alpha_{p+1} \sum_{i=1}^{p} a_i (q_i - \bar{q}_i) \tag{6}$$

with $A = R^p$, some positive weighting coefficients α_i (typically, we take $\alpha_i = 1/s_i$, where s_i are some scaling units for objectives, either defined by the user or determined automatically in the system, see further comments) and some $\alpha_{p+1} > 0$ that is sufficiently small as compared to ϵ and large as compared to $\bar{\epsilon}$ (typically, we take $\alpha_{p+1} = \epsilon/p$). This function is not only strongly monotone, but also $\bar{\epsilon}$-strongly monotone. For the more complicated form (1) of the positive cone D, function (6) modifies to:

$$s(q,\bar{q})=\min_{1\leq i\leq p} z_i(q_i,\bar{q}_i)+\alpha_{p+1}\sum_{i=1}^{p} z_i(q_i,\bar{q}_i) \tag{7}$$

where the functions $z_i(q_i,\bar{q}_i)$ are defined by:

$$z_i(q_i,\bar{q}_i)=\begin{cases}(q_i-\bar{q}_i)/s_i, & \text{if } 1\leq i\leq p', \\ (\bar{q}_i-q_i)/s_i, & \text{if } p'+1\leq i\leq p'', \\ min(z_i',z_i''), & \text{if } p''+1\leq i\leq p,\end{cases} \tag{8}$$

with

$$z_i'(q_i-\bar{q}_i)/s_i', \quad z_i''=(\bar{q}_i-q_i)/s_i'' \tag{9}$$

The coefficients s_i, s_i', s_i'' are scaling units for all objectives, either defined by the user (in which case $s_i'=s_i''$, the user does not need to define two scaling coefficients for a stabilized objective outcome) or determined automatically in the system; again, we use here $\alpha_{p+1}=\epsilon/p$.

Since the definition of an order-approximating achievement function requires that only its zero-level set should closely approximate the positive cone, many other forms of such functions are possible. For example, in some DIDAS systems the following function has been used:

$$s(q,\bar{q})=\min\left[\min_{i\leq i\leq p} z_i(q_i,\bar{q}_i),\ \frac{1}{\rho p}\sum_{i=1}^{p} z_i(q_i,\bar{q}_i)\right]+\frac{\epsilon}{p}\sum_{i=1}^{p} z_i(q_i,\bar{q}_i) \tag{10}$$

where the functions $z_i(q_i,\bar{q}_i)$ are defined as in (8), (9) and the coefficient $\rho\geq 1$ indicates to what extent the minimal overachievement is substituted by the sum of overachievements in the level sets for positive values of this function.

At any point $\hat{q}$ that is properly efficient with bound ϵ, an order-approximating function with $\bar{q}=\hat{q}$ strictly separates the sets $\hat{q}+\tilde{D}_\epsilon$ and Q_0. This and related properties of order-approximating functions result in the following characterization of D_ϵ -efficiency [19]:

Theorem 2. Let $s(q,\bar{q})$ be an order-approximating function with $\epsilon>\bar{\epsilon}\geq 0$. Then, for any $\bar{q}\in A$, each point that maximizes $s(q,\bar{q})$ over $q\in Q_0$ is efficient; if $\hat{q}$ is properly efficient with bound ϵ (D_ϵ-optimal), then the maximum of $s(q,\bar{q})$ with $\bar{q}=\hat{q}$ over $q\in Q_0$ is attained at $\hat{q}$ and is equal zero. Let, in addition, $s(q,\bar{q})$ be $\bar{\epsilon}$-strongly monotone with respect to q; then each point that maximizes $s(q,\bar{q})$ over $q\in Q_0$ is properly efficient with bound ϵ.

The essential difference between order-consistent achievement functions and other types of achievement functions, used in goal programming and based on norms, is that the aspiration or reference point $\bar{q}$ needs not to be unattainable in order to achieve efficiency; this is because order-consistent achievement functions remain monotone, even if the reference point crosses the efficient boundary of Q_0 . Somewhat simplifying, we can say that an order-consistent achievement function switches automatically from norm minimization to maximization when the aspiration point $\bar{q}$ crosses the efficient boundary and becomes attainable. On the other hand, the characterization by Theorem 2 is obtained without any convexity assumptions, because the order-approximating property of achievement functions results in a constructive though nonlinear separation of sets Q_0 and $\hat{q}+\tilde{D}$ even in nonconvex cases. In fact, the set Q_0 needs not to be even connected and the order-consistent achievement functions can be as well used to characterize solutions of multiobjective discrete or mixed programming. Theorem 2 is valid even if the decision outcomes are elements of infinite-dimensional complete normed (Banach) spaces, as in many cases of multiobjective dynamic trajectory optimization - see [18].

Order-approximating achievement functions have several interpretations. From the point of view of utility maximization, achievement function can be interpreted as an ad hoc approximation of the utility function of the user, based on the information that he conveyed to the decision support system: the partial preordering of the objective space (which objectives are to be maximized, which minimized and which stabilized) and the aspiration levels $\bar{q}$ for all objectives; if more information is already available, this ad hoc approximation can be improved - see further comments. The coefficient ϵ can be then interpreted as the weight that the user attaches to correcting the underachievement in the worst outcome by average overachievements in other outcomes. However, such an ad hoc approximation is not a classical utility function, since it is context-dependent: it explicitly depends on the aspiration levels $\bar{q}$ that summarize the experience of the user and change due to his learning during interaction, thus changing the approximation of the utility function. On the other hand, the achievement function (6) can have cardinal form: if $\alpha_i = 1/s_i$, then function (6) is independent on affine transformations of outcome space; the same applies to function (7).

When following the principle of an ideal organization, an order-approximating achievement function can be interpreted as the utility function of the staff that is aware of aspirations set by the boss; the maximum of the achievement function is then positive, if the staff can propose a solution that exceeds the aspiration levels, it is negative, if the staff cannot propose a solution that satisfies aspiration levels and only comes as closely as possible to them, and it is zero (Theorem 2) if the staff finds an efficient solution that produces outcomes strictly corresponding to the aspiration levels.

From the point of view of strictly satisficing rationality, one should take function (7) and set $p' = p'' = 0$, that is, let all outcomes be softly stabilized; this is actually done in goal programming approaches. From the point of view of program- and goal oriented planning, one should either assume that the primary objectives are constrained to be equal to their corresponding aspiration levels, thereby modifying the set of admissible decisions X_0 (such objectives or outcomes are called ***guided*** or ***strictly stabilized***), or assign much greater weights to primary objectives than to secondary objectives. We see that the quasisatisficing approach can be used by decision makers following either of these three frameworks of rationality.

Further mathematical properties of order-approximating achievement functions have been also investigated; for example, it can be shown that order-approximating functions give the strongest characterization of efficient solutions for cases where the set Q_0 is of an arbitrary, a priori unknown shape, which is a reasonable assumption in most applied cases [18]. Another important property of an order-approximating function of the form (6) or (7) is that its maximal point $\hat{q}$ depends Lipschitz-continuously on the aspiration point $\bar{q}$ in all cases when the maximum of this function is unique; thus, the user of the decision support system can continuously influence his selection of efficient outcomes by suitably modifying the aspiration or reference point.

Computationally, the maximization of an order-approximating achievement function is either simple - if Q_0 is a convex polyhedral set, then the problem of maximizing (6), (7) or (10) can be rewritten as a linear programming problem - or more complicated for nonlinear or nonconvex problems. In such cases, we must either represent (6), (7) or (10) by additional constraints, or apply nondifferentiable optimization techniques, since the definition of order-approximating achievement functions imply their nondifferentiability at $q=\bar{q}$. Often, it is advisable to use smooth order-approximating functions that give weaker necessary conditions of efficiency than in Theorem 2, but are better suited for computational applications - see further comments.

3. Phases of decision support in systems of DIDAS family.

A typical procedure of working with a system of DIDAS family consists of several phases:

A. The definition and edition of a substantive model of analysed process and decision situation by analyst(s);

B. The definition of the multiobjective decision problem using the substantive model, by the final user (the decision maker) together with analyst(s);

C. The initial analysis of the multiobjective decision problem, resulting in determining bounds on efficient outcomes and, possibly, a neutral efficient solution and outcome, by the user helped by the system;

D. The main phase of interactive, learning review of efficient solutions and outcomes for the multiobjective decision problem, by the user helped by the system;

E. An additional phase of sensitivity analysis (typically, helpful to the user) and/or convergence to the most preferred solution (typically, helpful only to users that adhere to utility maximization framework).

These phases have been implemented differently in various systems of DIDAS family; however, we describe them here comprehensively.

Phase A: Model definition and edition.

There are four basic classes of substantive models that have been used in various systems of DIDAS family: multiobjective linear programming models, multiobjective dynamic linear programming models, multiobjective nonlinear programming models and multiobjective dynamic nonlinear programming models. First DIDAS systems have not used any specific standards for these models; however, our accumulated experience has shown that such standards are useful and that they differ from typical theoretical formulations of such models (although they can be reformulated back to the typical theoretical form, but such reformulation should not bother the user).

A ***substantive model of multiobjective linear programming type*** consists of the specification of vectors of n decision variables $x \in R^n$ and of m outcome variables $y \in R^m$ together with linear model equations defining the relations between the decision variables and the outcome variables and with model bounds defining the lower and upper bounds for all decision and outcome variables:

$$y = Ax; \quad x^{lo} \le x \le x^{up}; \quad y^{lo} \le y \le y^{up} \tag{11}$$

where A is a $m \times n$ matrix of coefficients. Between outcome variables, some might be chosen as guided outcomes, corresponding to equality constraints; denote these variables by $y^c \in R^{m'} \subset R^m$ and the constraining value for them by b^c to write the additional constraints in the form:

$$y^c = A^c \ x = b^c; \quad y^{c,lo} \le b \le y^{c,up} \tag{12}$$

where A^c is the corresponding submatrix of A. Some other outcome variables can be chosen as optimized objectives or objective outcomes; actually, this is done in the phase B together with the specification whether they should be maximized, minimized or softly stabilized, but we present them here for the completeness of the model description. Some of the objective variables might be originally not represented as outcomes of the model, but we can always add them by modifying this model; in any case, the corresponding objective equations in linear models have the form:

$$q=Cx \tag{13}$$

where C is another submatrix of A. Thus, the set of attainable objective outcomes is $Q_0=CX_0$ and the set of admissible decisions X_0 is defined by:

$$X_0=\{x\in R^n\colon\ x^{lo}\le x\le x^{up};\ y^{lo}\le Ax\le y^{up};\ A^c x=b^c\} \tag{14}$$

By introducing proxy variables and constraints, the problem of maximizing functions (7) or (10) over outcomes (13) and admissible decisions (14) can be equivalently rewritten to a parametric linear programming problem, with the leading parameter $\bar{q}$; thus, in phases C, D, E, a linear programming algorithm called solver is applied. In initial versions of DIDAS systems for linear programming models, the typical MPS format for such models has been used when editing them in the computer; recent versions of DIDAS systems include also a user-friendly format of a spreadsheet.

A useful standard of defining a ***substantive model of multiobjective linear dynamic programming type*** is as follows. The model is defined on $T+1$ discrete time periods t, $0\le t\le T$. The decision variable x, called in this case *control trajectory*, is an entire sequence of decisions:

$$x = \{x[0],...x[t],...x[T-1]\}\in R^{nT},\ \ x[t]\in R^n \tag{15a}$$

and a special type of outcome variables, called ***state variables*** $w[t]\in R^{m'}$ is also considered. The entire sequence of ***state variables or state trajectory***:

$$w = \{w[0],...w[t],...w[T-1],w[T]\}\in R^{m*(T+1)} \tag{15b}$$

is actually one time period longer than x; the initial state $w[0]$ must be specified as given data. The fundamental equations of a substantive dynamic model have the form of ***state equations***:

$$w[t+1]=A[t]w[t]+B[t]x[t];\ \ t=0,..T-1,\ w[0]-\mathit{given} \tag{16a}$$

The model ***outcome equations*** have then the form:

$$y[t]=C[t]w[t]+D[t]x[t],\ \ t=0,..T-1; \tag{16b}$$

$$y[T]=C[T]w[T]\in R^{m''}$$

and define the sequence of outcome variables or ***outcome trajectory***:

$$y=\{y[0],...y[t],...y[T-1],y[T]\}\in R^{m''*(T+1)} \tag{15c}$$

The decision, state and outcome variables can all have their corresponding lower and upper bounds (each understood as an appropriate sequence of bounds):

$$x^{lo}\le x\le x^{up},\ \ w^{lo}\le w\le w^{up},\ \ y^{lo}\le y\le y^{up} \tag{16c}$$

The matrices *A[t], B[t], C[t[], D[t]* of appropriate dimensions can be dependent or independent on time t; in the latter case, the model is called ***time-invariant***. This distinction is important in multiobjective analysis of such models only in the sense of model edition: time-invariant models can be defined easier by automatic, repetitive edition of model equations and bounds for subsequent time periods.

Between the outcomes, some might be chosen to be equality constrained or guided along a given trajectory:

$$y^c[t]=e^c[t]\in R^{m''}\subset R^m,\ t=0,..T;\ \ e^c=\{e^c[0],...,e^c[t],...,e^c[T]\} \tag{17}$$

The optimized (maximized, minimized or stabilized) objective outcomes of such model can be actually selected in phase B among both state variables and outcome variables (or even decision variables) of this model; in any case, they form an entire *objective trajectory*:

$$q=\{q[0],...q[t],...q[T-1],q[T]\}\in R^{p*(T+1)},\ \ q[t]\in R^p \tag{18}$$

If we assume that the first components $q_i[t]$ for $1\leq i\leq p'$ are to be maximized, next for $p'+1\leq i\leq p''$ are to be minimized, last for $p''+1\leq i\leq p$ are to be stabilized (actually, the user in the phase B does not need to follow this order - he simply defines what to do with subsequent objectives), then the achievement function $s(q,\bar{q})$ - for example, originally given by (10) - in such a case takes the form:

$$s(q,\bar{q})=\min\left[\min_{0\leq t\leq T}\ \min_{0\leq i\leq p} z[t],\ \frac{1}{\rho(T+1)p}\sum_{t=0}^{T}\sum_{i=1}^{p} z_i[t]\right]+\frac{\epsilon}{(T+1)p}\sum_{t=0}^{T}\sum_{i=1}^{p} z_i[t] \tag{19}$$

where the functions $z[t]=z(q[t],\bar{q}[t])$ are defined by:

$$z_i[t]=\begin{cases}(q_i[t]-\bar{q}_i[t])/s_i[t], & \text{if } 1\leq i\leq p',\\ (\bar{q}_i[t]-q_i[t])/s_i[t], & \text{if } p'+1\leq i\leq p'',\\ \min z_i'[t],z_i''[t], & \text{if } p+1\leq i\leq p\end{cases} \tag{20}$$

where

$$z_i'[t]=(q_i[t]-\bar{q}_i[t])/s_i'[t],\ \ \ z_i''[t]=(\bar{q}_i[t]-q_i[t])/s_i''[t], \tag{21}$$

The user does not need to define time-varying scaling units $s_i[t]$ nor two different scaling units $s_i'[t],s_i''[t]$ for a stabilized objective: the time-dependence of scaling units and separate definitions of $s_i'[t],s_i''[t]$ are needed only in the case of automatic scaling in further phases.

A useful standard for a *substantive model of multiobjective nonlinear programming type* consists of the specification of vectors of n decision variables $x\in R^n$ and of m outcome variables $y\in R^m$ together with nonlinear model equations defining the relations between the decision variables and the outcome variables and with model bounds defining the lower and upper bounds for all decision and outcome variables:

$$y=g(x);\ \ x^{lo}\leq x\leq x^{up};\ \ y^{lo}\leq y\leq y^{up} \tag{22}$$

where $g{:}R^n\rightarrow R^m$ is a (differentiable) function. In fact, the user or the analyst does not have to define the function g explicitly; he can also define it recursively, that is, determine some further components of this vector-valued function as functions of formerly defined components. Between outcome variables, some might be chosen as guided outcomes corresponding to equality constraints; denote these variables by $y^c\in R^{n'}\subset R^m$ and the constraining value for them by b^c to write the additional constraints in the form:

$$y^c=g^c(x)=b^c;\ \ y^{c,lo}\leq b^c\leq y^{c,up} \tag{23}$$

where g^c is a function composed of corresponding components of g. In phase B, some other outcome variables can be also chosen as optimized objectives or objective outcomes. The corresponding objective equations have the form:

$$q=f(x) \tag{24}$$

where f is also composed of corresponding components of g. Thus, the set of attainable objective outcomes is $Q_0=f(X_0)$ where the set of admissible decisions X_0 is defined by:

$$X_0 = \{x \in R^n: \; x^{lo} \leq x \leq x^{up}; \; y^{lo} \leq g(x) \leq y^{up}; \; g^c(x)=b^c\} \tag{25}$$

In further phases of working with nonlinear models, an order-approximating achievement function must be maximized; for this purpose, a specially developed nonlinear optimization algorithm called *solver* is used. Since this maximization is performed repetitively, at least once for each interaction with the user that changes the parameter $\bar{q}$, there are special requirements for the solver that distinguish this algorithm from typical nonlinear optimization algorithms: it should be robust, adaptable and efficient, that is, it should compute reasonably fast an optimal solution for optimization problems of a broad class (for various differentiable functions g(x) and f(x)) without requiring from the user that he adjusts special parameters of the algorithm in order to obtain a solution. The experience in applying nonlinear optimization algorithms in decision support systems [26], [30] has led to the choice of an algorithm based on penalty shifting technique and projected conjugate gradient method. Since a penalty shifting technique anyway approximates nonlinear constraints by penalty terms, an appropriate form of an achievement function that differentiably approximates function (7) has been also developed and is actually used. This *smooth order-approximating achievement function* has the form:

$$s(q,\bar{q})=1-\left\{\frac{1}{p}\left[\sum_{i=1}^{p''}(w_i)^{\alpha}+\sum_{i=p''}^{p+1} max(w_i',w_i'')^{\alpha}\right]\right\}^{1/\alpha} \tag{26}$$

where w_i, w_i', w_i'' are functions of q_i, $\bar{q}_i$:

$$w_i(q_i,\bar{q}_i)=\begin{cases}(q_{i,max}-\bar{q}_i)/s_i, & \text{if } 1\leq i\leq p' \\ (q_i-q_{i,min})/s_i, & \text{if } p'+1\leq i\leq p''\end{cases} \tag{27a}$$

$$\left.\begin{array}{l} w_i'(q_i,\bar{q}_i)=(q_{i,max}-\bar{q}_i)/s_i' \\ w_i''(q_i,\bar{q}_i)=(\bar{q}_i-q_{i,min})/s_i'' \end{array}\right\}, \text{ if } p''+1\leq i\leq p, \tag{27b}$$

and the dependence on $\bar{q}_i$ results from a special definition of the scaling units that are determined by:

$$s_i=\begin{cases}(q_{i,max}-\bar{q}_i)/r_i, & \text{if } 1\leq i\leq p', \\ (\bar{q}_i-q_{i,min})/r_i, & \text{if } p'+1\leq i\leq p'',\end{cases} \tag{28a}$$

$$\left.\begin{array}{l} s_i'=(q_{i,max}-\bar{q}_i)/r_i \\ s_i''=(\bar{q}_i-q_{i,min})/r_i \end{array}\right\}, \text{ if } p''+1\leq i\leq p \tag{28b}$$

where r_i are additional weighting coefficients that might be defined by the user (however, the system does not need them and works also well if they are set by their default values $r_i=1$). In the initial analysis phase, the values $q_{i,max}$ and $q_{i,min}$ are set to the upper and lower bounds specified by the user for the corresponding outcome variables; later, they are modified, see further comments. The parameter $\alpha \geq 2$ is responsible for the approximation of the function (7) by the function (26): if $\alpha \rightarrow \infty$ and $\epsilon \rightarrow 0$, then these functions converge to each other (if $r_i=1$ and while taking into account the specific definition of scaling coefficients in (26-28)). However, the use of too large parameters results in badly conditioned problems when maximizing function (26), hence $\alpha=4 \cdots 8$ are suggested to be used.

The function (26) must be maximized with $q=f(x)$ over $x \in X_0$, while X_0 is determined by simple bounds $x^{lo} \leq x \leq x^{up}$ as well as by inequality constraints $y^{lo} \leq g(x) \leq y^{up}$ and equality constraints $g^c(x)=b^c$. In the shifted penalty technique, the following function is minimized instead:

$$
\begin{aligned}
P(x, \xi', \xi'', \xi, u', u'', v) = & -s(f(x),\bar{q})+ \\
& +\frac{1}{2}\sum_{i=1}^{p'} \xi_i'(max(0,g_i(x)-y_i^{up}+u_i'))^2+ \\
& +\frac{1}{2}\sum_{i=p'}^{p''+1} \xi_i''(max(0,y_i^{lo}-g_i(x)+u_i''))^2+ \\
& +\frac{1}{2}\sum_{i=p''}^{p+1} \xi\ (g_i^c(x)-b_i^c+v_i))^2
\end{aligned}
\tag{29}
$$

where ξ', ξ'', ξ are penalty coefficients and u', u'', v are penalty shifts. This function is minimized over x such that $x^{lo} \leq x \leq x^{up}$ while applying conjugate gradient directions, projected on these simple bounds if one of the bounds becomes active. When a minimum of this penalty function with given penalty coefficients and given penalty shifts (the latter are initially equal zero) is found, the violations of all outcome constraints are computed, the penalty shifts and coefficients are modified according to the shifted-increased penalty technique [40] and the penalty function is minimized again until the violations of outcome constraints are admissibly small. The results are then equivalent to the outcomes obtained by maximizing the achievement function (26) under all constraints. This technique is according to our experience one of the most robust nonlinear optimization methods.

We omit here the description of the useful standard for defining *substantive models of dynamic nonlinear programming type* that can be obtained by combining the previous cases.

Phase B. The definition of the multiobjective decision analysis problem.

For a given substantive model, the user can define various problems of multiobjective analysis by suitably choosing maximized, minimized, stabilized and guided outcomes. In this phase, he can also define which outcomes and decisions should be displayed to him additionally during interaction with the system (such additional variables are called *floating outcomes*). Since the substantive model is typically prepared by an analyst(s) in the phase A and further phases starting with the phase B must be performed by the final user, an essential aspect of all systems of DIDAS family is the user-friendliness of phase B and further phases; this issue has been variously resolved in consequent variants of DIDAS systems. In all these variants, however, the formulation of the achievement function and its optimization is prepared automatically by the system once phase B is completed.

Before the initial analysis phase, the user should also define some reasonable lower and upper bounds for each optimized (maximized, minimized or stabilized) variable, which results in an automatic definition of reasonable scaling units s_i for these variables. In further phases of analysis, these scaling units s_i can be further adjusted; this, however, requires an approximation of bounds on efficient solutions.

Phase C. Initial analysis of the multiobjective problem.

Once the multiobjective problem is defined, bounds on efficient solutions can be approximated either automatically or on request of the user.

The 'upper' bound for efficient solutions could be theoretically obtained through maximizing each objective separately (or minimizing, in case of minimized objectives; in the case of stabilized objectives, the user should know their entire attainable range, hence they should be both maximized and minimized). Jointly, the results of such optimization form a point that approximates from 'above' the set of efficient outcomes $\hat{Q}$, but this point almost never (except in degenerate cases) is in itself an attainable outcome; therefore, it is called the *utopia point*.

However, this way of computing the 'upper' bound for efficient outcomes is not always practical; many systems of DIDAS family use a different way of estimating the utopia point. This way consists in subsequent maximizations of the achievement function $s(q,\bar{q})$ with suitably selected reference points $\bar{q}$. If an objective should be maximized and its maximal value must be estimated, then the corresponding component of the reference point should be very high, while the components of this point for all other maximized objectives should be very low (for minimized objectives, they should be very high; stabilized objectives must be considered as floating in this case, that is, should not enter the achievement function). If an objective should be minimized and its minimal value must be estimated, the corresponding component of the reference point should be very low, while other components of this point are treated as in the previous case. If an objective should be stabilized and both its maximal and minimal values must be estimated, then the achievement function should be maximized twice, first time as if for a maximized objective and the second time as if for a minimized one. Thus, the entire number of optimization runs in utopia point computations is $p''+2(p-p'')$. This is especially important in dynamic cases, see further comments. It can be shown that this procedure gives a very good approximation of the utopia point $\hat{q}^{uto}$ in static cases, whereas the precise meaning of very high reference component should be interpreted as the upper bound for the objective minus, say, 0.1% of the distance between the lower and the upper bound, while the meaning of very low is the lower bound plus 0.1% of the distance between the upper and the lower bound.

During all these computations, the 'lower' bound for efficient outcomes can be also estimated, just by recording the lowest efficient outcomes that occur in subsequent optimizations for maximized objectives and the highest ones for minimized objectives (there is no need to record them for stabilized objectives, where the entire attainable range is anyway estimated). However, such a procedure results in the accurate, tight 'lower' bound for efficient outcomes - called ***nadir point*** $\hat{q}^{nad}$ - only if $p''=2$; for larger numbers of maximized and minimized objectives, this procedure can give misleading results, while an accurate computation of the nadir point becomes a very cumbersome computational task.

Therefore, some systems of DIDAS family offer an option of improving the estimation of the nadir point in such cases. This option consists in additional p'' maximization runs for achievement function $s(q,\bar{q})$ with reference points $\bar{q}$ that are very low, if the objective in question should be maximized, very high for other maximized objectives and very low for other minimized objectives, while stabilized objectives should be considered as floating; if the objective in question should be minimized, the corresponding reference component should be very high, while other reference components should be treated as in the previous case. By recording the lowest efficient outcomes that occur in subsequent optimizations for maximized objectives (and are lower than the previous estimation of nadir component) and the highest ones for minimized objectives (higher that the previous estimation of nadir component), a better estimation $\hat{q}^{nad}$ of the nadir point is obtained.

For dynamic models, the number of objectives becomes formally very high which would imply a very large number of optimization runs - $(p''+2(p-p''))*(T+1)$ - when estimating the utopia point; however, the user is confronted anyway with p objective trajectories which he can evaluate by 'Gestalt'. Therefore, it is important to obtain approximate bounds on entire trajectories. This can be obtained by $p''+2(p-p'')$ optimization runs organized as in the static case, with correspondingly 'very high' and 'very low' reference or aspiration trajectories.

Once the approximate bounds $\hat{q}^{uto}$ and $\hat{q}^{nad}$ are computed and known to the user, they can be utilized in various ways. One way consists in computing a neutral efficient solution, with outcomes situated approximately 'in the middle' of the efficient set. For this purpose, the reference point $\bar{q}$ is situated at the utopia point $\hat{q}^{uto}$ (only for maximized or minimized outcomes; for stabilized outcomes, the user-supplied reference component $\bar{q}_i$ must be included here) and the scaling units are determined by:

$$s_i = |\,\hat{q}_i^{uto} - \hat{q}_i^{nad}\,|, \quad 1 \le i \le p'' \tag{30a}$$

for maximized or minimized outcomes, and:

$$\left.\begin{aligned} s'_i &= \bar{q}_i - \hat{q}_i^{nad} + 0.01*(\hat{q}_i^{uto} - \hat{q}_i^{nad}) \\ s''_i &= \bar{q}_i^{uto} - \bar{q}_i + 0.01*(\hat{q}_i^{uto} - \hat{q}_i^{nad}) \end{aligned}\right\} \; p''+1 \le i \le p \tag{30b}$$

for stabilized outcomes, while the components of the utopia and the nadir points are interpreted respectively as the maximal and the minimal value of such an objective; the corrections by $0.01*(\hat{q}_i^{uto} - \hat{q}_i^{nad})$ ensures that the scaling coefficients remain positive, if the user selects the reference components for stabilized outcomes in the range $\hat{q}_i^{uto} \le \bar{q}_i \le \hat{q}_i^{nad}$ (if he does not, the system automatically projects the reference component on this range; the user-supplied weighting coefficients are automatically set to their default values $r_i = 1$ when computing a neutral efficient outcome). By maximizing the achievement function $s(q,\bar{q})$ with such data, the neutral efficient solution is obtained and can be utilized by the user as a starting point for further interactive analysis of efficient solutions.

Once the utopia and nadir point are estimated and, optionally, a neutral solution computed and communicated to the user, he has enough information about the ranges of outcomes in the problem to start the main interactive analysis phase.

Phase D. Interactive review of efficient solutions and outcomes.

In this phase, the user controls – by changing reference or aspiration points – the efficient solutions and outcomes computed for him in the system. It is assumed that the user is interested only in efficient solutions and outcomes; if he wants to analyse outcomes that are not efficient for the given definition of the problem, he must change this definition - for example, by putting more objectives in the stabilized or guided category - which, however, necessitates a repetition of phases B, C.

In the interactive analysis phase, an important consideration is that the user should be able to easily influence the selection of the efficient outcomes $\hat{q}$ by changing the reference point $\bar{q}$ in the maximized achievement function $s(q,\bar{q})$. It can be shown [19] that best suited for the purpose is the choice of scaling units determined by the difference between the slightly displaced utopia point and the current reference point:

$$s_i=\begin{cases}(\hat{q}_i^{uto}-\bar{q}_i+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}))/r_i, & \text{if } 1\leq i\leq p' \\ (\bar{q}_i-\hat{q}_i^{uto}+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}))/r_i, & \text{if } p'+1\leq i\leq p''\end{cases} \tag{31a}$$

for maximized or minimized outcomes. For stabilized outcomes, the scaling units are determined somewhat differently than in (30b):

$$\left.\begin{aligned} s_i^{\prime}&=(\hat{q}_i^{uto}-\bar{q}_i+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}))/r_i \\ s_i^{\prime}&=(\hat{q}_i^{uto}-\bar{q}_i+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}))/r_i\end{aligned}\right\}, \text{ if } p''+1\leq i\leq p \tag{31b}$$

It is assumed now that the user selects the reference components in the range $\hat{q}_i^{nad}\leq\bar{q}_i\leq\hat{q}_i^{uto}$ for maximized and stabilized outcomes or $\hat{q}_i^{uto}\leq\bar{q}_i\leq\hat{q}_i^{nad}$ for minimized outcomes (if he does not, the system automatically projects the reference component on these ranges). The weighting coefficients r_i might be used to further influence the selection of efficient outcomes, but the automatic definition of scaling units is sufficient for this purpose even if $r_i=1$ by default; thus, the user needs not be bothered by their definition. The interpretation of the above way of setting scaling units is that the user attaches implicitly more importance to reaching a reference component $\bar{q}_i$ if he places it close to the known utopia component; in such a case, the corresponding scaling unit becomes smaller and the corresponding objective component is weighted stronger in the achievement function $s(q,\bar{q})$. Thus, this way of *scaling relative to utopia-reference difference* is taking into account the implicit information given by the user in the relative position of the reference point. This way of scaling, used also in [32], [35], is implemented only in recent versions of systems of DIDAS family, especially in versions for nonlinear models.

When the relative scaling is applied, the user can easily obtain - by suitably moving reference points - efficient outcomes that are either situated close to the neutral solution, in the middle of efficient outcome set $\hat{Q}_0$, or in some remote parts of the set $\hat{Q}_0$, say, close to various extreme solutions. Typically, several experiments of computing such efficient outcomes give enough information for the user to select an actual decision - either some efficient decision suggested by the system, or even a different one, since even the best substantive model cannot encompass all aspects of a decision situation. However, there might be some cases in which the user would like to receive further support - either in analysing the sensitivity of a selected efficient outcome, or in converging to some best preferred solution and outcome.

Phase E. Sensitivity analysis and forced convergence.

For analysing the sensitivity of an efficient solution to changes in the proportions of outcomes, a ***multidimensional scan*** of efficient solutions is implemented in some systems of DIDAS family. This operation consists in selecting an efficient outcome, accepting it as a base $\bar{q}^{bas}$ for reference points, and performing p'' additional optimization runs with the reference points determined by:

$$\begin{aligned}&\bar{q}_j=\bar{q}_j^{bas}+\beta(\hat{q}_j^{uto}-\hat{q}_j^{nad}), \\ &\bar{q}_i=\bar{q}_i^{bas},\ i\neq j,\ 1\leq j\leq p'',\end{aligned} \tag{32}$$

where β is a coefficient determined by the user, $-1 \le \beta \le 1$; if the relative scaling is used and the reference components determined by (32) are outside the range $\hat{q}_j^{nad}$, $\hat{q}_j^{nad}$, they are projected automatically on this range. The reference components for stabilized outcomes are not perturbed in this operation (if the user wishes to perturb them, he might include them, say, in the maximized category). The efficient outcomes resulting from the maximization of the achievement function $s(q,\overline{q})$ with such perturbed reference points are typically also perturbed mostly along their subsequent components, although other their components might also change.

For analysing the sensitivity of an efficient solution when moving along a direction in the outcome space - and also as a help in converging to a most preferred solution - a *directional scan* of efficient outcomes can be implemented in systems of DIDAS family. This operation consists again in selecting an efficient outcome, accepting it as a base $\overline{q}^{bas}$ for reference points, selecting another reference point $\overline{q}$, and performing a user-specified number K of additional optimizations with reference points determined by:

$$\overline{q}(k)=\overline{q}^{bas}+\frac{k}{K}(\overline{q}-\overline{q}^{bas}), \quad 1 \le k \le K \tag{33}$$

The efficient solutions $\hat{q}(k)$ obtained through maximizing the achievement function $s(q,\overline{q}(k))$ with such reference points constitute a cut through the efficient set $\hat{Q}_0$ when moving approximately in the direction $\overline{q}-\overline{q}^{bas}$. If the user selects one of these efficient solutions, accepts as a new $\overline{q}^{bas}$ and performs next directional scans along some new directions of improvement, he can converge eventually to his most preferred solution - see [38]. Even if he does not wish the help in such convergence, directional scans can give him valuable information.

Another possible way of helping in convergence to the most preferred solution is choosing reference points as in (33) but using a harmonically decreasing sequence of coefficients (such as 1/j, where j is the iteration number) instead of user-selected coefficients k/K. This results in convergence even if the user makes stochastic errors in determining next directions of improvement of reference points, or even if he is not sure about his preferences and learns about them during this analysis, see [41]. Such a convergence - called here forced convergence - is rather slow and, after initial experiments, has not been yet implemented in systems of DIDAS family.

4. Review of various implementations of systems of DIDAS family.

There exist a number of various implementations of systems of DIDAS family. An early, prototype linear version was developed by Kalio, Lewandowski and Orchard-Hays [10]. This version utilized professional LP package SESAME available only on the IBM-370 mainframe computers, therefore it was not transferable. The user interface was rather poor and the usage of the system was limited to its authors and their collaborators.

The second, also linear, version of DIDAS family systems was developed by Lewandowski [42]. It was designed as pre- and postprocessor programs to a commercial LP package with standard MPSX input and output. Due to such design, it was easily transferable and many practical problems were solved using it on various computers. The main drawback of this system was that the interface between pre- and postprocessor and a the LP solver was based on reading and writing disk files, which was very time consuming for larger problems . An interaction with the user was very simple but inconvenient because of long time responses of the system transferring large amount of data.

The design goal of the next version of DIDAS was to eliminate, if possible, disk transfers and changes of data structures inside the system. It was done by Kreglewski and

Lewandowski [26] as a interactive multicriteria extension of MINOS linear programming system [44]; the reference point concepts were implemented accessing MINOS internal data structures. The user interface was redesigned and many new options added. However, the portability problems arose again: MINOS is not easily transferable.

The reference point approach was explored also by many others collaborating authors. A DIDAS/N system developed by Grauer and Kaden [43] was the first published nonlinear version of such a system. It was based on MINOS/Augmented [45] nonlinear programming system, an extended version of linear MINOS. Unfortunately, this solver is not robust and efficient enough for realistic nonlinear programming problems. Moreover, the user interface in the DIDAS/N system was rather complicated, hence applications of this system were rather limited. Later, Kaden and Kreglewski [30] developed another version of nonlinear DIDAS system. Earlier versions of DIDAS were also adapted for special purposes by Strubegger and Messner [36], [37].

Lewandowski and Kreglewski [46] developed another, general purpose nonlinear version of DIDAS system. It was based on a solver from Modular System for Nonlinear Programming [47] and written completely in FORTRAN, hence easily transferable to arbitrary computer. The user interface was reasonably simple, but preparation of data for the system was not quite straightforward.

The experiences of these developments led in 1985 to two new linear versions: DIDAS-MM and DIDAS-MZ. DIDAS-MM was a further development of the version with MINOS solver, with extended interactive features, special editor for dynamic linear models and graphic features. DIDAS-MZ is based on a linear programming solver from IMSL library which is widely accessible; therefore, DIDAS-MZ is much easier transferable.

In 1986, a new generation of DIDAS family systems was initiated, designed for work on IBM-PC-XT and compatible computers. These are: IAC-DIDAS-L1 and -L2 as well as IAC-DIDAS-N, described in other papers of this volume.

5. Applications of systems of DIDAS family.

The first implementation [10] of systems of DIDAS family was devoted to the application in forecasting and planning of the development of Finish forestry and forest industry sectors, based on a substantive model of linear dynamic type. Later, another version of DIDAS systems was applied [25] to planning of energy supply strategies, which led to other applications in the analysis of future energy- economy relations in Austria [36] and of future gas trade in Europe [37].

Parallely, applications to forecasting and planning agricultural production in Poland [29], to regional investment allocation in Hungary [49], to chemical industry planning [34] have been initiated. A special version of linear dynamic DIDAS was adapted to flood control problems [28]. A nonlinear version of DIDAS was first applied to issues of macroeconomic planning [48]; later applications of other nonlinear versions include problems of environmental protection of ground water quality [30].

Further applications of DIDAS family systems are reported in other papers in this volume.

References

[1] Naisbitt, J., Megatrends: Ten New Directions Transforming our Lives. Warner Books, New York, 1982.

[2] Van Hee, K., Operations research and artificial intelligence approaches to decision support systems. International Seminar: New Advances in Decision Support Systems, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1986.

[3] Wierzbicki, A.P., Negotiation and mediation in conflicts, II: Plural rationality and interactive decision processes. In M. Grauer, M. Thompson, A.P. Wierzbicki, editors: Plural Rationality and Interactive Decision Processes, Proceedings, Sopron 1984, Springer Verlag, Berlin.

[4] Dreyfus, S.E., Beyond rationality. In M. Grauer, M. Thompson, A.P. Wierzbicki, editors: Plural Rationality and Interactive Decision Processes, Proceedings, Sopron 1984, Springer Verlag, Berlin.

[5] Fishburn, P.C., Decision and Value Theory. Wiley, New York, 1964.

[6] Keeney, R.L. and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Trade-offs. Wiley, New York 1976.

[7] Wierzbicki, A.P., Penalty methods in solving optimization problems with vector performance criteria. VI Congress of IFAC, Boston 1975.

[8] Wierzbicki, A.P., Basic properties of scalarizing functionals for multiobjective optimization. Mathematische Operations- forschung und Statistik, Ser. Optimization 8, Nr 1, 1977.

[9] Wierzbicki, A.P., The use of reference objectives in multi- objective optimization. In G. Fandel and T. Gal, eds., Multiple Criteria Decision Making, Theory and Applications, Springer Verlag, Heidelberg 1980.

[10] Kallio, M., A. Lewandowski and W. Orchard-Hays, An implementation of the reference point approach for multi- objective optimization. WP-80-35, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1980.

[11] Simon, H.A., Models of Man. Macmillan, New York, 1957.

[12] Simon, H.A., Administrative Behavior. MacMillan, New York, 1958.

[13] Galbraith, J.K., The New Industrial State, Houghton-Mifflin, Boston, 1967.

[14] Rapoport, A., Uses of experimental games. In M. Grauer, M.Thompson and A.P. Wierzbicki, editors: Plural Rationality and Interactive Decision Analysis, Springer Verlag, Berlin, 1985.

[15] Axelrod, R., The Evolution of Cooperation. Basic Books, New York, 1985.

[16] Charnes and Cooper, Goal programming and multiple objective optimization, J. Oper. Res. Soc. 1, pp 39-54, 1975.

[17] Ignizio, J.P., Goal programming - a tool for multiobjective analysis. Journal for Operational Research, 29, pp 1109-1119, 1978.

[18] Wierzbicki, A.P., A mathematical basis for satisficing decision making. Mathematical Modelling 3, pp 391-405, 1982.

[19] Wierzbicki, A.P., On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR-Spektrum 8, pp 73-87, 1986.

[20] Glushkov, V.M., Basic principles of automation in organizational management systems (in Russian), Upravlayushcheye Sistemy i Mashiny, 1, 1972.

[21] Pospelov, G.S. and V.A.Irikov, Program- and Goal-Oriented Planning and Management (in Russian), Sovietskoye Radio, Moscow, 1976.

[22] Umpleby, S.A., A group process approach to organizational change. In H. Wedde, ed., Adequate Modelling of Systems, Springer-Verlag, Berlin, 1983.

[23] Lewandowski, A., S. Johnson and A.P. Wierzbicki, A Selection Committee Decision Support System: Implementation, Tutorial Example and Users Manual. International Institute for Applied Systems Analysis, Laxenburg, Austria, 1986; presented also at the MCDM Conference in Kyoto, Japan, August 1986.

[24] Lewandowski, A., and M. Grauer The reference point approach - methods of efficient implementation. WP-82-26, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1982.

[25] Grauer, M., A.Lewandowski and L. Schrattenholzer, Use of the reference level approach for the generation of efficient energy supply strategies. WP-82-19, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1982.

[26] Kreglewski, T. and A.Lewandowski: MM-MINOS - an integrated interactive decision support system. CP-83-63, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1983.

[27] Lewandowski, A., T. Rogowski and T. Kreglewski, A trajectory- oriented extension of DIDAS and its applications. In M. Grauer, M. Thompson, A.P. Wierzbicki, editors: Plural Rationality and Interactive Decision Processes, Proceedings, Sopron 1984, Springer Verlag, Berlin.

[28] Lewandowski, A., T. Rogowski and T. Kreglewski, Application of DIDAS methodology to flood control problems - numerical experiments. In M. Grauer, M. Thompson, A.P. Wierzbicki, editors: Plural Rationality and Interactive Decision Processes, Proceedings, Sopron 1984, Springer Verlag, Berlin.

[29] Makowski, M., and J. Sosnowski, A decision support system for planning and controlling agricultural production with a decentralized management structure. In M. Grauer, M. Thompson, A.P. Wierzbicki, editors: Plural Rationality and Interactive Decision Processes, Proceedings, Sopron 1984, Springer Verlag, Berlin.

[30] Kaden, S., and T. Kreglewski, Decision support system MINE - problem solver for nonlinear multi-criteria analysis. CP-86-5, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1986.

[31] Grauer, M., A. Lewandowski and A.P. Wierzbicki, DIDAS - theory, implementation and experience. In M. Grauer and A.P. Wierzbicki, editors: Interactive Decision Analysis, Springer Verlag, Berlin, 1983.

[32] Nakayama, H., and Y. Sawaragi, Satisficing trade-off method for multiobjective programming. In M. Grauer and A.P. Wierzbicki, editors: Interactive Decision Analysis, Springer Verlag, Berlin, 1983.

[33] Sakawa, M., Interactive fuzzy decision making for multi- objective nonlinear programming problems. In M. Grauer and A.P. Wierzbicki, editors: Interactive Decision Analysis, Springer Verlag, Berlin, 1983.

[34] Gorecki, H., J. Kopytowski, T. Rys and M. Zebrowski, A multiobjective procedure for project formulation - design of a chemical installation. In M. Grauer and A.P. Wierzbicki, editors: Interactive Decision Analysis, Springer Verlag, Berlin, 1983.

[35] Steuer, R. and E.V. Choo, An interactive weighted Chebyshev procedure for multiple objective programming. Mathematical Programming 26, pp 326-344, 1983.

[36] Strubegger, M., An approach for integrated energy-economy decision analysis: the case of Austria. In G. Fandel, M. Grauer, A. Kurzanski and A.P. Wierzbicki, eds., Large-Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach, Springer Verlag, Berlin, 1985.

[37] Messner, S., Natural gase trade in Europe and interactive decision analysis, In G. Fandel, M. Grauer, A. Kurzanski and A.P. Wierzbicki, eds., Large-Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach, Springer Verlag, Berlin, 1985.

[38] Korhonen, P. and J. Laakso, Solving a generalized goal programming approaches using a visual interactive approach. European Journal of Operational Research 26, pp 355-363, 1986.

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

[40] Wierzbicki, A.P., Models and Sensitivity of Control Systems, Elsevier, Amsterdam, 1984.

[41] Michalevich, M.V., Stochastic approaches to interactive multicriteria optimization problems, WP-86-10, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1986.

[42] Lewandowski, A., A Program Package for Linear Multiple Criteria Reference Point Optimization - Short User Manual, WP-82-80, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1982.

[43] Grauer,M. and S. Kaden, A Nonlinear Dynamic Interactive Decision Analysis and Support System (DIDAS/N) Users Guide, WP-84-23, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1984.

[44] Murtagh, B.A. and M.A. Saunders MINOS User's Guide, Technical Report SOL-77-9, Systems Optimization Laboratory, Stanford University, 1977.

[45] Murtagh, B.A. and M.A. Saunders MINOS/Augmented, Technical Report, SOl-80-14, Systems Optimization Laboratory, Stanford University, 1980.

[46] Lewandowski, A. and T. Kreglewski, A nonlinear version of DIDAS system, Collaborative volume: Theory, Software and Test Examples for Decision Support Systems, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1985.

[47] Kreglewski, T., T. Rogowski, A. Ruszczynski, J. Szymanowski, Optimization methods in FORTRAN, PWN, Warsaw, 1984 (in Polish).

[48] Grauer, M. and E.Zalai, A Reference Point Approach to Nonlinear Macroeconomic Planning, WP-82-134, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1982.

[49] Majchrzak, J., The implementation of the multicriteria reference point optimization approach to the Hungarian regional investment allocation model, WP-81-154, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1982.

Modern Techniques for Linear Dynamic and Stochastic Programs

Andrzej Ruszczynski

Institute of Automatic Control, Warsaw University of Technology

1.Introduction

In the last three decades the theory and computational methods of linear programming developed into a powerful tool for analysing linear models of economic planning and control. Modern linear programming packages (see, e.g., [17],[19]) are capable of solving problems with thousands of variables and constraints. Still, linear programming as the area of research is far from being closed. On the one hand, the practice poses new large and complex problems which result from the tendency to describe more and more complex objects of decision making by mathematical models. On the other hand, the trends in modern computer and information technology create a demand for user-friendly decision support systems with an intimate interaction between the decision maker and the computer. The computer is often just a personal computer and this implies very specific requirements from the optimization software involved in such systems: it should be capable of solving large models, fast, use computer resources in an economic way, and it should allow for easy changes in the model.

A detailed discussion of all these issues goes far beyond the scope of this paper. We shall focus our attention here on two main sources of large scale linear models arising in decision making: dynamic structure and stochasticity. We shall discuss the ways in which general linear programming techniques can be specialized for these models to meet some of the computational goals pointed out above. Next, we shall present two nonstandard techniques which appear to be particularly useful for the problems in question.

2. Dynamic structure and stochasticity as sources of large linear models

It is well known that every linear optimization problem can be equivalently expressed in the following form

$$\textit{minimize} \quad c^T x$$

subject to

$$Ax=b, \qquad (2.1)$$
$$x^{min} \le x \le x^{max},$$

where x is the vector of activities (including slack/surplus variables), c is a vector of cost coefficients associated with these activities, A is a technology matrix, and b is a vector of resources or demands, which impose conditions on the admissible activities x. In real-life large scale models, the dimension of x (the number of columns of A) and the dimension of b (the number of rows of A) may go into thousands. On the other hand, it is typical that each resource or demand condition (a row of $Ax = b$) involves only few activities and each activity appears in only a relatively small number of conditions. As a result, the constraint matrix A in (2.1) is usually sparse: most of its entries are zeros. Its density (the proportion of the number of nonzeros to the size) may be less than 1% and it is clear that this feature should be exploited by the methods for solving (2.1). In fact, all modern linear programming codes make use of this feature and contain very sophisticated techniques for storing and factorizing sparse matrices, solving equations with them, and updating the factorization when the data change (see [5],[23]).

However, there exist important classes of problems in which sparsity alone is not the only feature of the constraint matrix. One of these classes are *linear dynamic-structured problems*, in other words - *linear control problems*. In the simplest formulation of such a problem our variables (activities) are related to time stages $t=0,1,2,..,T$. At each stage t, we deal with two groups of variables: *state variables* s_i and *control variables* u_t. The variables from the neighboring periods are related through the *state equation*

$$s_{t+1}=Gs_t+Ku_t+b_t,\quad t=0,1,2,..,T-1, \tag{2.2}$$

where G and K are some matrices of appropriate dimensions and b_t are some known vectors. Let the initial state s_0 be fixed and let us write our linear objective function as

$$f(u,s)=\sum_{t=0}^{T-1}(q_t^T u_t+c_{t+1}^T s_{t+1}). \tag{2.3}$$

Assuming that the only additional constraints on the state and control variables are simple lower and upper bounds

$$s_t^{min}\leq s_t\leq s_t^{max},\quad t=1,2,..,T, \tag{2.4}$$

$$u_t^{min}\leq u_t\leq u_t^{max},\quad t=1,2,..,T-1, \tag{2.5}$$

we can easily write our problem in form (2.1) with

$$\begin{aligned} x&=(u_0,s_1,u_1,s_2,..,u_{T-1},s_T),\\ c&=(q_0,c_1,q_1,c_2,..,q_{T-1},c_T),\\ b&=(b_0,b_1,..,b_{T-1}),\end{aligned} \tag{2.6}$$

and

$$a=\begin{bmatrix} K & I & & & & & & & \\ & -G & -K & I & & & & & \\ & & & -G & -K & I & & & \\ & & & & & & \cdots & & \\ & & & & & & -G & -K & I \end{bmatrix} \tag{2.7}$$

We see that the number of rows and columns of A increase proportionally to to the number of periods T, and even for relatively small dimensions of the activities related to a single period the whole problem may have a remarkable size. On the other hand, the matrix (2.7) is not only sparse, but has a very regular *staircase* structure with multiple occurrence of the same (usually also sparse) matrices G, K and I. We have to take advantage of it if we aim at solving dynamic problems of realistic dimensions.

Let us now pass on to the second class of problems which are of special interest for us. Let us assume that some of the entries of the technology matrix A and the right-hand side b in the linear model (2.1) are uncertain and that this uncertainty is crucial for the decision making. One of possible modelling approaches to such a situation (see, e.g., [12]) is to assume that A and b are random and may attain one of finite many realizations with some known probabilities:

$$(A_1,b_1)\quad \textit{with probability}\quad p_1>0, \tag{2.8}$$

$$(A_2,b_2)\quad \textit{with probability}\quad p_2>0,$$

$\cdots$

(A_L,b_L) *with probability* $p_L>0,$

where $\sum_{l=1}^{L} p_l=1$. Under these circumstances, however, it is in general no longer possible that the decision x satisfies the constraints $A_lx=b_l$ for all realizations $l=1,2,..,L$. Therefore, we have to extend our model by introducing some corrective activities y_l associated with the realizations $l=1,2,..,L$, which compensate the discrepancy b_l-A_lx. If we describe our capabilities of correction by a matrix W and assign to y_l the cost vector q and the bounds y^{min} and y^{max}, the correction problem will take the form

$$\text{minimize } q^Ty$$

subject to

$$Wy_l=b_l-A_lx, \quad y^{min}\le y\le y^{max}. \tag{2.9}$$

Our aim is now to find such a decision x that makes the correction always possible and minimizes the sum of the direct cost c^Tx and the expected future correction cost $\sum_{l=1}^{L} p_lq^Ty_l$. The whole problem can be again written as a large scale linear model:

$$\text{minimize } c^Tx+p_1q^Ty_1+p_2q^Ty_2+\cdots+p_Lq^Ty_L$$

subject to

$$\begin{array}{llllll} A_1x & +W_1y_1 & & & & =b_1 \\ A_2x & & +W_2y_2 & & & =b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ A_Lx & & & & W_Ly_L & =b_L \\ \multicolumn{6}{l}{x^{min}\le x\le x^{max}} \\ \multicolumn{6}{l}{y^{min}\le y_l\le y^{max}, l=1,2,..,L} \end{array} \tag{2.10}$$

The constraint matrix of (2.10),

$$A=\begin{bmatrix} A_1 & W & & \\ A_2 & & W & \\ \cdots & \cdots & \cdots & \cdots \\ A_L & & & W \end{bmatrix} \tag{2.11}$$

has the size proportional to the number L of realizations taken into account, which leads to very large problems already for underlying deterministic models of medium size. Still, similarly to the dynamic case, A is not only sparse but has a very regular (so-called *dual angular*) structure, with multiple occurrence of the correction matrix W and some similarities of the realizations $A_1,A_2,..,A_L$. It is intuitively clear that we have to take advantage of that in the method for solving such problems.

3. Specialized versions of the simplex method

When dealing with special classes of problems for which general efficient techniques already exist, it is a natural direction of research to investigate the possibility of exploiting the features of these special problems within the general approach. So, we shall discuss here some most promising specializations of the acknowledged method of linear programming, the *primal simplex method*, for the two classes in question: dynamic and stochastic problems.

In the primal simplex method the constraint matrix A in (2.1) is split into a square nonsingular *basis matrix* B and a matrix N containing all the remaining columns of A, not included into B. This implies division of the activities x into *basic variables* x_B and *nonbasic variables* x_N . At each iteration of the method the nonbasic variables are fixed on their lower or upper bounds, and the values of the basic variables are given by

$$x_B = B^{-1}(b - Nx_N). \tag{3.1}$$

We always choose basis matrices B so that

$$x_B^{min} \le x_B \le x_B^{min}, \tag{3.2}$$

where x_B^{min} and x_B^{max} are subvectors of $x^{\min}$ and x^{max} implied by the division of x into x_B and x_N . Such an x is called a *basic feasible solution*, and at each iteration we try to find a better basic feasible solution by performing the following steps.

Step 1. Find the price vector p by solving

$$\pi^T B = c^{T_B}, \tag{3.3}$$

where c_B is the subvector of c associated with x_B .

Step 2. Price out the nonbasic columns a_j of A (i.e. columns of N) by calculating

$$z_j = c_j - \pi^T a_j \tag{3.4}$$

until a column a_s is found for which $z_s < 0$ and $x_s = x^{\min}$, or $z_s > 0$ and $x_s = x^{max_s}$.

Step 3. Find the direction of changes of basic variables d_B by solving

$$Bd_B = a_s. \tag{3.5}$$

Step 4. Determine from $x_B^{min}, x_B^{max}, x_B$ and $d_B$the basic variable x_{Br} which first achieves its bound when x_s changes.

Step 5. Replace the r-th column of B with a_s,x_{Br} with x_s and calculate values of the new basic variables from (3.1).

This general strategy can be deeply specialized to account for the features of problems under consideration. These improvements can be divided into three groups:

a) representation of the problem data, i.e. the way in which the matrix A is stored and its columns a_j recovered for the purpose of Step 2;

b) techniques for solving equations (3.1), (3.3) and (3.5), which includes special methods for factorizing the basis matrix B and updating this factorization;

c) pricing strategies, i.e. methods for selecting nonbasic columns a_j at Step 2 to be priced out for testing whether they could be included into B at the current iteration.

Let us discuss these issues in more detail.

Problem data structures

The repeated occurrence of the matrices G, K and I in the constraint matrix (2.7) of the dynamic model suggests a generalization of the concept of *supersparsity* employed in large linear programming systems [1]. It is sufficient to store the matrices G and K as files of packed columns (G and K may be sparse themselves). Any time a specific column a_j of A is needed, we can easily calculate from its number j and from the dimensions of activities related to a single period which column of $-K$ or of $\begin{bmatrix} I \\ -G \end{bmatrix}$ and on which position will appear in a_j . Thus the problem data can be compressed in this case to the size of one period and easily stored in the operating memory of the computer, even for very large problems. In a nonstationary problem, where some of the entries of K and G depend on t, we can still store in this way all the stationary data, and keep an additional file of time-dependent entries. The recovery of a column of A would then be slightly more complicated, with a correction to account for the nonstationary entries, but still relatively easy to accomplish. Storage savings would be still significant, because we have grounds to expect that only some entries of A change in time.

The same argument applies to the constraint matrix (2.11) of the stochastic problem. It is sufficient to store the realizations $A_1,A_2,..,A_L$ and W to reconstruct columns of A, if necessary. But we can go here a little deeper, noting that in practical problems it is unlikely that all the entries of the technology matrix are random. If only some of them are stochastic, many entries of $A_1,A_2,..,A_L$ will have identical values and our problem data structure will still suffer from a considerable redundancy. Thus, we can further compress the structure, as it was done in [16]: we represent each A as

$$A_l = A^0 + \Delta_l$$

where A^0 contains as nonzeros only the deterministic entries of A_l, and Δ_l contains as only nonzeros the l-th realization of the random entries. Therefore it is sufficient to store the nonzeros of A^0 together with its sparsity pattern, the sparsity pattern of the random entries (which is common for all Δ_l), and the nonzeros of Δ_l, $l=1,2,..,L$. This structure will only slightly exceed the storage requirements of the underlying deterministic model.

Representation of the basis inverse

It is clear that for constraint matrices of the form (2.7) or (2.11) the basis matrices B inherit their structure. Although general techniques for factorizing sparse matrices (see, e.g., [5],[23],[28]) are in principle able to cope with such bases, there is still room to exploit their structure within the factorization and updating algorithms.

Let us at first discuss this matter on the simple control problem with the constraint matrix (2.7). Assuming that all the state vectors $s_1,s_2,..,s_T$ are basic, we obtain the following form of the basis matrix

$$B_0 = \begin{bmatrix} I & & & & \\ -G & I & & & \\ & -G & I & & \\ \dots & \dots & \dots\dots & \dots & \dots \\ & & & I & \\ & & & -G & I \end{bmatrix} \tag{3.6}$$

B_0 is lower triangular and the equations involving B_0 or B_0^T can be simply solved by substitution. To solve $B_0 d=a$, we partition d into $(d_1,d_2,..,d_T)$ and a into $(a_0,a_1,..,a_{T-1})$ according to the periods, and solve the state equations

$$d_{t+1}=Gd_t+a_t, \quad t=0,1,..,T-1 \tag{3.7}$$

with $d_0=0$. Noting that in (3.4) we have $a_t=0$ for $t<\tau$ we can start simulation in (3.7) from τ. To solve $\pi^T B_0=c$ we need only to back-substitute in the *adjoint equations*

$$\pi^T=G^T\pi_{t+1}+c_t, \quad t=T,T-1,..,1 \tag{3.8}$$

with $\pi_{t+1}=0$. Again, noting that c_B in (3.3) changes only on one position from iteration to iteration, we can start the simulation in (3.8) from the position at which the change occurred.

In general, the basis matrix is not so simple as (3.6) and some controls are basic, while some state variables are nonbasic. The basis matrix is still staircase, but the blocks on the diagonal (which in (3.6) are all I) are not necessarily square and invertible:

$$B_0 = \begin{bmatrix} -K_1 & J_1 & & & & & & \\ & -G_1 & -K_2 & J_2 & & & & \\ & & & -G_2 & & & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & & -G_{T-1} & -K_T & J_T \end{bmatrix} \tag{3.9}$$

where $J_1,J_2,..,J_T$ are some submatrices of I; $K_1,K_2,..,K_T$ are submatrices of K and $G_1,G_2,..,G_{T-1}$ are submatrices of G. A factorization of B is necessary to represent it in a form suitable for solving equations with B and B^T and for corrections when a column of B is exchanged.

We can of course specialize the elimination procedures of [5] or [13], because we exactly know where to look for nonzeros in particular rows and columns of B. This idea of blockwise elimination has been analysed in [14], [22] and [32]. There is, however, a more promising global approach which aims at exploiting features similar to those that led from (3.6) to the equations (3.7) and (3.8). Namely, we would like to transform somehow B to a staircase matrix

$$\tilde{B} = \begin{bmatrix} \tilde{B}_{11} & & & & & \\ \tilde{B}_{21} & \tilde{B}_{22} & & & & \\ & \tilde{B}_{32} & \tilde{B}_{33} & & & \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & \tilde{B}_{T-1,T} & \tilde{B}_{T,T} \end{bmatrix} \tag{3.10}$$

having the diagonal blocks $\tilde{B}_{tt}$ square and nonsingular. Solving equations with $\tilde{B}$ would be almost as simple as with B_0 and would require only inversion of $\tilde{B}_{tt}$, $t=1,2,..,T$.

In [20] the pass from B to $\tilde{B}$ is achieved by representing

$$B=\tilde{B}F \tag{3.11}$$

with F chosen in such a way that $\tilde{B}$ inherits as many columns of B as possible. In particular, all the state columns of B will appear in $\tilde{B}$, so that the diagonal blocks $\tilde{B}_{tt}$ will have large parts common with the identity and will be easy to invert. Moreover, F has also a very special structure

$$F=\begin{bmatrix} D & \\ E & I \end{bmatrix} \tag{3.12}$$

with D square, invertible, and of relatively low size. Solving the equations with B or B^T resolves now itself to the factorization of $\tilde{B}_{tt}$ (which is easy) and factorization of D (see [20]). Updating the factors is rather involved, unfortunately.

Another approach has been suggested in [1]. Since B_0 is particularly easy to invert, we aim at using B_0 as $\tilde{B}$. We do not construct factors as in (3.11) but rather add new rows and columns to B_0 and work with a larger matrix

$$\hat{B} = \begin{bmatrix} B_0 & U \\ V & \end{bmatrix} \tag{3.13}$$

Here U contains columns which are in B but not in B_0, and V contains units in columns which are in B_0 but not in B, to explicitly nullify the variables corresponding to these columns. The solution to

$$B\begin{bmatrix} s_B \\ u_B \end{bmatrix} = a \tag{3.14}$$

can be now computed by

$$u_B = (VB_0^{-1}U)^{-1}VB_0^{-1}a, \tag{3.15}$$

$$s = B_0^{-1}(a - Uu_B). \tag{3.16}$$

Thus we need only to solve equations with B_0 , which is particularly simple, and to factorize the matrix $VB_0^{-1}U$, which is of much smaller size than B. Similar formulae can be derived for the backward transformation (3.3). Updating the factors is much more simple than for (3.11),(3.12), because the general form (3.13) does not change when rows of V and columns of U are added or deleted.

Let us now pass to the stochastic problem (2.10). Supposing that the basis contains only the correction activities, its form is particularly simple

$$B_0 = \begin{bmatrix} W_1 & & & \\ & W_2 & & \\ & & \cdots & \\ & & & W_L \end{bmatrix} \tag{3.17}$$

where W_l , l=1,2,..,L are square nonsingular submatrices of W. The inversion of B_0 resolves now itself to the inversion of $W_1, W_2, \ldots, W_l$, which can be done independently. We can also exploit here some similarities between the W's (common columns) to further simplify their inversion (see the bunching procedure discussed for other purposes in [32]).

In general, however, the basis matrix will be of the form

$$B = \begin{bmatrix} \tilde{A}_1 & \tilde{W}_1 & & \\ \tilde{A}_2 & & \tilde{W}_2 & \\ \cdots & \cdots & \cdots & \cdots \\ \tilde{A}_L & & & \tilde{W}_L \end{bmatrix} \tag{3.18}$$

with the blocks $\tilde{W}_l$ $l=1,2,..,L$, not necessarily square and nonsingular. Again, we would like to transform B into a form more suitable for inversion. At the first sight, since B is lower block triangular, both approaches discussed for the dynamic problem are applicable here. We can aim at obtaining factors as in (3.11) with a $\tilde{W}$ of dual angular structure having invertible diagonal blocks. We can also apply a method based on the Sherman-Morrison formulae (3.15)-(3.16) and work with a matrix of the form (3.13).

The relation with the dynamic model, however, follows from rather superficial algebraic similarity of the problem matrices (lower block triangular structure). In fact, in the dynamic model we deal with a phenomenon that evolves in time, whereas the stochastic model describes a phenomenon spread in space. Thus, while we had grounds to assume that many state variables will be basic in the dynamic model (which implied the choice of B_0), we cannot claim the same with respect to the correction activities in the stochastic model and specify in advance some of them to be included into W. Therefore, the approach of [1] must be slightly modified here. Instead of working with B, we would prefer to operate on a larger matrix

$$\hat{B} = \begin{bmatrix} W & & & & \tilde{A}_1 \\ J_1 & & & & \\ & W & & & \tilde{A}_2 \\ & J_2 & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & W & \tilde{A}_L \\ & & & J_L & \\ V_1 & & & & \\ & V_2 & & & \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ & & & V_L & \end{bmatrix} \tag{3.19}$$

in which some of the rows of the matrix V, which are used to nullify the nonbasic correction activities, are added to W to make the diagonal blocks $\begin{bmatrix} W \\ J \end{bmatrix}$ square and invertible. Under these circumstances, however, the block diagonal part of $\hat{B}$ is no longer constant, contrary to the matrix B_0 in the form (3.13) for dynamic problems. The representation (3.19) and the resulting updating schemes were analysed in the dual (transposed) form in [12], and [27]. The resulting formulae, however, are so involved and distant from the essence of the underlying problem, that it is not clear whether this particular direction can bring a significant progress.

The approach (3.11) might be more prospective here, but we should be aware of the fact that it is natural to expect that many first stage activities x will be basic, because corrections are usually more expensive. Hence, the blocks $\tilde{W}_l$ in (3.18) will be far from square and adding to them columns to achieve the block diagonal $\tilde{B}$ will inevitably increase the size of D in (3.12).

Summing up this part of our discussion, we can conclude that implementations of the simplex method for large dynamic and stochastic problems lead to very detailed linear algebraic techniques that try to exploit the structure of basis matrices to develop improved inversion methods. Although there is still a lot to be done in this direction, one can hardly expect a qualitative progress here.

Pricing strategies

Let us now pass to the problem of selecting nonbasic columns to be priced out at a given iteration for testing whether they could be brought into the basis. Since the selection of a variable to enter the basis largely determines the variable to leave, pricing strategies have a considerable influence on iteration paths of the simplex method and this influence grows with the size of the problem. There are two acknowledged techniques for general large scale linear programs (cf., e.g., [18]):

a) *partial pricing*, where at each iteration a certain subset of nonbasic columns are priced out to select the one to enter;

b) *multiple pricing*, where a list of prospective candidates is stored, and they are priced out again at the next iteration.

These general ideas can be further specialized for the two classes of problems in question. The lower block triangular structure of A in (2.7) and (2.11) suggests a natural division of the set of columns into subsets treated together by partial pricing strategies. These subsets correspond to periods in (2.7) and to the first stage decision x and the realizations in (2.11). This idea was thoroughly investigated experimentally in [7] and the conclusions can be summarized as follows:

- rank the blocks (periods, realizations) equally and use them in a cyclic fashion;
- within each block (if it is still large enough) rank the columns equally and also use them in a cyclic fashion.

Again, pure linear algebraic concepts seem to be insufficient to fully specialize the pricing strategies. We should somehow exploit our knowledge of the essence of the underlying model to gain further improvements.

Noting that the dynamic model describes a phenomenon that evolves in time, we have grounds to expect that similar sets of activities will appear in the basis in the neighboring periods. This suggests a simple modification of the partial pricing strategy described above: if a prospective column has been found in period k, price out the corresponding columns from the next periods and bring them to the basis, as long as possible. The initial experiments reported in [9] indicate that this simple modification may improve the performance significantly (by 20-30% on problems of size 1000 by 2000.on IBM PC/XT).

In the stochastic case the situation is generally analogous, and only slightly more complicated. If a correction variable is basic for the realization (A_l,b_l) , we have grounds to expect that the corresponding variables will be basic for some neighboring realizations (A_j,b_j) However, contrary to the dynamic model, the notion of 'neighboring realizations' is not so clear and is difficult to implement. Nevertheless, this possibility should at least be investigated experimentally.

4. Feasible direction methods

The main disadvantage of the simplex method when applied to dynamic or stochastic models is that it changes only one nonbasic activity at a time. We have already observed that periods in the dynamic model and realizations in the stochastic model exhibit close similarities. This results in very long iteration paths of the simplex method with some subsequences of iterations used to realize similar changes for many periods or realizations. It would be much more convenient to perform these changes simultaneously.

The *feasible direction methods* (see [8],[20]) may help us to implement this idea (the simplex method is a feasible direction method, too, but with particularly simple directions). The main difference between these methods and the simplex method is that we change many nonbasic variables at a time and allow x_N to have values between their bounds at intermediate steps. We still preserve the division of x into x_B and x_N and still keep the conditions (3.1) and (3.2). However, steps 2, 3 and 4 of the simplex method are modified as follows.

Step 2a. Price out nonbasic columns a_j of A by calculating

$$z_j=c_j-\pi^T a_j \tag{4.1}$$

and select a subset S of columns a_l such that $z_j<0$ for $x_j=x_j^{\min}$, $z>0$ for $x_j=x_j^{\max}$, $z_j\neq0$ for $x_j^{\min}<x_j<x_j^{\max}$, (a subset of prospective candidates).

Step 3a. Determine a direction d_N of change of the nonbasic variables x_N such that

$$d_j z_j <0 \quad for \quad j\in S, \tag{4.2}$$

$$d_j=0 \quad for \quad j\notin S, \tag{4.3}$$

(in the simplex method d_N has only one nonzero component). Determine the direction of change of the basic variables by solving

$$Bd_B = Nd_N = A_s d_s, \tag{4.4}$$

where A_s is a submatrix of N formed from the columns selected in *Step 2a*, and d_s is the nonzero subvector of d_N.

Step 4a. Determine from $x_B^{\min}, x_B^{\max}$, x_B , d_B and $x_s^{\min}$, $x_s^{\max}$, x_s and d_s the variable which as first achieves its bound, when x_s moves in the direction d_s .

At first we note that when one of the variables which change their values (a basic from x_B or a nonbasic from x_s) will hit its bound, some nonbasic variables will be out of their bounds. So, we should either accept the fact that nonbasics can have arbitrary values in the course of calculation, or construct a basic solution from the current one without increasing the objective value. The second idea has been analysed in [20], where a detailed auxiliary algorithm has been described to pass to such a basic solution. This, however, involves many additional steps which may considerably diminish the advantages of changing many nonbasics in a major step. The radical solution of [8] seems to be more promising: we allow nonbasics to have values between their bounds. Under this assumption the division of x into basics and nonbasics is no longer determined uniquely by the algorithm. If the previous basics are still between their bounds, we can maintain the division to save on updating. When one of the basics hits its bound we can choose among x_s the variable to replace it. In general, as discussed in [8], we should aim at constructing such a basis that allows for an efficient next iteration. This may e.g. be accomplished by selecting a nonbasic which is possibly far away from its bounds. However, there is a need for a more theoretically grounded approach, which could perhaps be based on the analysis of the dual problem.

Since the algebra of the feasible direction method is close to that of the simplex method, we can of course use here all the tricks developed for compact inversion of basis matrices discussed in the previous section.

Leaving aside these technical points, let us now focus our attention on the specialization of the strategy of the feasible direction method to problems having dynamic or stochastic structure. The crucial question here is the choice of the direction of change of nonbasic variables. Although in theory the only limitations are the conditions (4.2), (4.3), in practice we have to use more restrictive conditions to limit the number of columns of N to be priced out. Again, as it was in the case of the primal simplex method, we can take advantage of the structure of the constraint matrix and of the similarities of the blocks. Thus, we can try to select to x_s at a given iteration similar activities from different periods/realizations and then make one major step of the method. The only difference is that previously we performed sequences of similar steps bringing to the basis corresponding activities from different blocks, while here we at first select a group of related candidates and then change them simultaneously.

An important feature of the feasible direction approach is the freedom for specifying the starting point. Indeed, once we abandoned the the requirement that all nonbasic variables are on their bounds, we are free to start the calculation from a solution which need not be basic. This may help solving practical problems, where reasonable nonbasic solutions can be specified by the user.

Summing up, the feasible direction approach appears to be a promising idea for large scale problems having a dynamic or stochastic structure. It retains the algebraic advantages of the simplex method and provides more freedom for exploiting the structure to shorten iteration paths. The potential of this approach is far from being exploited.

5. The regularized decomposition method

The idea of applying decomposition methods to linear programs of dynamic or stochastic structure has been known since 25 years [3], but it is still attractive and provides a framework for new ideas. We shall focus our attention here on the stochastic problem (2.10), whose structure directly suggests the application of decomposition, and we shall discuss the application of the new *regularized decomposition method* suggested in [24]. As for dynamic problems, the approaches suggested in the literature so far are entirely different and still of rather theoretical importance (see, e.g., [5], [6], [10], [11]).

By formulating the dual to (2.10) we obtain a problem of primal angular structure, to which the Dantzig-Wolfe decomposition method can be applied [4]. Since applying the Dantzig-Wolfe method to the dual is equivalent to applying *the Benders decomposition* to the primal [16], we shall discuss our basic ideas in primal terms.

It can be readily seen that if x is fixed in (2.10) the minimization with respect to $y_1, y_2, .., y_L$ can be carried out separately by solving for $l=1,2,..,L$ *the second-stage subproblems*

$$\text{minimize } q^T y$$

subject to

$$Wy = b_l - A_l x, \tag{5.1}$$
$$y^{\min} \le y \le y^{\max}.$$

Let us denote the optimal value of (5.1) by $f_l(x)$, and take the convention that $f_l(x)=+\infty$, if (5.1) is unsolvable. Then our problem (2.10) can be equivalently formulated as follows:

$$\text{minimize } F(x) \equiv c^T x + \sum_{l=1}^{L} p_l f_l(x)$$

subject to

$$x^{\min} \le x \le x^{\max}, \tag{5.3}$$

$$x \in X_l, \ l=1,2,..,L, \tag{5.4}$$

where

$$X_l = \{x : f_l(x) < +\infty\}. \tag{5.5}$$

We introduce the condition (5.4) to the problem formulation, because we are going to use separate approximations for f_l and for their domains X_l.

Much is known about the functions f_l and the sets X_l (see, e.g., [9]). In particular, each X_l is a convex closed polyhedron and each f_l is convex and piecewise linear on X_l. Although the pieces of f_l and the facets of f_l are not given explicitly, for each x we can determine a piece of f_l active at $\tilde{x}$, or a linear constraint defining X_l, which is violated at $\tilde{x}$.

Indeed, let (5.1) be solvable at $x=\tilde{x}$ and let π denote the vector of simplex multipliers associated with the solution. Then it follows from the duality relations in linear programming that for every x

$$f_l(x) \ge \pi^T (b_l - A_l x), \tag{5.6}$$

and the equality holds for $x=\tilde{x}$. If (5.1) is not solvable for $x=\tilde{x}$, then phase I of the simplex method or the dual simplex method will stop at a certain iteration, at which it will not be possible to move a basic variable y_{Br} towards its feasibility interval $[y_{Br}^{\min},y_{Br}^{\max}]$. If π is the r-th row of the basis inverse (if the dual method is used and $y_{Br}>y_{Br}^{\max}$), then

$$X_l\subseteq\{x\colon \pi^T(b_l-A_lx)\le y_{Br}^{\max}\}. \tag{5.7}$$

Similar formulae hold for the case of $y_{Br}<y_{Br}^{\min}$ and for the phase I of the primal simplex method.

We shall call the linear inequalities following from (5.6) *objective cuts*, and the inequalities following from (5.7) *feasibility cuts*. Each objective cut can be written as

$$\alpha_l+g_l^Tx\le f_l(x) \tag{5.8}$$

with $g_l=-A_l\pi$, $\alpha_l=\pi^Tb_l$. Each feasibility cut can be expressed in a similar fashion:

$$\bar{\alpha}_l+\bar{g}_l^Tx\le 0 \tag{5.9}$$

with $\bar{g}_l=-A_l^T\pi$ and an appropriately defined $\bar{\alpha}_l$. Functions f_l and sets X_l are polyhedral and there can be only finite many (although usually quite a few) such cuts.

Next, if we have objective cuts (5.8) for all $l=1,2,..,L$ we can construct an *aggregate cut*

$$\sum_{l=1}^{L} p_lf_l(x)\ge\alpha+g^Tx, \tag{5.10}$$

where (α,g) is computed from (α_l,g_l) by means of averaging

$$\alpha=\sum_{l=1}^{L} p_l\alpha_l, \tag{5.11}$$

$$g=\sum_{l=1}^{L} p_lg_l. \tag{5.12}$$

We can now describe the version of the Benders decomposition method for stochastic programs, known as *L-shaped algorithm* [30].

Let (α^j,g^j), $j\in J$, be the set of aggregate cuts (5.10) known so far, and let $\bar{\alpha}^j,\bar{g}^j)$, $j\in\bar{J}$, be the set of feasibility cuts generated previously. At each iteration of the method we perform the following operations.

Step 1. Solve the master problem:

$$\mathit{minimize}\quad \tilde{F}(x)=c^Tx+v$$

subject to

$$\alpha^j+(g^j)^Tx\le v,\ \ j\in J, \tag{5.14}$$

$$\bar{\alpha}^j+(\bar{g}^j)^Tx\le 0,\ \ j\in J, \tag{5.15}$$

$$x^{min}\le x\le x^{max}.$$

Let $\tilde{x}$ be the solution to (5.13)-(5.16).

Step 2. Solve for l=1,2,..,L the subproblems (5.1) at $x=\tilde{x}$. If any of them is infeasible, generate the corresponding feasibility cut (5.9), append it to (5.15) and go to Step 1.

If all subproblems are feasible, check whether $\sum_{l=1}^{L} p_lf_l(\tilde{x})=v$. If this condition is satisfied, then stop; otherwise generate objective cuts (5.8), the aggregate cut (5.10), append it to (5.14) and go to Step 1.

It is not difficult to observe that this method exactly corresponds to the Dantzig-Wolfe method applied to the dual of (2.10): the cuts passed to the master (5.13)-(5.15) are the proposals passed to the master in the Dantzig-Wolfe method.

The attractiveness of this approach follows from the fact that the solution procedure closely reflects the structure of the original problem. It also allows for some parallelism in subproblem solution. It has, however, inherent drawbacks common for all purely linear cutting plane methods (cf., e.g., [29]), and for the Dantzig-Wolfe method (which is in fact their dual counterpart):

- the number of cuts (5.14), (5.15) increases in the course of calculation;
- the master problem is unstable: new cuts may imply rapid changes of $\tilde{x}$;
- convergence is slow.

These drawbacks led to the idea of the regularized decomposition method [24], which combines the Benders decomposition with modern stable techniques of nonsmooth optimization [15]. The main idea of the method is to change the master program, which generates successive points x^k at which the subproblems are solved. We aim at constructing such a master which would be able to use the information gained in the past not only in the form of cuts, but also in the form of the best point x found so far.

The method uses objective and feasibility cuts (5.8) and (5.9) as before. It does not, however, average them to form aggregate cuts (5.10), but rather maintains separate sets of cuts for each component f_l :

$$\alpha_l^j+(g_l^j)^T x \leq f_l(x), \ \ j\in J, \ \ l=1,2,..,L.$$

Next, the master problem, although quite similar to (5.13)-(5.16), is augmented with a quadratic penalty term for the distance of $\tilde{x}$ to the best point x^k found so far:

$$minimize \ \ \tilde{F}^k(x)\equiv\frac{1}{2}\|x-x^k\|+c^T x+\sum_{l=1}^{L} p_l v_l \tag{5.17}$$

subject to

$$\alpha_l^j+(g_l^j)^T x\leq v_l, \ \ j\in J, \ \ l=1,2,..,L, \tag{5.18}$$

$$\bar{\alpha}^j+(\bar{g}^j)^T x\leq 0, \ \ j\in \bar{J}, \tag{5.19}$$

$$x^{min}\leq x\leq x^{max}. \tag{5.20}$$

The existence of this quadratic term stabilizes the master problem, i.e. makes it less sensitive to the changes in the set of cuts (5.18)-(5.19). It also allows for skipping outdated cuts and keeping the total size of the master limited.

The logic of the regularized decomposition method can be summarized as follows.

Step 1. Solve the regularized master (5.17)-(5.20), getting a trial point $\tilde{x}$ and objective estimates v_l, $l=1,2,..,L$.

Step 2. Solve for $l=1,2,..,L$ the subproblems (5.1) at $x=\tilde{x}$.

a) If (5.1) is infeasible, then append the feasibility cut (5.9) to (5.19).

b) If (5.1) is feasible, but $f_l(x)>v_l$, then append the objective cut (5.8) to the set of cuts J_l in (5.18).

Step 3. Change the regularizing point x^k according to the following rules.

a) If there were infeasible subproblems (5.1), set $x^{k+1}=x^k$.

b) If $F(\tilde{x})=c^T\tilde{x}+\sum_{l=1}^{L}p_lv_l$, then set $x^{k+1}=\tilde{x}$.

c) If $F(\tilde{x})\leq\gamma F(x^k)+(1-\gamma)(c^T\tilde{x}+\sum_{l=1}^{L}p_lv_l)$ and exactly $n+L$ constraints were active in (5.17)-(5.20), then also set $x^{k+1}=\tilde{x}$; otherwise set $x^{k+1}=x^k$.

Step 4. Delete from the cuts (5.18)-(5.19) some of those which were not active at the last solution $\tilde{x}$ to the master, and go to Step 1.

It is easy to observe that the number of active cuts (i.e. linearly independent constraints with positive Lagrange multipliers) never exceeds $n+L$, where n is the dimension of x and L is the number of blocks (realizations). Since at Step 2 at most L new cuts may enter (either a feasibility cut or an objective cut for each l), the total number of cuts need not exceed $n+2L$. In fact, it is usually much smaller, if many bounds (5.20) are active.

It has been proved in [24] (for the general case of minimization of a sum of polyhedral functions) that the rules for changing the regularizing point x^k at Step 3 guarantee that the sequence x^k is convergent in finite many iterations to the solution of our problem. This result obviously applies also to the particular problem we are interested in.

It is easy to observe that the use of the quadratic term in (5.17) implies that the regularizing point x^k has a great influence on the solution of the master problem. In particular, the starting point x^0 influences considerably the whole iteration path, which is obviously not true for the linear decomposition method. This may significantly reduce the effort required for solving practical problems, where a good starting point is available.

These important theoretical features have been obtained at the expense of replacing a purely linear master problem (5.13)-(5.16) by the quadratic problem (5.17)-(5.20). To make the regularized decomposition method really competitive, we need an efficient computational technique for solving the regularized master.

Such a technique can be based on the *active set strategy*. It consists in selecting a subset of the constraints (5.18)-(5.20) to be satisfied as equalities, solving the resulting equality constrained subproblem, changing the active set, solving the new subproblem, etc. The active set is increased, when a cut not included in it is violated, and it is decreased, when a cut in the active set has a negative Lagrange multiplier in the subproblem.

The equality constraints defined by an active set can be compactly written in the form

$$a+G^Tx-E^Tv=0, \tag{5.21}$$

where a is composed of the constant terms $\alpha^j,\bar{\alpha}^j$corresponding to the active cuts, G has columns $g^j,\bar{g}^j$, and E is a zero-one matrix whose j-th column is the unit vector e^l if the j-th cut is an objective cut for f_l , and is a zero column otherwise. Active bounds (5.20) can also be put into (5.21) with particularly simple columns of G (unit vectors). Thus each equality constrained subproblem has the form: minimize (5.17) subject to (5.21). Denoting by λ the vector of Lagrange multipliers corresponding to the active cuts (5.21), we can formulate the following necessary and sufficient conditions of optimality:

$$E\lambda = p, \tag{5.22}$$

$$E^T v + G^T G\lambda = G^T(x^k - c) + a, \tag{5.23}$$

where $p=(p_1,p_2,..,p_L)$ is the vector of probabilities. The primal solution is defined by

$$x = x^k - c - G^T\lambda. \tag{5.24}$$

The number of active cuts does not exceed $n+L$ and so does the size of the system (5.22)-(5.23). However, the specific structure of E (unit or zero columns and full row rank) makes it possible to further reduce the dimension by representing

$$E=(I,N),$$

$$G=(G_B,G_N),$$

$$a=(a_B,a_N),$$

$$\lambda=(\lambda_B,\lambda_N).$$

After eliminating analytically v and λ_B from (5.22)-(5.23) we obtain the equivalent system

$$\hat{G}_N^T \hat{G}_N \lambda = \hat{G}_N^T(x - c - \bar{g}_N) + \hat{a}_N, \tag{5.25}$$

where

$$\hat{G}_N = G_N - G_B N,$$

$$\hat{a}_N = a_N - N^T a_B,$$

$$\bar{g}_N = G_B p.$$

The system (5.25) has dimension not exceeding the dimension of x, independently of the number of blocks L, and can be solved by stable numerical techniques for least-squares problems (see [2], [24]). In the implementation [25] additional advantages have been drawn from the activity of simple bounds, which further reduces the dimension of (5.25).

Summing up, not only the regularized master (5.17)-(5.20) has a smaller number of cuts than (5.13)-(5.16), but the effort for solving it is comparable with the effort for solving linear problems of the same size. These observations have been confirmed by the initial experiments with the regularized decomposition method for large scale stochastic programs, which we shall report in an extended form elsewhere [26]. They indicate that the method solves medium-size problems (200 by 500) 2...3 times faster than purely linear techniques, is capable of solving very large problems (problems of size 2500 by 5000 in ca. 1 min. on IBM 3033) and the growth of costs is sublinear when the number of realizations L increases.

Conclusions

We discussed some modern computational approaches to large scale linear programs arising from dynamic and stochastic models. In our opinion, two directions deserve more attention as promising tools for decision support systems:

- *feasible direction methods* with special compact inverse techniques borrowed from implementations of the simplex method and with specialized direction-finding procedures;

- *the regularized decomposition method* with decentralized or parallel subproblem solution.

The common feature of these methods is the freedom in specifying the starting point and its strong influence on the cost of calculations, which is crucial for decision support systems, where we usually solve repeatedly similar models. The methods are also more flexible than simplex-based approaches and provide a potential for an interactive control of calculations and for some parallelism. On the other hand, they both can use computer resources at least so economically as the simplex methods and are capable of solving large models.

References

[1] J. Bisschop and A. Meeraus, "Matrix augmentation and structure preservation in linearly constrained control problems", Mathematical Programming 18(1980) 7-15.

[2] J.W. Daniel et al., "Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization", Mathematics of Computation 30(1976) 772-795.

[3] G. Dantzig, Linear Programming and Extensions, Princeton 1963.

[4] G. Dantzig and A. Madansky, "On the solution of two-stage linear programs under uncertainty", in Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability, vol 1, University of California Press, Berkeley 1961, pp. 165-176.

[5] J.J.H. Forrest and J.A. Tomlin, "Updated triangular factors of the basis to maintain sparsity in the product form simplex method", Mathematical Programming 2(1972) 263-278.

[6] R. Fourer, "Solving staircase linear programs by the simplex method, 1: inversion", Mathematical Programming 23(1982) 274-313.

[7] R. Fourer, "Solving staircase linear programs by the simplex method, 2: pricing", Mathematical Programming 25(1983) 251-292.

[8] R. Gabasov and F.M. Kirillova, Linear Programming Methods, Isdatelstvo BGU, Minsk 1977. (in Russian)

[9] J. Gondzio and A. Ruszczynski, "A package for solving dynamic linear programs", Institute of Automatic Control, Warsaw University of Technology, 1986.

[10] J. Ho and E. Loute, "A set of staircase linear programming test problems", Mathematical Programming 20(1981) 245-250.

[11] J. Ho and A. Manne, "Nested decomposition for dynamic models", Mathematical Programming 6(1974) 121-140.

[12] P, Kall, "Computational methods for solving two-stage stochastic linear programming problems", ZAMT 30(1979) 261-271.

[13] P. Kall, K. Frauendorfer and A. Ruszczynski, "Approximation techniques in stochastic programming", in: Y. Ermoliev and R. Wets (eds), Numerical Methods in Stochastic Programming, Springer-Verlag, Berlin 1986 (to appear).

[14] M. Kallio and E. Porteus, "Triangular factorization and generalized upper bounding techniques", Operations Research 25(1977) 89-99.

[15] K. C. Kiwiel, Methods of Descent for Nondifferentiable Optimization, Springer-Verlag, 1985.

[16] L. S. Lasdon, Optimization Theory for Large Systems, Macmillan, New York 1970.

[17] R. Marsten, "The design of the XMP linear programming library", ACM Transactions of Mathematical Software 7(1981) 481-497.

[18] B. Murtagh, Advanced Linear Programming, McGraw-Hill, 1981.

[19] B. Murtagh and M. Saunders, "MINOS 5.0. User's guide", System Optimization Laboratory, Stanford University, 1984.

[20] K. G. Murty and Y. Fathi, "A feasible direction method for linear programming", Operations Research Letters 3(1984) 121-127.

[21] A. Perold and G. Dantzig, "A basis factorization method for block triangular linear programs", in I. Duff and G. Stewart (eds), Sparse Matrix Proceedings, SIAM, Philadelphia, 1979, pp. 283-313.

[22] A. Propoi and V. Krivonozhko, "The simplex method for dynamic linear programs", RR-78-14, IIASA, 1978.

[23] J. Reid, "A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases", Mathematical Programming 24(1982) 55-69.

[24] A. Ruszczynski, "A regularized decomposition method for minimizing a sum of polyhedral functions", Mathematical Programming 35(1986) 309-333.

[25] A. Ruszczynski, "QDECOM: The regularized decomposition method. User's manual", Institute of Operations Research, University Zurich, 1985.

[26] A. Ruszczynski, "Regularized decomposition of stochastic programs: algorithmic techniques and numerical results", in preparation.

[27] B. Strazicky, "Some results concerning an algorithm for the discrete recourse problem", in: M. Dempster (ed.), Stochastic Programming, Academic Press, London 1980, pp. 263-274.

[28] E. Toczylowski, "A hierarchical representation of the inverse of sparse matrices", SIAM J. Alg. Disc. Math. 5(1984) 43-56.

[29] J. M. Topkis, "A cutting plane algorithm with linear and geometric rates of convergence", JOTA 36(1982) 1-22.

[30] R. Van Slyke and R. J.-B. Wets, "L-shaped linear programs with applications to optimal control and stochastic programming", SIAM J. on Applied Mathematics 17(1969) 638-663.

[31] R.J.-B. Wets, "Stochastic programming: solution techniques and approximation schemes", in: A.Bachem et al. (eds), Mathematical Programming: The State of the Art, Springer-Verlag, Berlin 1983, pp. 507-603.

[32] R. J.-B. Wets, "Large scale linear programming techniques in stochastic programming", in: Y. Ermoliev and R. Wets (eds), Numerical Methods in Stochastic Programming, Springer-Verlag, Berlin 1986 (to appear).

Theoretical Guide for NOA2: a FORTRAN Package of Nondifferentiable Optimization Algorithms

Krzysztof C. Kiwiel, Andrzej Stachurski

Systems Research Institute, Polish Academy of Sciences.

ABSTRACT

This paper forms a theoretical guide for NOA2, a package of FORTRAN subroutines designed to locate the minimum value of a locally Lipschitz continuous function subject to locally Lipschitzian inequality and equality constraints, general linear constraints and simple upper and lower bounds. The user must provide a FORTRAN subroutine for evaluating the (possibly nondifferentiable and nonconvex) problem functions and their single subgradients. The package implements several descent methods, and is intended for solving small-scale nondifferentiable minimization problems on a professional microcomputer.

1. Introduction

NOA2 is a collection of FORTRAN subroutines designed to solve small-scale nondifferentiable optimization problems expressed in the following standard form

$$\textit{minimize}\quad f(x):=\max\{\, f_j(x)\colon j=1,\dots,m_O \,\}, \tag{1a}$$

$$\textit{subject to}\ F_j(x) \le 0 \text{ for } j=1,\dots,m_I, \tag{1b}$$

$$F_j(x) = 0 \text{ for } j=m_I+1,\dots,m_I+m_E, \tag{1c}$$

$$Ax \le b, \tag{1d}$$

$$x_i^L \le x_i \le x_i^U \text{ for } i=1,\dots,n, \tag{1e}$$

where the vector $x=(x_1,\dots,x_n)^T$ has n components, f_j and F_j are locally Lipschitz continuous functions, and where the m_A by n matrix A, the $m_A-\textit{vector}$ b and the n-vectors x^L and x^U are constant; A is treated as a dense matrix.

The nonlinear functions f_j and F_j need not be continuously differentiable (have continuous gradients, i.e. vectors of partial derivatives). In particular, they may be convex. The user has to provide a FORTRAN subroutine for evaluating the problem functions and their single subgradients (called generalized gradients by Clarke (1983)) at each x satisfying the linear constraints (1d,e). For instance, if F_j is smooth then its subgradient $g_{F_j}(x)$ equals the gradient $\nabla F_j(x)$, whereas for the max function

$$F_j(x) = \max\{\, F_j(x;z)\colon z \in Z\} \tag{2}$$

which is a pointwise maximum of smooth functions $F_j(\cdot,\cdot)$ on a compact set Z, $g_{F_j}(x)$ may be calculated as the gradient $\nabla_x F_j(x;z(x))$ (with respect to x), where $z(x)$ is an arbitrary solution to the maximization problem in (2). (Surveys of subgradient calculus, which generalizes rules like $\nabla(F_1+F_2)(x) = \nabla F_1(x)+\nabla F_2(x)$, may be found in Clarke (1983) and Kiwiel (1985a).)

NOA2 implements the descent methods of Kiwiel (1985a-d,1986a, 1986c,1987), which stem from the works of Lemarechal (1978) and Mifflin (1982).

A condensed form of problem (1) is to

$$\text{minimize } f(x) \text{ over all } x \text{ in } R \tag{3a}$$

$$\text{satisfying } F_I(x) \leq 0, \tag{3b}$$

$$F_E(x)=0, \tag{3c}$$

$$Ax \leq b, \tag{3d}$$

$$x^L \leq x \leq x^U, \tag{3e}$$

where f is the *objective* function,

$$F_I(x) = \max\{F_j(x)\colon j=1,...,m\}$$

is the *inequality constraint* function,

$$F_E(x) = \max\{\max[F_j(x),-F_j(x)]\colon j=m_I+1,...,m_I+m_E\}$$

is the *equality constraint* function, the m_A inequalities (3d) are called the *general linear constraints* , whereas the *box constraints* (3e) specify upper and lower *simple bounds* on all variables.

The standard form (1) is more convenient to the user than (3), since the user does not have to program additional operations for evaluating the functions F_I and F_E and their subgradients. On the other hand, the condensed form facilitates the description of algorithms.

The linear constraints are treated specially by the solution algorithms of NOA2, which are feasible with respect to the linear constraints, i.e. they generate successive approximations to a solution of (1) in the set

$$S_L = \{x\colon Ax \leq b \text{ and } x^L \leq x \leq x^U\}.$$

The user must supply an initial estimate $\tilde{x}$ of the solution that satisfies the box constraints ($x^L \leq \tilde{x} \leq x^U$); the orthogonal projection of $\tilde{x}$ onto S_L is taken as the algorithm's starting point.

Two general techniques are used to handle the nonlinear constraints. In the first one, which minimizes an exact penalty function for (1) over S_L, the initial point need not lie in

$$S_F = \{x\colon F_I(x) \leq 0 \text{ and } F_E(x) = 0\}$$

and the successive points converge to a solution from outside of S_F . The second one uses a feasible point method for the nonlinear inequality constraints, which starts from a point in

$$S_I = \{x\colon F_I(x) \leq 0\}$$

and keeps the successive iterates in S_I. The choice between the two techniques is made by the user, who may thus influence the success of the calculations. For a given level of final accuracy, the exact penalty technique usually requires less work than the feasible point technique. On the other hand, the feasible point technique may be more reliable and is more widely applicable, since it does not in fact require the evaluation of f and F_E outside of $S_L \cap S_I$.

NOA2 is designed to find solutions that are locally optimal. If the nonlinear objective and inequality constraint functions are convex within the set S_L, and the nonlinear equality constraints are absent, any optimal solution obtained will be a global minimum. Otherwise there may exist several local minima, and some of these may not be global. In such cases the chances of finding a global minimum are usually increased by restricting the search to a sufficiently small set S_L and choosing a starting point that is "sufficiently close" to a solution, but there is no general procedure for determining what "close" means, or for verifying that a given local minimum is indeed global.

NOA2 stands for *Nondifferentiable Optimization Algorithms* , version 2.0.

In the following sections we introduce some of the terminology required, and give an overview of the algorithms used in NOA2.

2. An overview of algorithms of NOA2

The algorithms in NOA2 are based on the following general concept of descent methods for nondifferentiable minimization. Starting from a given approximation to a solution of (1), an iterative method of descent generates a sequence of points, which should converge to a solution. The property of descent means that successive points have lower objective (or exact penalty) function values. To generate a descent direction from the current iterate, the method replaces the problem functions with their piecewise linear (*polyhedral*) approximations. Each linear piece of such an approximation is a linearization of the given function, obtained by evaluating the function and its subgradient at a trial point of an earlier iteration. (This construction generalizes to the nondifferentiable case the classical concept of using gradients to linearize smooth functions.) The polyhedral approximations and quadratic regularization are used to derive a local approximation to the original optimization problem, whose solution (found by quadratic programming) yields the search direction. Next, a line search along this direction produces the next approximation to a solution and the next trial point, detecting the possible gradient discontinuities. The successive approximations are formed to ensure convergence to a solution without storing too many linearizations. To this end, subgradient selection and aggregation techniques are employed.

2.1. Unconstrained convex minimization

The unconstrained problem of minimizing a convex function f defined on R^n is a particular case of problem (1). In NOA2 this problem may be solved by the method with subgradient selection (Kiwiel, 1985a).

Let $g_f(y)$ denote the subgradient of f at y calculated a subroutine supplied by the user. In the convex case

$$f(x) \geq f(y) + <g_f(y),x-y> \text{ for } \mathit{all} \ x, \tag{4}$$

where $< \cdot , \cdot >$ denotes the usual inner product. Thus at each y we can construct the *linearization* of f

$$\bar{f}(x;y) = f(y) + <g_f(y),x-y> \text{ for } \mathit{all} \ x, \tag{5}$$

which is a lower approximation to f.

Given a user-provided initial point x^1, the algorithm generates a sequence of points x^k, k=2,3,..., that is intended to converge to a minimum point of f. At the k-th iteration the algorithm uses the following *polyhedral approximation* to f

$$\hat{f}^k(x) = \max\{\bar{f}(x;y^j):\ j\in J_f^k\} \tag{6}$$

derived from the linearizations of f at certain *trial points* y^j of earlier iterations j, where the index set $J_f^k\subset\{1,...,k\}$ typically has $n+2$ elements. Note that $\hat{f}^k$ may be a tight approximation to f in the neighborhood of trial points y^j, for j in J_f^k, since $f(y^j)=\hat{f}^k(y^j)$.

The best direction of descent for f at x^k is, of course, the solution $\hat{d}^k$ to the problem

$$\textit{minimize}\ \ f(x^k+d)\ -\textit{all}\ \ d\ \ \textit{in}\ \ R^n,$$

since $x^k+\hat{d}^k$ minimizes f. The algorithm finds an approximate descent direction d^k to

$$\textit{minimize}\ \ \hat{f}^k(x^k+d)+|d|^2/2\ -\textit{all}\ \ d, \tag{7}$$

where the regularizing penalty term $|d|^2/2$ tends to keep x^k+d^k in the region where $\hat{f}^k$ may be a good approximation to f ($|\cdot|$ denotes the Euclidean norm); without this correction term, problem (7) needs not have a bounded solution.

The nonpositive quantity

$$v^k = \hat{f}^k(x^k+d^k)-f(x^k) \tag{8}$$

is an *optimality measure* of x^k, since

$$f(x^k) \leq f(x) + |v^k|^{1/2}|x-x^k|-v^k \ \text{ for } \textit{all}\ \ x. \tag{9}$$

The algorithm terminates if

$$|v^k| \leq \epsilon_s(1+|f(x^k)|), \tag{10}$$

where ϵ_s is a positive final *accuracy tolerance* provided by the user. Thus for $\epsilon_s=10^{-l}$ and $l\geq 4$, we may hope to achieve the relative accuracy of about (l-1) leading digits in the objective value (considering also zeros after the decimal point as significant), i.e. typically at termination

$$|f(x^*)-f(x^k)|\ \ \textit{is about}\ \ 10^{-(l-1)}\textit{max}\ \{|f(x^*)|,1\}, \tag{11}$$

where x^* is a minimum point of f. Of course, such estimates may be false for ill-conditioned problems. In practice v^k usually converges to a negative number, small relative to $\textit{max}\ \{f(x^k),1\}$.

The stopping criterion (10) usually works with ϵ_s set to 10^{-4} or 10^{-6}, but it is not always reliable. For instance, if f is polyhedral and bounded from below then termination should occur at some iteration with $v^k=0$ (and optimal x^k). In practice, computer rounding errors prevent the vanishing of v^k. The search direction finding subproblem (7) is solved in NOA2 by the subroutine QPDF4 for quadratic programming (Kiwiel, 1986b), which calculates the quantity

$$\tilde{v}^k = \hat{f}^k(x^k+d^k)-f(x^k) \tag{12}$$

and gives v^k a nonpositive value according to some dual estimate; in theory $\tilde{v}^k$ should equal v^k. The smallness of $|\tilde{v}^k-v^k|$ relative to $|v^k|$ indicates good accuracy of QPDF4. The accuracy usually deteriorates in the neighborhood of a minimum point of f (when too small accuracy tolerance ϵ_s prevents termination), or earlier for ill-conditioned problems. The case of $\tilde{v}^k\geq 0$, i.e. inability to find a descent direction, enforces abnormal termination.

If the algorithm does not terminate, then the negative value of v^k (see (8)) *predicts the descent* $f(x^k+d^k)-f(x^k)$ for the step from x^k to x^k+d^k. Usually v^k over-estimates the descent because $f(\cdot)\geq\hat{f}^k(\cdot)$ and $\hat{f}^k$ need not agree with f at x^k+d^k if its linearizations do not reflect all discontinuities in the gradient of f around x^k (too few of them to make up $\hat{f}^k$, or they were calculated at y^j far from x^k). Thus two cases are possible when a line search is made to explore f along the segment joining x^k and x^k+d^k. Either $\hat{f}^k$ is a good model of f and it is possible to make a *serious step* by finding a stepsize $t_L^k>0$ such that the next iterate

$$x^{k+1} = x^k+t_L^k\, d^k$$

has a lower objective value than x^k, or a *null step* $x^{k+1}=x^k$ ($t_L^k=0$) combined with calculating the linearization $f(\cdot;y^{k+1})$ at a new trial point

$$y^{k+1} = x^k + t_R^k\, d^k$$

with $t_R^k\in(0,1]$ may be used to get the next improved model $\hat{f}^{k+1}$ of f. Since $0\leq t_L^k\leq t_R^k$, t_L^k and t_R^k are called *left* and *right* stepsizes respectively, although they may coincide if $t_L^k>0$. More specifically, a serious step with $t_L^k>0$ is made if

$$f(x^{k+1}) \leq f(x^k)+m_L\, t_L^k v^k, \tag{13a}$$

$$t_L^k\geq\bar{t} \quad or \quad \alpha_f(x^k,x^{k+1}) > m_v\,|v^k|, \tag{13b}$$

where m_L, m_v and $\bar{t}$ are positive parameters less than 1, whereas

$$\alpha_f(x,y) = f(x) - \bar{f}(x;y) \tag{14}$$

is the linearization error of $f(\cdot;y)$ at x. These conditions ensure a significant objective decrease (i.e. t_L^k and $m_L t_L^k v^k$ cannot be too small). On the other hand, a null step with $t_L^k=0$ and $t_R^k\in[\bar{t},1]$ must ensure that the new linearization satisfies

$$\bar{f}(x^k+d^k;y^{k+1})-f(x^k) \geq m_R[\hat{f}^k(x^k+d^k)-f(x^k)] = m_R v^k$$

for some fixed $m_R\in(0,1)$, so that its incorporation will make $\hat{f}^{k+1}$ a better approximation to f along the direction d^k from $x^{k+1} = x^k$ than $\hat{f}^k$ was, thus enhancing generation of a better next direction d^{k+1}.

For technical reasons, the line search parameters must be positive and satisfy $m_L+m_v<m_R<1$ and $\bar{t}\leq1$. By changing the standard values $m_L=0.1$, $m_R=0.5$, $m_v=0.01$ and $\bar{t}=0.01$, the user may strongly influence the algorithm's efficiency on a given problem. Note that the total amount of work in solving a problem depends on the number of function and subgradient evaluations as well as on the number of iterations. The algorithm may require only one objective evaluation per iteration. This is justified if the cost of one objective evaluation dominates the effort of auxiliary operations (mainly at quadratic programming) per iteration. In the reverse case, one may wish to decrease the number of iterations at the cost of increasing the number of objective evaluations.

More specifically, the line search checks if *trial stepsizes* $t\in[\bar{t},1]$, starting with $t=1$, satisfy the sufficient descent criterion

$$f(x^k+td^k) \leq f(x^k)+m_L t v^k$$

and thus are candidates for t_L^k. Hence if the threshold stepsize $\bar{t}$ is set to 1, only t=1 need be tested, and a serious step with $t_L^k=1$ will occur if

$$f(x^k+d^k) - f(x^k) \leq m_L\, [\hat{f}^k(x^k+d^k) - f(x^k)]$$

(see (8) and (13a)), i.e. $\hat{f}^k$ must be very close to f at x^k+d^k if m_L approaches 1. In practice $m_L>0.5$ may result in many null steps (the algorithm concentrates on improving its models $\hat{f}^k$ of f between infrequent serious steps), whereas $m_L<0.1$ may produce (damped) oscillations of $\{x^k\}$ around the solution (little descent is made at each serious step). For a smaller threshold $\bar{t}<1$, more stepsizes t are tested (typically two for $\bar{t}=0.1$, three for $\bar{t}=0.01$), and there are fewer null steps. In practice decreasing $\bar{t}$ from 1 to 0.01 will usually decrease the number of iterations at the cost of more function evaluations.

It is worth adding that for a polyhedral f one may frequently use the values $m_L=0.9$, $m_R=0.95$, $m_v=0.01$ and $\bar{t}=1$, which parameter values, however, are usually inefficient for more general functions.

To sum up, it is reasonable to set m_L and $\bar{t}$ in the ranges [0.1,0.9] and [0.01,1] respectively, and use $m_v=0.001$ and $m_R=(1+m_L)/2$.

The user may trade-off storage and work per iteration for speed of convergence by choosing the maximum number M_g of past subgradients (linearizations) involved in the approximations $\hat{f}^k$ (whereas more linearizations increase the model accuracy). To ensure convergence, the algorithm selects the linearizations active at the solution to subproblem (7) for keeping (their indices enter J_f^{k+1} together with $k+1$), whereas inactive past linearizations may be dropped (i.e. overwritten in the memory by new ones, if necessary). More linearizations enhance faster convergence by producing more accurate $\hat{f}^k$, but the costs of solving subproblem (7) may become prohibitive. Using M_g greater than its minimal possible value n+3, $M_g=2n$ say, frequently increases the overall efficiency.

An additional increase of modelling accuracy may be possible when f is the pointwise maximum

$$f(x) = \max\{\ f_i(x)\colon\ i=1,...,m_O\}$$

of several convex functions f_i with subgradients g_{f_i}. The user may choose a positive *activity tolerance* ϵ_a and the maximum number l_a of additional linearizations of f_i at x^k that will augment $\hat{f}^k$. Then subproblem (7) employs

$$\hat{f}^k(x) = \max\{\bar{f}(x;y^j);\ \ y^j \in J_f^k;\ f_i(x^k)+<g_{f_i}(x^k),x-x^k>\colon\ \ i\in L^k\}, \tag{15}$$

where L^k contains at most l_a indices of the ϵ_a-active functions $f_i(x^k)\geq f(x^k)-\epsilon_a$. However, these additional linearizations may overwrite some past ones (if M_g is too small), and this may or may not increase the accuracy of $\hat{f}^k$ at points remote from x^k.

If space limitations prevent the algorithm from storing sufficiently many $(M_g>n+3)$ past subgradients, the algorithm may be run with $M_g>3$ by employing subgradient aggregation instead of selection. This will usually - sometimes even drastically - decrease the speed of convergence.

The algorithm described so far is rather sensitive to the objective scaling, especially to the multiplication of f by a positive constant, mainly due to the presence of the arbitrary quadratic term in subproblem (7). For greater flexibility, the user may choose a positive *weight* u in the following version of (7)

$$\textit{minimize}\ \ \hat{f}^k(x^k+d)\ +\ u\ |d|^2/2\ -\textit{all}\ \ d. \tag{16}$$

The standard value $u=1$ suffices for well-scaled problems. If f varies rapidly, increasing u will decrease $|d^k|$, thus localizing the search for a better point to the neighborhood of x^k. For instance, if the initial derivative v^1 of f at x^1 in the direction d^1 is "large" (e.g. $v^1<-10^5$), one may try a larger u, $u=100$ say, in the next algorithm's run on the same, or related problem. On the other hand, too "large" u will produce many serious, but short steps with very small $|x^{k+1}-x^k|$, and convergence will be slow. We may add that for piecewise linear objectives smaller values of u are less dangerous than too large. Moreover, large errors may arise in the solution of (16) by the subroutine QPDF4 if u is small ($u<10^{-4}$); then it is better to multiply f by a small number and set $u=1$.

In the general case of $u>0$, the optimality estimate (9) becomes

$$f(x^k) \le f(x) + u|v^k|^{1/2}|x-x^k| - v^k \text{ for } \textit{all } x. \tag{17}$$

This suggests that the accuracy tolerance ϵ_s should be decreased when a larger u is used; otherwise, "false" convergence will occur.

2.2. Linearly constrained convex minimization

The box constrained problem with a convex f

$$\textit{minimize} \quad f(x), \tag{18a}$$

$$\textit{subject} \quad x_i^L \le x_i \le x_i^U \text{ for } i=1,...,n, \tag{18b}$$

can be solved in NOA2 by a modification of the method described in the preceding section (Kiwiel, 1985c,1986c,1987).

The presence of finite upper and lower bounds ensures the existence of a solution and prevents divergence of the algorithm, which must occur when there is no solution (then $|x^k|$ tends, in theory, to infinity; in practice - until an arithmetic overflow terminates the calculation). It is always advisable to place bounds of the form $-1000\le x_i\le 1000$, which should not be active when the solution lies inside the box.

The objective f and its subgradient g_f will be evaluated only inside the box $[x^L,x^U]$. This may be used to eliminate regions where f is undefined. For example, if $f(x)=x_1^{1/2}+exp(x_2)$, it is essential to place bounds of the form $x_1\ge 10^{-5}$, $x_2\le 20$.

If the user specifies an infeasible initial point x^1, it is projected on the box (by replacing x_i^1 with $\max\{x_i^L, min(x_i,x_i^U)\}$). Successive x^k remain in the box.

At the k-th iteration, an approximate feasible descent direction d^k is found to

$$\textit{minimize} \quad \hat{f}^k(x^k+d) + u|d|^2/2, \tag{19a}$$

$$\textit{subject} \quad x_i^L \le x_i^k+d_i \le x_i^U, \text{ for } i=1,...,n. \tag{19b}$$

This subproblem is a natural extension of (16). Consequently, the preceding remarks on the choice of parameters remain in force.

We may add that the introduction of box constraints only slightly increases the work at the search direction finding.

For the problem with general linear constraints

$$\textit{minimize} \quad f(x), \quad \textit{subject} \quad Ax \le b, \tag{20}$$

the search direction finding subproblem becomes

$$\textit{minimize} \quad \hat{f}^k(x^k+d) + u|d|^2/2, \tag{21a}$$

$$\textit{subject} \quad A\ (x^k+d) \le b. \tag{21b}$$

Due to rounding errors, the calculated direction d^k need not be "strictly" feasible. To measure the infeasibility of a direction d we use the *constraint violation function*

$$v_2(d) = \max_{x^k}\{\ h(x^k+d),0\}$$

defined in terms of

$$h(x) = max\ \{\ A_i x - b^i\colon\ i=1,...,m_A\}, \tag{22}$$

where A_i denotes the i-th row of A. Subproblem (21) is equivalent to the unconstrained problem

$$\textit{minimize} \quad \hat{f}^k(x^k+d) + u|d|^2/2 + cv_2(d) \ -\textit{all}\ \ d \tag{23}$$

when the *penalty parameter* c is sufficiently large. Hence we may test increasing values of c until the solution of (23) is feasible, and hence solves (21). Starting from $c=\rho$, where $\rho>0$ may be provided by the user, each successive c is multiplied by 10 until the solution d^k of (23) passes the *feasibility test*

$$h(x^k+d^k) \le \epsilon_F, \tag{24}$$

where ϵ_F is a positive absolute feasibility tolerance. If this test is failed by even "very large" c, the calculation terminates. This occurs if $c>1/\epsilon_M$, where ϵ_M is the *relative machine accuracy* (the smallest positive ϵ for which $1+\epsilon>1$ in the computer's arithmetic).

No computational difficulties should arise if the linear constraints are well-scaled and the feasibility tolerance ϵ_F is large enough. In particular, it may be necessary to ensure that the coefficients of A are of order 1 and $\epsilon_F \ge \epsilon_M^{1/2}$. For instance, if the coefficients of A result from measurements corrupted by errors of magnitude 10^{-6}, one should set $\epsilon_F=10^{-6}$.

If the initial point specified by the user is not feasible to within the tolerance ϵ_F, the algorithm tries to project it onto the feasible set (by using a version of (23)). If the projection is successful, each successive x^k satisfies the linear constraints to within ϵ_F. Moreover, f(y) and $g_f(y)$ are calculated only at ϵ_F - feasible points with $h(y)\le\epsilon_F$.

A combination of the preceding techniques is used for the problem

$$\textit{minimize} \quad f(x) \ -\textit{all}\ \ x$$

$$\textit{satisfying} \quad Ax \le b,\ \ x^L \le x \le x^U.$$

In this case, all trial points satisfy the simple bounds exactly, and the general linear constraints to within ϵ_F.

2.3. Exact penalty methods for convex constrained problems

The convex minimization problem

$$\textit{minimize} \quad f(x) \ -\textit{all} \ x \tag{25a}$$

$$\textit{satisfying} \quad F_j(x) \le 0 \ \text{ for } \ j=1,...,m_I, \tag{25b}$$

$$F_j(x) = 0 \ \text{ for } \ j=m_I+1,...,m_I+m_E, \tag{25c}$$

where the functions f and F_j, $j=1,...,m_I$, are convex and the functions F_j, $j=m_I+1,...,m_I+m_E$, are affine (linear), may be solved in NOA2 by the unconstrained minimization of the *exact penalty function*

$$e(x;\rho) = f(x) + \rho \, F_+(x), \tag{26}$$

where $\rho>0$ is a *fixed penalty* coefficient, and the constraint violation is measured by

$$F_+(x) = \max\{F(x),0\},$$

$$F(x) = \max\{F_j(x)\colon j=1,...,m_I, \ |F_j(x)|\colon j=m_I+1,...,m_I+m_E\}.$$

Each solution x_ρ to the problem

$$\textit{minimize} \quad e(x;\rho) \quad -\textit{all} \ x \ \textit{in} \ R^n \tag{27}$$

solves (25) if it is feasible $(F(x_\rho)\le 0)$. This holds if ρ is sufficiently large, (25) has a solution and its constraints satisfy the generalized Slater constraint qualification, i.e. for some x_S

$$F_j(x_S)<0, \quad j=1,...,m_I, \ F_j(x_S)=0, \ j=m_I+1,...,m_I+m_E.$$

The methods with a fixed penalty coefficient require the user to specify a sufficiently large ρ. For well-scaled problems one may usually choose ρ in the interval [10,100]. If ρ is too small, (27) need not be equivalent to (25), and the algorithm may diverge when the penalty function has no finite minimum. On the other hand, too large ρ hinders the minimization of the penalty function, which becomes ill-conditioned. (If ρ is large, the algorithm must hug the boundary of the feasible set.)

The first method in NOA2 solves (26) by one of the algorithms for unconstrained minimization. At the k-th iteration, a polyhedral approximation $\hat{e}^k(\cdot;\rho)$ to $e(\cdot;\rho)$ is constructed from the past linearizations of $e(\cdot;\rho)$ (see (5) and (6)). (These linearizations are calculated as in (5) from subgradients of the functions of (25), which are evaluated by the user's subroutine.) The k-th search direction d^k is chosen to

$$\textit{minimize} \quad \hat{e}^k(x^k+d;\rho) + u\,|d|^2/2 \ -\textit{all} \ d \tag{28}$$

(see (16)). Termination occurs if

$$|v^k| \le \epsilon_S \,(1+|e(x^k;\rho)|) \tag{29a}$$

and

$$F(x^k) \le \epsilon_F, \tag{29b}$$

where ϵ_S and ϵ_F are positive final accuracy and feasibility tolerances, provided by the user, whereas v^k is a dual estimate of the predicted descent $\hat{e}^k(x^k+d^k;\rho)-e(x^k;\rho)$, which satisfies the optimality estimate

$$f(x^k) \leq f(x^*) + u\, |v^k|^{1/2}\, |x^* - x^k| - v^k, \tag{30}$$

where x^* is a solution to (25). This method does not exploit the specific structure of $e(\cdot;\rho)$.

The second method exploits the additive structure of $e(\cdot;\rho)$ by constructing separate polyhedral approximations $\hat{f}^k$ and $\hat{F}^k$ to the objective f and constraint function F. Thus the method may use a more accurate polyhedral approximation to $e(\cdot;\rho)$

$$\hat{e}^k(x;\rho) = \hat{f}^k(x) + \rho \max\{\hat{F}^k(x), 0\} \tag{31}$$

in the search direction finding subproblem (28), which usually enhances faster convergence.

Both methods may be allowed to choose the penalty coefficient automatically during the calculations (Kiwiel, 1985d). Then at the k-th iteration we set $\rho=\rho^k$ in (28) and (31). The initial ρ^1 may be specified by the user. The penalty coefficient is increased only if x^k is an approximate solution to (27) (i.e. x^k minimizes $e(\cdot;\rho^k)$ to within some positive tolerance δ^k), but it is significantly infeasible (i.e. $F(x^k)$ is "large"). The specific rule for updating ρ^k is

$$\text{if } -v^k \geq \delta^k \textit{ or } F(x^k) \leq -v^k \textit{ set } \rho^{k+1}=\rho^k \text{ and } \delta^{k+1}=\delta^k; \tag{32a}$$

$$\textit{otherwise set } \rho^{k+1}=\rho^k \text{ and } \delta^{k+1}=c_v\delta^k, \tag{32b}$$

where $c_\rho>1$ and $c_v\in(0,1)$ are parameters that increase the penalty and decrease the accuracy tolerance of unconstrained minimization δ^k; $\delta^1=|v^1|$. Usually one may use $\rho^1=10$, $c_\rho=2$ *or* $c_\rho=10$, and $c_v=0.1$. Larger values of c_ρ and c_v enable a faster growth of the penalty coefficient at earlier iterations, if the initial ρ^1 was too small. On the other hand, very large values of penalty coefficients slow down convergence.

When employing the exact penalty methods, the user should place sensible upper and lower bounds on all variables. If the box defined by such bounds is not too large, the penalty coefficient will quickly reach a suitable value and then will stay constant. Moreover, box constraints ensure the existence of a solution and prevent the algorithm from diverging.

We may add that the automatic choice of the penalty coefficient may produce a very large value of ρ^k. The methods terminate at the k-th iteration if $\rho^{k+1}>1/\epsilon_M$, where ϵ_M is the relative machine precision. Such abnormal termination may indicate that the constraints are not regular (e.g. are inconsistent), or that they are ill-scaled.

In the current version of NOA2 additional general linear constraints $Ax\leq b$ can be handled only by the first method that does not exploit the structure of the penalty function.

2.4. The constraint linearization method

The convex constrained problem

$$\textit{minimize} \quad f(x), \quad \textit{subject} \quad F(x) \leq 0 \tag{33}$$

with a convex f and a convex F satisfying the Slater condition ($F(x_S)<0$ for some x_S) may be solved in NOA2 by the constraint linearization method (Kiwiel, 1987), which is frequently more efficient than the algorithms of the preceding section.

At the k-th iteration the algorithm uses polyhedral approximations $\hat{f}^k$ and $\hat{F}^k$ to f and F in the search direction finding subproblem

$$\textit{minimize} \quad \hat{f}^k(x^k+d) + u|d|^2/2 \tag{34a}$$

$$\textit{subject} \quad \hat{F}^k(x^k+d) \leq 0, \tag{34b}$$

where $u>0$ is the weight of the regularizing quadratic term. Its solution d^k is an approximate descent direction for the exact penalty function (26), provided that the penalty parameter $\rho=\rho^k$ is greater than the Lagrange multiplier $\tilde{\rho}^k$ of the constraint (34b). Hence the algorithm sets $\rho^k=\rho^{k-1}$ if $\tilde{\rho}^k\leq\rho^{k-1}$; otherwise

$$\rho^k = \max\{\tilde{\rho}^k, \ c_\rho\rho^{k-1}\}, \tag{35}$$

where $c_\rho>1$ is a user-specified parameter (usually $c_\rho=2$), and $\rho^0=0$. With $\hat{e}^k(\cdot;\rho^k)$ given by (31), the predicted descent

$$v^k = \hat{e}^k(x^k+d^k;\rho^k) - e(x^k;\rho^k)$$

satisfies the optimality estimate (30), which justifies the termination test (29). The line search from x^k along d^k uses the rules of Section 2.1, applied to $e(\cdot;\rho^k)$.

Subproblem (34) is solved by finding d^k to

$$\textit{minimize} \quad \hat{f}^k(x^k+d) + u|d|^2/2 + c \ \textit{max}\{\hat{F}^k(x^k+d),0\}, \tag{36}$$

where the penalty coefficient c is chosen as in Section 2. (cf. (23)). Abnormal termination with $c>1/\epsilon_M$ may indicate violation of the Slater constraint qualification, ill-scaling of the constraints, or that the infeasibility tolerance e_F is too tight. These factors also may enforce termination due to $\rho^k>1/\epsilon_M$.

Additional linear constraints

$$Ax \leq b, \quad x^L \leq x \leq x^U$$

are handled by the techniques of Section 2.2. In this case the Slater constraint qualification reads: $F(x_S)<0$, $Ax \leq b$ and $x^L \leq x_S \leq x^U$ for some x_S. Once again, we stress that the presence of box constraints may be crucial to the algorithm's convergence.

2.5. Feasible point methods for convex problems

The convex constrained problem (33) may be solved in NOA2 by the feasible point method (Kiwiel, 1985a), which uses polyhedral approximations $\hat{f}^k$ and $\hat{F}^k$ of f and F in the search direction finding subproblem

$$\textit{minimize} \quad \hat{H}^k(x^k+d) + u|d|^2/2 \ -\textit{all} \ \ d, \tag{37}$$

where $u>0$ is a scaling parameter, whereas

$$\hat{H}^k(x) = \textit{max}\{\ \hat{f}^k(x)-f(x^k),\hat{F}^k(x)\}$$

is the k-th polyhedral approximation to the improvement function

$$H(x;x^k) = \textit{max}\{\ f(x)-f(x^k),F(x)\} \ \text{ for } \textit{all} \ \ x.$$

Thus, if $F(x^k)<0$, we wish to find a feasible $(\hat{F}^k(x^k+d^k)<0\ )$ direction of descent $(\hat{f}^k(x^k+d^k)<f(x^k)\ )$, whereas for $F(x^k)>0$, d^k should be a descent direction for F at x^k $(\hat{F}^k(x^k+d^k)<0)$, since then we would like to decrease the constraint violation.

The algorithm runs in two phases. At phase I successive points x^k are infeasible, and the line search rules of Section 2.1 are applied to F. Finding a feasible x^k starts phase II, in which the line search rules are augmented to ensure feasibility of successive iterates. Of course, phase I will be omitted if the initial point x^1 is feasible.

The algorithm requires the Slater constraint qualification ($F(x_S)<0$ for some x^S); otherwise, it may terminate at a point x^k that is an approximate minimizer of F.

The algorithm is, in general, more reliable than the exact penalty methods of Sections 2.3 and 2.4, because it does not need to choose penalty coefficients. Unfortunately, its convergence may be slower, since it cannot approach the boundary of the feasible set at a fast rate.

Additional linear constraints are handled as in Section 2.2.

2.6. Methods for nonconvex problems

Minimization problems with nonconvex objectives and constraints are solved in NOA2 by natural extensions (Kiwiel, 1985a, 1985b, 1986a, 1986c) of the methods for convex minimization described in the preceding sections. Except for the constraint linearization method of Section 2.4, each method has two extensions, which differ in the treatment of nonconvexity. The methods use either subgradient locality measures, or subgradient deletion rules for localizing the past subgradient information. Advantages and drawbacks of the two approaches depend on specific properties of a given problem.

For simplicity, let us consider the unconstrained problem of minimizing a locally Lipschitz continuous function f, for which we can calculate the linearization

$$\bar{f}(x;y) = f(y) + <g_f(y),x-y>$$

by evaluating f and its subgradient g_f at each y. At the k-th iteration, several such linearizations computed at trial points y^j, $j\in J_f^k$, are used in the following polyhedral approximation to f around the current iterate x^k

$$\hat{f}^k(x)=f(x^k)+max\{-\alpha_f(x^k,y^j)+<g_f(y^j),x-x^k>:\ j\in J_f^k\},\tag{38}$$

where the *subgradient locality measures*

$$\alpha_f(x^k,y^j) = max\ \{\ |f(x^k)-\bar{f}(x^k,y^j)|,\ \ \gamma_s|x^k-y^j|^2\}\tag{39}$$

with a parameter $\gamma_s\geq0$ indicate how much the subgradient $g_f(y^j)$ differs from being a subgradient of f at x^k. Observe that in the convex case with $\gamma_s=0$ the approximation (38) reduces to the previously used form (6) (cf. (4)). More generally, for $\gamma_s>0$ the subgradients with relatively large locality measures cannot be active in $\hat{f}^k$ in the neighborhood of x^k. Thus even in the nonconvex case $\hat{f}^k$ may be a good local approximation to f; provided that it is based on sufficiently local subgradients. This justifies the use of $\hat{f}^k$ in the search direction finding subproblems of the preceding sections (cf. (7), (16), (19), (21), (28), (37)).

Ideally, the value of the locality parameter γ_s should reflect the degree of nonconvexity of f. Of course, for convex f the best value is $\gamma_s=0$. Larger values of γ_s decrease the influence of nonlocal subgradient information on the search direction finding. This, for instance, prevents the algorithm from concluding that x^k is optimal because $\hat{f}^k$ indicates that f has no descent direction at x^k. On the other hand, a large value of γ_s may cause that after a serious step all the past subgradients will be considered as nonlocal at the search direction finding. Then the algorithm will be forced to accumulate local subgradients by performing many null steps with expensive line searches.

In the strategy described so far the influence of a subgradient on $\hat{f}^k$ decreases "smoothly" when this subgradient becomes less local. More drastic is the ***subgradient deletion strategy*** , which simply drops the nonlocal past subgradients from $\hat{f}^k$. In this case, we set $\gamma_s=0$ in (39) and define the ***locality radius***

$$a^k = \max\{\ |x^k-y^j| : j\in J_f^k\ \} \tag{40}$$

of the ball around x^k from which the past subgradients were collected. As before, the approximation $\hat{f}^k$ is used to generate a search direction d^k. A locality reset of the approximation occurs if

$$|d^k| \leq m_a\ a^k, \tag{41}$$

where m_a is a positive parameter. This involves dropping from J_f^k an index j with the largest value of $|x^k-y^j|$, i.e. the most nonlocal subgradient is dropped so as to decrease the locality radius a^k. If the next d^k satisfies (41), another reset is made, etc. Thus resets decrease the locality radius until it is comparable with the length of the search direction $|d^k|$.

Dropping the j-th subgradient corresponds to replacing $\alpha_f(x^k,y^j)$ in (38) by a large number. Moreover, the frequency of resets is proportional to the value of m_a in the test (41). Therefore, our preceding remarks on the choice of γ_s are relevant to the selection of m_a.

In practice one may use $\gamma_s=1$ and $m_a=0.1$, increasing them to $\gamma_s=10$ and $m_a=0.5$ for strong nonconvexities.

Both strategies use line searches similar to that of Section 2.1. Additionally, the subgradient resetting strategy requires that a null step ($x^{k+1}=x^k$) should produce a trial point y^{k+1} close to x^k in the sense that $|y^{k+1}-x^k|$ is of order a^k. Since $|y^{k+1}-x^k|=t_R^k|d^k|$, the right stepsize t_R^k should be sufficiently small. This can be ensured either by testing progressively smaller initial trial stepsizes, or by introducing the direct requirement

$$|y^{k+1} - x^k| \leq c_d\ a^k \quad \text{if} \quad x^{k+1}=x^k,$$

where $c_d\in[0.1,0.5]$ is a parameter, e.g. $c_d=m_a$.

References:

Clarke, F.H.(1983). *Optimization and Nonsmooth Analysis.* Wiley Interscience, New York.

Kiwiel, K.C.(1985a). *Methods of Descent for Nondifferentiable Optimization.* Springer-Verlag, Berlin.

Kiwiel, K.C.(1985b). *A linearization algorithm for nonsmooth minimization.* Mathematics of Operations Research 10, 185-194.

Kiwiel, K.C.(1985c). *An algorithm for linearly constrained convex nondifferentiable minimization problems.* Journal of Mathematical Analysis and Applications 105, 452-465.

Kiwiel, K.C.(1985d). *An exact penalty function algorithm for nonsmooth convex constrained minimization problems.* IMA Journal of Numerical Analysis 5, 111-119.

Kiwiel, K.C.(1986a). *An aggregate subgradient method for nonsmooth and nonconvex minimization.* Journal of Computational and Applied Mathematics 14, 391-400.

Kiwiel, K.C.(1986b). *A method for solving certain quadratic programming problems arising in nonsmooth optimization.* IMA Journal of Numerical Analysis 6, 137-152.

Kiwiel, K.C.(1986c). *A method of linearizations for linearly constrained nonconvex nonsmooth minimization.* Mathematical Programming 34, 175-187.

Kiwiel, K.C.(1987). *A constraint linearization method for nondifferentiable convex minimization.* Numerische Mathematik (to appear).

Lemarechal, C.(1978). *Nonsmooth optimization and descent methods.* Report RR-78-4, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Mifflin, R.(1982). *A modification and an extension of Lemarechal's algorithm for nonsmooth minimization.* Mathematical Programming Study 17, 77-90.

Implicit Utility Function and Pairwise Comparisons

Janusz Majchrzak

Systems Research Institute, Polish Academy of Sciences.

1. INTRODUCTION

A new approach for multicriteria decision making is briefly presented here exploiting the pairwise comparisons of alternatives and assuming the existence of an implicit utility function of the decision maker. We plan to extend DISCRET along this direction in the future.

The basic problem in the area of the interactive decision support systems is the extraction and utilization of the preferences of decision maker (DM). A rather large number of approaches have been developed during last decade. This chapter reports some basic ideas of a new approach, which seems to be promising because of its conceptual and methodological simplicity. The presented approach is based on the pairwise comparisons of alternatives and linear approximations of the DM's utility function. Since the approach is at an early stage of development and its several aspects still have to be investigated, just some basic ideas and motivations will be presented.

The basic feature of the approach is that th DM is not forced to compare pairs of alternatives which are presented to him but he chooses himself a subset of alternatives to be evaluated. An underlying quasiconvex DM utility function is assumed.

2. MOTIVATIONS

Let us consider the following multicriteria decision making problem. A decision maker (a person or an institution) wants to buy a new car and has some difficulties in choosing from the variety of models available on the market. He is not an expert in cars and he knows just a few models: his old car and those possessed by his friends and relatives. So, all he is able to say about his preferences is a number of statements concerning cars he knows, like for example:

VW Golf is preferred to Opel Kadett,

Fiat Uno is preferred to Peugeot 205, etc.

He refuses to compare cars he doesn't know or to supply any other kind of information about them. The reference point approach might be adopted in this case, but what if the DM would not be satisfied with the result?

The task can be formulated as follows. A relatively small number of pairwise comparisons of alternatives is available. What can be said about the DM's preferences on the basis of this small amount of information and what can be said about the quality of that information ? Note that a statement: "a cheap good car is preferred to an expensive bad car" is a rather low quality information since, once price and performance have been established as criteria, this is an obvious statement. The DM should be informed about the quality of the alternatives evaluation he had made. Also his inconsistencies should be discovered.

3. BASIC IDEAS

Let F be the space of m criteria, $\Lambda = R^m_+$ be the domination cone and let $Q \subset F$ be the set of feasible alternatives. We will assume that there exist an underlying implicit quasiconvex utility function $U: \rightarrow R$ behind the DM's preferences. The DM need not recognize it existence; however, we will assume that whenever he decides that alternative $b \in Q$ is preferred to alternative $a \in Q$, it is equivalent to $U(b) > U(a)$.

The DM's utility function U is in general a nonlinear function of criteria. Identifying such function usually requires large amount of data and a significant computational effort. Therefore, keeping nonlinearity of U in mind, we shall restrict ourselves to a set of linear approximations of U only.

Suppose that k pairs of alternatives were compared by the DM:

b_i is preferred to a_i, $a_i,b_i \in Q$, $i = 1,...,k$

This set of data may be considered as a set W of k vectors in the criteria space F, pointing from a less preferred alternative a_i to a more preferred alternative b_i.

$$W = \{ w_i : w_i = [a_i,b_i] , a_i,b_i \in Q , i=1,...,k \}$$

Let us also consider the set V of normalized vectors $w_i \in W$:

$$V = \{ v_i : v_i = \frac{w_i}{\| w_i \|} \quad i = 1,\ldots,k \}$$

Each of the vectors v_i represents a direction of improvement in the space of criteria of the function $U(f)$. Hence, the cone C spanned by vectors $v_i \in V$ is the cone of improvement for $U(f)$ and can be defined as:

$$C = \{ \sum_{i=1}^{i=k} \alpha_i v_i : \alpha_i \in R_+ , v_i \in V\}$$

The cone C^* is the corresponding polar cone and can be defined as:

$$C^* = \{ y_i \in F : <y,v_i> \geq 0 , v_i \in V, i=1,2,...,k\}$$

Both cones C and C^* can be expressed by their generators. The set of cone generators is the minimal subset of vectors belonging to that cone that still span the cone. The generators of cones C and C^* will be denoted by c and c^*, respectively.

$$C = \{ \sum_j \alpha_j c_j , \alpha_j \in R_+ , c_j \in \tilde{C} \}$$

$$C^* = \{ \sum_j \alpha_j c_j^* , \alpha_j \in R_+ , c_j^* \in \tilde{C}^* \}$$

where $\tilde{C}$ and $\tilde{C}^*$ are corresponding generator sets.

Let us return to the pairwise comparisons. Since we shall consider linear approximations of the utility function, for the sake of presentation simplicity, assume that U is linear. If the DM has decided that alternative $b \in Q$ is preferred to alternative $a \in Q$, then $U(b) > U(a)$. It is clear that $<v,u> > 0$, where $v = [a,b]$, and u is a vector normal to hyperplanes $U(f) = const$. Hence, the vector u is contained in cone C^*.

From the above analysis it follows that an accurate determination of the vector u normal to the hyperplanes of U will be possible only in the case when the cone C^* is spanned by a single vector (namely u). In this case the DM's utility function (or rather actually its linear approximation only) has been obtained and we can easily calculate the

DM's most preferred solution by minimizing U over the set Q.

In general, because of obvious reasons, the cone C will be smaller than a halfspace and its polar cone C^* will have a nonempty (relative) interior. In such a case, each of the vectors contained in C^* may appear to be the vector u. Fortunately, we can restrict ourselves to the generators c^* of the cone C^* only. Considering each c_j^* to be the vector u (minimizing linear function based on c_j^*) one can obtain a set of $q_j \in Q$ being the linear approximation minimizers of DM's utility function. These elements q_j define a subset $S \subset Q$ of nondominated elements of Q in which the DM's most preferred alternative (minimizer of iU) is contained.

As it can be seen now, our approach does not pretend to determine the DM's most preferred solution exactly. It will rather tend to find a domain in which it is contained. The more information about DM's preferences is contained in alternatives pairwise comparisons supplied, the smaller this domain will be. Besides, also a good candidate for the most preferred solution may be presented to the DM. It can be obtained a kind of average vector for the cone C^*: a sum of c_j^*, a sum of v_i, a gravity center of v_i, etc. The author's favorite method for the candidate selection is the calculation of the minimal (Euclidean) norm element from the convex hull spanned by the cone C^* generators c_j^*. This technique based on the method of P. Wolfe [1] appeared to be very useful in our approach, serving also for some other purposes. Let us denote the minimal norm element from the convex hull spanned by the set V of vectors v as

$$z = MNECH(C^*)$$

The minimizer of the linear function based on vector z will be chosen as the candidate for the DM's most preferred solution.

4. SOME DETAILS

In this chapter, we shall discuss the basic cases that can occur for different sets of pairwise comparisons of alternatives supplied by the DM.

Case 1. Cone C is a halfspace of F and $\|z\| = 0$.

As it has been already mentioned, in this case the linear approximation of the DM's utility function is defined by the vector u normal to the halfspace spanned by C. The DM's most preferred solution may be found by the optimization of the linear function based on u.

Case 2. Cone C is not a halfspace of F and $\|z\| = 0$.

Since the DM's utility function is assumed to be quasiconvex, the set V of pairwise comparisons supplied by the DM is inconsistent. Conflicting elements should be selected from the set V and presented to the DM. They are those elements which spann a convex hull containing zero and hence cause $\|z\| = 0$. Their selection is automatic during the calculation of the element z.

Case 3. Cone C is contained in a halfspace of F, it contains the domination cone Λ and $\|z\| \geq 0$.

This is the basic case. After the set of linear functions based on vectors c^* optimization, a subset of nondominated elements of set Q will be obtained. This subset is defined by the set of linear approximation optimizers of the utility function. A candidate for the DM's most preferred solution will be found by optimizing over the set Q the linear function based on vector z. Notice that if the number of supplied pairwise comparisons is small (too small to spann a non-degenerate cone C), then generators of the domination cone Λ

can be added to the set V.

Case 4. Cone C is contained in the domination cone Λ and $\|z\| \geq 0$. This is the case of a low quality of information contained in pairwise comparisons of alternatives supplied by the DM (and corresponds to statements like: "a good cheap car is preferred to an expensive bad car"). The DM should be informed about this fact and perhaps he will be able to give some more restrictive statements. If he refuses for some reasons, we cannot proceed along the Case 3 line. However, instead of of considering the supplied information as being of a discriminative type we can treat it as an instructive type information. Each of the vectors $v \in V$ can be treated now as an approximation of the DM's improvement direction or his utility gradient approximation. Hence, we can proceed just like in Case 3, taking the cone C instead of C^* into consideration. Of course the DM should be aware of the new interpretation of the information he has supplied.

Cases 3 and 4 can be distinguished a priori by checking whether $C \supset \Lambda$ or $C \subset \Lambda$, respectively.

5. CONCLUDING REMARKS

If the DM is able to supply a large amount of results on alternatives evaluations, then a technique similar to one presented in [2] should be used in order to eliminate dominated alternatives from further considerations. If it is not the case, the DISCRET package methodology should be applied. Actually, the presented approach is planned to be included into the DISCRET framework.

Several aspects of the presented approach are still to be further investigated. The main one is how to select a small sample of such alternatives that their evaluation by the DM may result in significant improvement of an approximation of DM's preferences.

REFERENCES

[1] P.Wolfe,"Finding the Nearest Point in a Polytope", *Mathematical Programming, Vol.11, pp 128-149, 1975.*

[2] M.Koksalan, M.H.Karwan, S.Zionts, "An Improved Method for Solving Multiple Criteria Problems Involving Discrete Alternatives", *IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-14, No.1, pp.24-34, 1984.*

[3] J.Majchrzak,"Methodological Guide to the Decision Support System DISCRET for Discrete Alternatives Problems", *in this volume.*

Safety Principle in Multiobjective Decision Support in the Decision Space Defined by Availability of Resources

Henryk Gorecki, A.M. Skulimowski

Institute of Automatic Control
Academy of Mining and Metallurgy, Krakow.

ABSTRACT

We consider the situation where a decision-maker in a multicriteria optimization problem must follow additional constraints in the criteria space defined by availability of resources. The set defined by such constraints - called demanded set - is assumed to be uncertain as a result of a priori experts estimations. The analysis of numerous real-life situations showed that a method of looking for a non-dominated solution on the so-called skeleton allows to find a solution maximally safe with respect to the random perturbations of the demanded set. We formulate a maximal safety principle as a requirement that the expected value of distance from the solution chosen to the boundary of the demanded set were maximal. Then we prove that the search executed on the skeleton curve satisfies this principle for a class of demanded sets defined by aspiration levels.

1. INTRODUCTION

The choice of a compromise solution fulfilling additional conditions with regard to its location in the criteria space is essential in numerous real-life multiple criteria optimization problems. For instance, the choice of a technological process from many variants proposed by experts, taking into account the total cost of investment and the minimal necessary time to start the production, is often based on the analysis of upper and lower bounds for values of the above criteria, (Gorecki, 1981). Such bounds are usually not strict; they are called aspiration levels and are assumed to be imposed independently by experts or the decision-maker after the formulation of the problem, therefore serving as an additional information for selecting the compromise solution.

The nature of aspiration levels is often uncertain and the arising set of demanded values of criteria may be represented as random or fuzzy set. When selecting a compromise solution, the decision-maker is obliged to take into account the possibility of unexpected change of aspiration levels using an uncertainty handling technique. For the case where the demanded set is defined by two aspiration levels such a method has been proposed by Gorecki (1981). In his approach the search for a non-dominated solution has been executed on a line which joins the aspiration levels, and lies inside of so-called skeleton of the demanded set. An outline of the skeleton method may be found in Gorecki (1981) and Wiecek (1984). The numerical implementation of this method has then been developed by Gorecki et al. (1982, 1985). Here, we will present some of its theoretical foundations.

Throughout this paper we will assume that the set defined by the lower and upper aspiration levels, called demanded set, and the attainable set of the criteria values have a non-empty intersection. Then we will analyse the problem of selecting a non-dominated compromise solution from this intersection which is - in some sense - most reliable to the changes of the demanded set. Namely, we look for a problem solution on a specific class of lines called the ordinal skeleton curves of the demanded set. The solution thus obtained will possess the property that the expected value of the distance from the boundary of the demanded set is maximal, or equivalently, that the probability of remaining within the demanded set – which boundary changes according to some random rules – is maximal.

In this paper we will concentrate our attention on the particular case of the criteria space constraints, namely on the sets defined by aspiration levels of the form

$$Q := (q_1-\Theta)\cap(q_2+\Theta) \tag{1}$$

where: q_1 and q_2 are the aspiration levels for criteria, denoting the minimal admissible, and the most desired values of the criteria, respectively, and Θ is the positive cone of the partial order in the criteria space. Usually $\Theta=R^N_+$, and

$$Q = \prod_{i=1}^{N}[q_{1i};q_{2i}] \tag{2}$$

where $q_1=(q_{11},...,q_{1N})$ and $q_2=(q_{21},...,q_{2N})$, $q_{1i}\le q_{2i}$ for $1\le i\le N$ and the product of intervals is understood in the Cartesian sense.

2. PROBLEM FORMULATION

Let us consider multicriteria minimization problem /MMP/

$$(F{:}\,U\to R^N)\to\min(\Theta) \tag{3}$$

where $F=(F_1,F_2,...,F_N)$ is the vector objective function, U is a subset of the decision space, and Θ is a closed, convex and pointed cone defining the partial order $\le_\Theta$ in the criteria space R^N. We assume that the set U and the function F are convex, therefore the attainable set $F(U)$ is also convex.

The solution u to the problem will be called *non-dominated* iff

$$(F(u)-\Theta)\cap F(U)=\{F(u)\} \tag{4}$$

The set of non-dominated decision will be called the Pareto set and denoted by $P(U,F,\Theta)$ while the corresponding set of non-dominated valuations

$$FP(U,\Theta):=F(P(U,F,\Theta)) \tag{5}$$

will be called the *compromise set* . We will also use the notation $P(V,\Theta):=P(V,id_E,\Theta)$ whenever $V\subset E$.

Moreover, we assume that in the criteria space two points are distinguished, $q_1:=(q_{11},...,q_{1N})$ and $q_2:=(q_{21},q_{22},...,q_{2N})$ such that $q_2\le_\Theta q_1$. The points q_1 and q_2 will be called the upper, and the lower aspiration levels for the problem (3), respectively.

The aspiration levels are set up by experts independently from the base problem formulation and define so-called demanded set Q for the values of the criteria (cf. formula (1)). We will assume that q_1 is attainable and that

$$P((q_1-\Theta)\cap F(U),\Theta)=(q_1-\Theta)\cap FP(U,\Theta) \tag{6}$$

On the contrary, q_2 is assumed to be unattainable strictly dominating point for the attainable set $F(U)$, i.e.

$$FP(U,\Theta) \cap (q_2+\Theta) \neq \phi \tag{7}$$

and

$$P(q_1+\Theta) \cap F(U),\Theta)=(q_1+\Theta) \cap FP(U,\Theta) \tag{8}$$

(cf. Skulimowski (1986a)).

Another additional assumption which will be used at this stage of problem solution is that there exist reasonable estimates of the scale coefficients for each scalar criteria $F_1,...,F_N$. This will enable us to measure the distance of the criteria inside the demanded set basing on locally comparable units of the coordinates of the criterion function. A method of deriving locally comparable units has been proposed by Gorecki (1981) who used the differences between the coordinates of the barycenter of Q and q_1 as the relative units of criteria.

Since this kind of information imposes certain knowledge of the trade-offs between criteria which in our model are uncertain, in the real-life applications we will repeat the execution of the algorithm described in the following section interactively, with the slightly varying values of the scale coefficients.

The demanded set Q plays the role of additional constraints imposed on the solution to the MMP. At this stage we will assume that every non-dominated solution found inside Q is admissible for the decision maker. However, the estimates of q_1 and q_2 are usually uncertain and the satisfactory solution to the problem is the one which lies inside of the actual demanded set Q_η perturbed by a random factor η. To maximize the probability of $u_{opt} \in Q_\eta$ we will define a special class of algorithms of the line-search for a non-dominated solution to MMP inside of Q.

3. THE SEARCH FOR A NON-DOMINATED SOLUTION ON A CURVE

The idea of the algorithm of finding a compromise non-dominated solution presented below consists in replacing the original MMP (3) by a search for a non-dominated solution belonging to a curve g which lies inside the demanded set Q. If Q is defined by (1), g begins at an attainable reference point q_1 and ends at an unattainable dominating one, q_2 i.e. $g(0)=q_1$ and $g(1)=q_2$. The solution thus found belongs to the intersection of $FP(U,\Theta)$ and $g^*:=g([0;1])$, and is non-dominated provided that the set $FP(U,\Theta)$ divides the demanded set into two parts. The latter condition is fulfilled e.g. when $F(U)$ is convex and (6) is satisfied.

The algorithm of the search.

The choice of the curve g is based on the analysis of the specific properties of elements of g^*. Consequently we will consider the curves which satisfy the maximal safety principle, i.e. those for which the probability that the compromise solution chosen will remain within the randomly perturbed demanded set is maximal.

This may be achieved by selecting the curve maximizing the mean value of distance from the boundary of the demanded set. Considering moreover the fact that some criteria may turn out to be redundant leads us to choosing the so-called ordinal skeleton curve (Gorecki (1981), Wiecek (1984)) as the curve the search should be expected on.

The general algorithm of the search on a curve g may be sketched as follows:

```
Step 0 : selection of $g$, the choice of the algorithm $A$ of detection of a non-dominated
point $p$ on $g^*$,
        $f_o := q_2, \; i := 1, \; r_o := 1$
Step 1 :
        $f_i = A(f_{i-1}, r_{i-1})$,
Step 2 : check whether $f_i$ is attainable; set $r_i := 1$ if $f_i$ is attainable,
otherwise $r_i := 0$,
Step 3 :
        $e_i := \| f_i - f_{i-1} \|$
if $r_i \leq r_{i-1}$ and $e_i < e_0$
then
        $p := \frac{f_i + f_{i-1}}{2}$
    stop
    else $i := i+1$, go to 1.
```

The result of an execution of the algorithm is a non-dominated solution p. The Pareto-optimality of p is an immediate consequence of the assumption that q_1 and q_2 are separated by the non-dominated surface $FP(U,\Theta)$. The uniqueness is assured if g is a linearly ordered subset of Q which will be assumed further on. The maximal safety of p will be discussed in the following section.

In selecting a curve g so that safety principle is satisfied, a crucial role is played by the norm in the criteria space since it determines the value of the distance of the solution chosen from the exterior (or, equivalently, boundary) of Q. On the other hand, choice of distance influences the properties of the probability distribution of finding a non-dominated point along a curve. The justification of the choice of the norms l_1 or l_∞ in the criteria space is contained in the following subsection.

The algorithm is assumed to possess the following properties

a) $A(f,r) \in g^*$ whenever $f \in g^*$

b) $\| f_{i+1} - f_i \| < \| f_i - f_{i-1} \|$ for $i > 1$

c) the assumed number of iterations of A depends only on the value of $||q_1 - q_2||$, not on the shape of g^*.

To check whether a point F_i belonging to g^* is attainable one should examine the existence of solutions to the equation

$$f_i = F(u)$$

In convex cases this may be done as proposed by Wiecek (1984).

The value of e_0 must be sufficiently small to assure the accuracy of the method. The recommended value which proved to be adequate in numerical experiments is

$$e_0 := 10^{-4} \min\{\rho_i(Q); \; 1 \leq i \leq N\}$$

where $\rho_i(Q)$ is the diameter of the projection of Q on the i-th axis in the criteria space.

The choice of a distance in the criteria space

We will start this section from the following definition:

Definition 1 : A curve $g:[0,1]\rightarrow E$ is linearly ordered iff

$$\textit{for each } t_1,t_2\in[0,1]:\ t_1\leq t_2 \Rightarrow\ g(t_1)\leq_\Theta g(t_2) \tag{9}$$

where $\leq_\Theta$ is the partial order in E. The set of all linearly ordered curves linking the points x and y will be denoted by $L(x,y)$.

Further on we will require that the following property of the line-search for a non-dominated solution, imposed by the choice of the class searching algorithms, is satisfied.

Assumption 1 . Let x and y be two elements of the demanded set Q such that $x\leq_\Theta y$. Then the probability of finding a non-dominated point on a linearly ordered curve connecting x and y is constant and does not depend on the choice of this curve.

On the other hand we may require that the search on a curve gives better results when the curve is longer which can be formalized as

Assumption 2. The probability of detecting a non-dominated point on a curve linking two points is proportional to the length of this curve.

Consequently, the Assumptions 1 and 2 imply that all linearly ordered curves linking two fixed points in the criteria space should have the same length. The above requirements imply the limitations in the choice of the distance and the derived differential form (element of distance) which defines the length of the curve.

It is easy to see that the following statement is true.

Proposition 1 : The Assumptions 1 and 2 are fulfilled by the length of the curve generated by the l_1 or l_∞ norm, i.e. by

$$\lambda_1(g):=\int_g dx(l_1)=\int_{[0,1]}(\sum_{i=1}^{n}|g_i(t)|)dt \tag{10}$$

where $g=(g_1,...,g_N)$ is the curve considered, and $x(l_1)$ is the element of length associated to the L_1 norm. The length of g for l_∞ norm is defined similarly to (10).

Proof of the Proposition 1 is given in Gorecki and Skulimowski (1986b), i.e. we prove that

$$\textit{for each } x,y\in Q,\ a,b\in L(x,y):\ \lambda_1(a)=\lambda_1(b) \tag{11}$$

Observe that only certain distances in R^N satisfy the above requirement (11), e.g. it is not fulfilled by the Euclidean distance.

The Assumptions 1 and 2, and the subsequent distance in the criteria space are in compliance with the assumption about the class of algorithms applied for looking for a non-dominated point on a curve, namely we will assume that these algorithms satisfy:

Assumption 3 . The a priori imposed maximal number of steps of an algorithm of detecting a non-dominated point on a curve g connecting the elements x and y of the criteria space does not depend on the choice of g but on the differences between coordinates of x and y. In particular, it may be defined as

$$\max\left\{Ent(\frac{y_i-x_i}{s_i}):1\leq i\leq N\right\},$$

where $Ent(r)$, $r \in R$, is the smallest integer exceeding r, and s_i, $1 \leq i \leq N$, are desired steps of quantification of criteria.

4. THE SAFETY PRINCIPLE

We will start this section with some basic definitions and properties. Let us recall that the *demanded set* Q is a closed and connected subset of the criteria space such that

$$FP(U,\Theta) \cap Q \neq \phi \tag{12}$$

Remark 1 : When (12) is not satisfied but Q contains some dominating points for the attainable set then Q may be regarded as a target set and a distance scalarization technique may be applied (Skulimowski, 1985a).

Further on we will restrict our consideration to the case where the demanded set appears as a result of upper and lower estimates for the values of the criteria.

Definition 2 . The *interval demanded set* for the problem (3) is given by the formula

$$Q_I := (q_1 + \Theta) \cap (q_2 - \Theta)$$

where:

$$q_1 \leq q_2, \quad q_1 \notin F(U), \quad q_2 \in F(U)$$

Interval demanded set in the case $\Theta := R^N_+$ may be represented as

$$Q_I = \prod_{i=1}^{N} [q_1^i ; q_2^i]$$

where q_1^i, q_2^i are lower and upper estimates of the i-th criterion demanded values respectively.

Definition 3 . The subset S_I of the interval demanded set Q_I defined by the formula

$$S_I := \{x \in Q_I \colon \exists G_i, G_j,\ i \neq j \text{ –facets of } Q_I, \text{ such that} \\ d(x, \partial Q_I) = d(x, G_i) = d(x, G_j)\} \tag{13}$$

where ∂Q_I - the boundary of Q_I - will be called the *skeleton* of Q_I.

Now, let $C(Q)$ be the subset of Q consisting of points maximally distant from the boundary of Q, i.e.

$$C(Q) := \{x \in Q \colon \forall\ y \in Q, d(y, \partial Q \leq d(x, \partial Q))]\} \tag{14}$$

and let q_1 and q_2 be two distinct elements of ∂Q such that $q_2 \leq_\Theta q_1$. If Q is convex then for each element q of the boundary of Q there exist a unique half-line $v(q)$ starting in q and such that the function $d(\bullet, \partial Q)$ grows fastest on $v(q)$ in a neighborhood of each point belonging to $v(q)$. It is easy to see that $v(q)$ links q and $C(Q)$ and it is linearly ordered. Thus we may formulate the following

Definition 4 . The *ordinal skeleton* of Q is the set

$$S_0 := v(q_1) \cup v(q_2) \cup C(Q)$$

It is evident that if $Q = Q_I$ then $S_0 \subset S_I$.

Observe that the narrower are the experts' estimations concerning a criterion F_i the smaller scope of decision is left to the decision-maker. Consequently, in some extreme cases certain criteria can be regarded rather as the constraint functions. Moreover, when

Remark 2: The property (15) of the curve S can serve as a definition of the ordinal skeleton curve in the case when the demanded set is different from Q_I. The proved property (15) of the skeleton curve is closely related to another definition of the safety of the solution admitted.

Definition 6. A non-dominated y solution to the problem (3) will be called *maximally safe* with respect to the change of bounds of Q iff for each $x \in FP(U,\Theta)$

$$P(x \in Q_\eta) \leq P(y \in Q_\eta) \tag{16}$$

where η is a probability distribution in the space of closed and convex subsets of the criteria space.

Now let us observe that Proposition 3 implies the following result concerning the safety of the solution to MMP chosen on the skeleton curve S.

Theorem 1. Let X be an arbitrary subset of Q_I. The probability distribution η defining the changes of ∂Q is assumed uniform. Then the maximally safe element of X with respect to the changes of Q belongs to S whenever $S \cap X \neq 0$.

Corollary 2 : A maximal safe non-dominated point belongs to S or is closest to S in $F(U) \cap Q$.

5. AN APPLICATION TO A DESIGN PROBLEM

Let us consider the problem of designing a construction lift taking into account the set of parameters which decide about the commercial success of the product. These criteria include the time of evaluation of the project F_1, the lifting capacity F_2, the maximal range of the arm F_3. We assume that may exist other criteria such as reliability coefficient F_4 or the production price per unit F_5 which should be simply optimized, without paying attention to the constraints in the criteria space and are not included in the model of preferences here presented. The total cost of design and investment may be regarded as a constraint, together with the employment, materials and technology limitations. We assume that all constraints form a set U of admissible design strategies. The annual net income anticipated I may serve as an aggregated utility function which, however, depends on the above listed criteria in an unknown way. We can only assume that I is monotonically depending on the measure of fulfillment of the market's expectations which are expressed by the set Q.

According to the preference model presented in the preceding subsections U is defined by upper and lower limitations for the values of criteria. These parameters can have the following practical interpretation:

F_{1l} - the minimal time necessary to distribute an announcement about the new product to the potential customers, also - if all or a prevailing part of lifts is to be sold to one company - the minimal supply time required by this company;

F_{1u} - estimated upper limit of period warranting a sufficient market's demand, or the maximal supply time required by the commissioning company, or the estimated time a similar lift will be designed and offered by other producers;

F_{2l} - minimal lifting capacity admissible for lifts of this type;

F_{2u} - maximal reasonable lifting capacity estimated basing on the knowledge of potential scope of applications of lifts;

F_{3l}, F_{3u} - similarly as above - the minimal admissible, and maximal reasonable values of the range of arm.

Each criterion should be optimized inside of the bounds F_{il}, F_{iu}, $1 \leq i \leq 3$, whereas F_1 should be minimized, the other criteria - maximized. To treat the functions F_i in an uniform way we will instead maximize the function $(-F_1)$.

The demanded set Q can be expressed in the form

$$Q = \prod_{i=1}^{3} [F_{il}, F_{iu}]$$

The bounds of Q are uncertain as the values of F_{il} and F_{iu}, $1 \leq i \leq 3$ are only estimates of the real user's needs. By Theorem 1 the strategy chosen on the skeleton set S ensures that the probability of remaining within a perturbed set Q_η maximal, η being a random perturbation coefficient of Q. Roughly speaking, the better the solution chosen fits into the set Q_η, the higher is the income I, on the other hand I should be monotonic with respect to the criteria $F_1, F_2, \ldots, F_N$. Thus we can conclude that I should be monotonically proportional to the utility function defined by the formula

$$G(u) = d(\tilde{F}(u), \partial Q) I_1(\tilde{F}(u)) + I_2(\hat{F}(u)) \tag{17}$$

where $d(\bullet, \partial Q)$ is the distance to the boundary of Q, $\tilde{F} = (F_1, F_2, F_3)$, $\hat{F} = (F_4, F_5)$ and I_1 and I_2 are certain order representing functions defined so that the maximum of G were non-dominated and situated within $Q \times R^2$ (cf. also formula in the final subsection). Let us note that the values of I_1 and I_2 are entirely independent if the values of $\tilde{F}$ and $\hat{F}$ are not depending on each other.

Hence it follows that the maximal safety with respect to $\tilde{F}$ of a compromise solution chosen is not conflicting with the goal of optimizing F in $Q \times R^2$. According to the results of the preceding subsection such a compromise value of F should be found on the skeleton curve S.

Since we do not impose any decision choice rule for the remaining criteria F_4 and F_5 we might consider two subcases:

1. $\tilde{F}$ and $\hat{F}$ are independent - then we get a family of solutions of form

   $$(\tilde{F}_c, \hat{f})_{\hat{f} \in \hat{F}P(U, \Theta)}$$

   where $\tilde{F}_c$ is the compromise value of $\tilde{F}$ found on the skeleton curve S.
2. the values of $\hat{F}$ are uniquely determined by $\tilde{F}$ - then we get a unique solution

   $$(\tilde{F}_c, \hat{F}(\tilde{F}_c)).$$

6. FINAL REMARKS

The algorithm of solving the MMP basing on the search on the skeleton curve has been implemented in FORTRAN and applied to solve real-life problems. The reader is referred to Gorecki et al. (1982, 1985) for a detailed study of decision making in the development analysis in the chemical industry.

The applications presented there show the adequacy of the decisions made via the skeleton method. Some properties of the MMP solution choice algorithm based on applying the skeleton curve have also been discussed by Wiecek (1984). The main idea of this algorithm is the same as in the general algorithm with the curve g replaced by the skeleton curve S. This algorithm can be repeated interactively, with the modified scale coefficients and the lower, and upper experts' estimations, q_2 and q_1, respectively.

The method turned out to be useful as well in case where the existence of the intersection of S and the set of non-dominated points could not been taken for granted basing on the assumptions concerning the objective F and the feasible set U. In particular, a heuristic version of the method could be applied to select a compromise solution in the case of non-convex attainable set $F(U)$ provided the decision-maker is modifying the upper and lower estimates q_1 and q_2 in accordance with the initial information about the location of $FP(U,\Theta)$ he is assumed to posses. The theoretical analysis of such a class of methods, applying the search on the skeleton curve as a single step of the procedure, with the demanded set systematically modified during and interactive decision-making process challenges the perspectives of the further development of the method.

Another possibility of investigating the theoretical fundamentals of the method consists in interpreting the search for a non-dominated solution on S as maximizing certain utility function φ which admits its local maxima on S. In this approach φ can be taken as the membership function of certain fuzzy set which describes the uncertainty of the demanded set Q. This function can have the form

$$\varphi_Q(x) := \frac{d(x,q_1)}{d(q_1,q_2)} \frac{d(x,\partial Q)}{\max\{d(y,\partial Q): y \leq_\Theta x\}}$$

It follows immediately from the above formula that φ_Q has the desired property mentioned above, i.e.

$$0 \leq \varphi_Q(x) \leq 1$$

$$\varphi_Q(x) = 1 \Leftrightarrow x = q_2,$$

$$argmax\{\varphi_Q(x) : x \leq_\Theta q_1\} \in S$$

and, moreover, φ_Q is order representing (Wierzbicki, 1980).

These properties could provide for a combination of fuzzy set theory and the skeleton method.

REFERENCES

Gorecki H., (1981). Problem of choice of an optimal solution in a multicriteria space. *Proceedings of the 8th IFAC World Congress* . Kyoto 1981; Pergamon Press, London, Vol. 10, pp 106-110.

Gorecki, H., A.M. Skulimowski (1986a). A joint consideration of multiple reference points in multicriteria decision-making. *Found. Contr. Engrg. 11; No. 2.*

Gorecki, H., A.M. Skulimowski (1986b). Group Decision-Making Maximally Safe with Respect to the Change of Aspiration Levels. (to appear).

Gorecki, H., G. Dobrowolski, J. Kopytowski, M. Zebrowski (1982). The Quest for a Concordance between Technologies and Resources as a Multiobjective Decision Process. In: M. Grauer, A. Lewandowski, and A.P. Wierzbicki Eds. *Multiobjective and Stochastic Optimization, IIASA Collaborative Proc.* Ser. CP-82-S12, Laxenburg, Austria, pp 463-476.

Gorecki, H., G. Dobrowolski, T. Rys, M. Wiecek, M. Zebrowski (1985). Decision Support System Based on the Skeleton Method - the HG Package. *Interactive Decision Making, Proc. Sopron 1984* , pp 269-280.

Skulimowski, A.M. (1985a). Solving vector optimization problems via multilevel analysis of foreseen consequences.
Found. of Control Engrg., 10; No. 1.

Skulimowski, A.M. (1985b). Generalized distance scalarization in Banach spaces, *Rev. Belge de Stat., Inf. Rech. Operationelle, 25, No.1* , pp 3-14.

Skulimowski, A.M. (1986). A sufficient condition for Θ-optimality of distance scalarizing procedures. To appear in: Proc. of the Int. Conference on Vector Optimization (J.Jahn, W.Krabs (Eds.)), Darmstadt, 5-8 August 1986.

Wierzbicki, A.P. (1980). On the Use of Reference Objectives in Multicriteria Optimization. In: *W.Fandel, T.Gal (Eds.) Multiple Criteria Decision Making - Theory and Application.*

Wiecek, M. (1984). The Skeleton Method in Multicriteria Problems. *Found. Contr. Engrg., 9, No.4,* pp 193-200.

Methodological Guide to HYBRID 3.01.: a Mathematical Programming Package for Multicriteria Dynamic Linear Problems

Marek Makowski and Janusz Sosnowski

Systems Research Institute of the Polish Academy of Sciences

1. INTRODUCTION

The purpose of this report is to provide sufficient understanding of mathematical, methodological and theoretical foundations of the HYBRID package. Section 1 contains executive summary, short program description and general remarks on solution techniques and package implementation. Section 2 contains mathematical formulation of various types of problems that can be solved by HYBRID. Section 3 presents methodological problems related to solution techniques. Section 4 presents foundations of the chosen solution technique and documents the computational algorithm. Section 5 contains short discussion of testing examples.

1.1. Executive summary

HYBRID is a mathematical programming package which includes all the functions necessary for the solution of linear programming problems. The current version of HYBRID is referred to further on as HYBRID 3.01. HYBRID 3.01. may be used for both static and dynamic LP problems (in fact also for problems with structure more general then the classical formulation of dynamic linear problems). HYBRID 3.01. may be used for both single- and multi-criteria problems. Since HYBRID is designed for real-life problems, it offers many options useful for diagnostic and verification of a problem being solved.

HYBRID is a member of a decision analysis and support system DIDAS family which is designed to support usage of multicriteria optimization tools. HYBRID can be used by an analyst or by a team composed of a decision maker and an analyst or - on last stage of application - by a decision maker alone. In any case we will speak further on about a user of a HYBRID package.

HYBRID can serve as a tool which helps to choose a decision in a complex situation in which many options may and should be examined. Such problems occur in many situations, such as problems of economic planning and analysis, many technological or engineering design problems, problems of environmental control. To illustrate possible range of applications, let us list problems for which the proposed approach either has been or may be applied: planning of agriculture production policy in a decentralized economy (both for governmental agency and for production units) [2], flood control in a watershed [25], planning formation and utilization of water resources in an agricultural region, scheduling irrigation, planning and design of purification plant system for water or air pollution.

To avoid a possible misleading conclusion that the usage of HYBRID may replace a real decision maker, we should stress that HYBRID is designed to help a decision maker to concentrate on real decision making while HYBRID takes care on cumbersome computations and provides information that serves for analysis of consequences of different options or alternatives. A user is expected to define various alternatives or scenarios, changing his preferences and priorities when learning about consequences of possible decisions. This problem is shortly discussed in Section 5 and illustrated in the tutorial example.

HYBRID could be used for that purpose as a "stand alone" package, however - after a possible modification of a problem in an interactive way - one can also output the MPS-format file from HYBRID to be used in other packages. The later approach can be used also for a transformation of a multicriteria problem to an equivalent single-criteria LP. Diagnostic functions are not performed by many other linear programming packages, e.g., by MINOS – it is interesting to note that the authors of MINOS actually advise the user to debug and verify the problem with another package before using MINOS.

HYBRID can be used for solving any linear programming problem but it is specially useful for dynamic problems; this covers a wide area of applications of operation researches. Many optimization problems in economic planning over time, production scheduling, inventory, transportation, control dynamic systems can be formulated as linear dynamic problems [17]. Such problems are also called multistage or staircase linear programming problems [18],[19]. A dynamic problem can be formulated as an equivalent large static LP and any commercial LP code may be used for solving it, if the problem corresponds to single objective optimization. For multicriteria problems, a preprocessor may be used for transformation of that problem to an equivalent LP one. The system DIDAS, described in other papers in this volume, is a package that is composed of preprocessor and postprocessor for handling transformation of multicriteria problem and processing results respectively [20]. Those pre- and postprocesors are linked with an LP package. HYBRID 3.01. has generally similar structure . The main difference is that - instead of an LP package - another algorithm is applied, which exploits the dynamics of a problem. Similarly as some other systems of DIDAS family, HYBRID has the advantage of handling a problem as a dynamic one which results in an easy way of formulation of criteria and of interpretation of results, since one may refer to one variable trajectory contrary to a "static" formulation of dynamic problems which involves separate variables for each time period.

HYBRID has been designed more for real-world problems that require scenario analysis than for academic (e.g., randomly generated) problems. Thus HYBRID is oriented towards an interactive mode of operation in which a sequence of problems is to be solved under varying conditions (e.g., different objective functions, reference points, values of constraints or bounds). Criteria for multiobjective problems may be easily defined and updated with the help of the package.

The HYBRID 3.01 is available in two versions: one for mainframes and one for PC. Each version require a FORTRAN compiler that accepts full standard of FORTRAN-77. Implementation on a particular computer requires only changes in a routine that reads system date and time.

The package has been tested on VAX 11/780 (for f77 compiler under Berkeley UNIX 4.2) and on a PC compatible with PC IBM/XT. The minimal configuration of PC consists of 512kB RAM. Intel coprocessor 8087 is strongly recommended (in fact required by some FORTRAN compilers).

1.2. SHORT PROGRAM DESCRIPTION

1.2.1. Preparation of a problem formulation

A problem to be solved should be defined as a mathematical programming model.

Firstly, a set of variables that sufficiently describe the problem - for the sake of the desired analysis - should be selected. It is desired - however not necessary - to define the problem in such a way as to possibly exploit the problem structure (further on referred to as a dynamic problem). Secondly, a set of constraints which defines a set of admissible (i.e. acceptable or recognized as feasible by a decision maker) solutions should be defined. Finally a set of criteria which could serve for a selection of a solution should be defined.

The formal definition of criteria can be performed in HYBRID in an easy way. However, it should be stressed that any definition of a complex problem usually requires cooperation of a specialist - who knowns the nature and background of the problem to be solved - with a system analyst who can advise on a suitable way of formal definition. It should be clearly pointed out that a proper definition can substantially improve the use of any computational technique. For small problems used for illustration of the method, it is fairly easy to define a problem. But for real life problems, this stage requires a close cooperation between a decision maker and a team of analysts as well as a substantial amount of time and resources.

For real life problems, the following steps are recommended:

1. Mathematical formulation of the problem being solved should be defined.
2. A data base for the problem should be created. This may be done on PC with a help of a suitable commercial product (such as Framework, dBase, Symphony, Lotus 1-2-3). Original data should be placed in this data base. A user need not worry about possible range of quantities (which usually has an impact on computational problems) because HYBRID provides automatic scaling of the problem.
3. Verification of the data base and of the model formal definition should be performed.
4. The corresponding MPS standard file should be created. This may be done by a specialized problem generator (easily written by a system analyst), or an universal generator such as GEMINI (developed at IIASA) or GAMMA (part of FMPS package on UNIVAC) or by any appropriate utility program of data base software. We strongly discourage the user from creating the MPS file with help of a standard text editor.

1.2.2. Problem verification

This stage serves for the verification of model definition which is crucial for real application of any mathematical programming approach.

First stage consists of preprocessing the MPS file by HYBRID, which offers many options helpful for that task. HYBRID points to possible sources of inconsistency in model definition. Since this information is self-explaining, details are not discussed here. It is also advisable to examine the model printout by rows and by columns, which helps to verify model specification and may help in tracing possible errors in MPS file generation.

Second stage consist of solving optimization problems for selected criteria which helps in the analysis of consistency of solutions. For larger problems a design and application of a problem oriented report writer is recommended. HYBRID optionally generates a "user_file" for that purpose which contains all information necessary for the analysis of a solution.

After an analysis of a solution, a user may change any of the following parameters: values of coefficients, values of constraints and also any parameters discussed in next section. This may be done with help of the interactive procedure which instead of MPS file uses "communication region" that contains problem formulation processed by HYBRID. Therefore, a user needs no longer to care about original MPS file which has the backup function only.

1.2.3. Problem analysis

Problem analysis consist of consecutive stages:

- analysis of obtained solution
- modification of the problem
- solution of modified problem.

Analysis of a solution consists of following steps (some of which are optional):

1. The user should examine of values of selected criteria. Since the solution obtained in HYBRID is Pareto optimal, the user should not expect improvement in any criteria without worsening some other criteria. But values of each criterion can be mutually compared. It is also possible to compute the best solutions for each criterion separately. A point (in criteria space) composed of best solutions is called the "utopia" point (since usually it is not attainable). HYBRID provides also a point composed of worst values for each criterion. This point is called "nadir" point. Such information help to define a reference point (desired values of criteria) because it is reasonable to expect values of each criterion to lie between utopia and nadir point.
2. The user may also at this stage make modifications to the original problem without involving the MPS file.
3. For dynamic problems, HYBRID allows also for examination (in also a problem oriented report writer.

Modification of the problem may be done in two ways:

1. At this stage, the user can modify the formulation of the original problem. But main activity in this stage is expected after the model is well defined and verified and no longer requires changes in parameters that define the set of admissible (acceptable) solutions. It should be stressed, that each change of this set usually results in change of the set of Pareto-optimal solutions and both utopia and nadir points should be computed again.
2. If the values of all constraints and coefficients that define the admissible set of solutions are accepted, the user should start with computations of utopia point. This can be easily done in an interactive way. After utopia and corresponding nadir points are obtained (which requires n solutions of the problem, where n is the number of criteria defined) the user can also interactively change any number of the following parameters that define the selection of an efficient solution to the multicriteria problem:
   - Reference point (i.e. desired values for each criterion) might be changed. This point may be attainable or non-attainable (cf sec.2.4.).

- Weights attached to each criterion can be modified.
- Reference trajectories in dynamic case can be changed as reference points.
- Regularization parameters in selection function can be adjusted.

3. Additionally, the user can temporarily remove a criterion (or a number of criteria) from analysis. This option results in the computation of a Pareto optimal point in respect to remaining "active" criteria but values of criteria that are not active are also available for review.

Solution of a problem. The problem defined by a user (after possible modification) is transformed by HYBRID to an equivalent LP problem which is solved without interaction of a user (an experienced user may however have an access to the information that characterizes the optimization run).

1.2.4. Remarks relevant to dynamic problems.

HYBRID allows for solving both static and dynamic LP problems. Static problems can be interpreted as problems for which a specific structure is not recognized nor exploited. But many real life problems have specific structure which - if exploited - can result not only in much faster execution of optimization runs but also remarkably help in problem definition and interpretation of results.

Numerous problems have dynamic nature and it is natural to take advantage of its proper definition. HYBRID offers many options for dynamic problems, such as:

1. In many situations, the user may deal with generic names of variables. A generic name consists of 6 first characters of a name while 2 last characters corresponds to the period of time. Therefore, the user may for example refer to the entire trajectory (by generic name) or to value of a variable for a specific time period (by full name). Such approach corresponds to a widely used practice of generating trajectories for dynamic problems.
2. The user may select any of 4 types of criteria that correspond to practical applications. Those can be defined for each time period (together with additional "global" conditions), but this requires rather large effort. Therefore, for dynamic problems, criteria are specified just by the type of criterion and the generic name of the corresponding variable. Types of criteria are discussed in details later.
3. A problem can be declared as a dynamic one by the definition of periods of time. For a dynamic problem, additional rules must be observed. These rules correspond to the way in which the MPS file has to be sorted and to the way in which names for rows and columns are selected. These rules follow a widely accepted standard of generation of dynamic problems. The formulation of a dynamic problem, which is accepted by HYBRID is actually an extension of the classical formulation of dynamic problem (cf Section 2.2.). In this formulation a model may contain also a group of constraints that do not follow the standard of state equations.

1.2.5. General description of the package and data structure

The package is constructed in modules to provide a reasonably high level of flexibility and efficiency. This is crucial for a rational use of computer resources and for planned extensions of the package and possible modification of the algorithm (see Section 5).

The package consists of three subpackages:

- Preprocessor that serves to process data, enables a modification of the problem, performs diagnostics and may supply information useful for verification of a problem. The preprocessor also transforms a multicriteria problem to a parametric single criteria optimization problem, helps in the interactive change of parameters, etc.
- Optimization package called solver of a relevant optimization problem (either static or dynamic)
- Postprocessor that can provide results in the standard MPS format and can also generate the "user file" which contains all information needed for the analysis of a solution; the later option makes it easier to link HYBRID to a specialized report-writer or a graphic package.

All three subpackages are linked by communication region, that contains all data packed in an efficient way. From the user point of view, HYBRID 3.01 is still one package that may be easily used for different purposes chosen via specification file.

The chosen method of allocating storage in the memory takes maximal advantage of the available computer memory and of the features of typical real-world problems. In general, the matrix of constraints is large and sparse, while the number of all essential, non-zero coefficients that take different numerical values is much smaller than the number of all non-zero coefficients. A super-sparse-matrix technique is therefore applied to store the data that define the problem to be solved. This involves the construction of a table of these essential coefficients. In addition, all indices and logical variables are packed so that one four-byte word is being used for four indices (2 logical and 2 integer). All data is packed in blank common to minimize the storage area used.

Special commands of HYBRID support model verification and problem modification. This is necessary to facilitate scenario analysis and to reduce the problems caused by inappropriate scaling (cf sec. 3.8.).

The data format for the input of MPS file and the output of LP results follows standards adopted by most commercial mathematical programming systems (cf e.g. [24]).

1.2.6. Outline of the solution technique

HYBRID uses a particular implementation of the Lagrange multiplier method for solving linear programming problems. General linear constraints are included within an augmented Lagrangian function. The LP problem is solved by minimizing a sequence of quadratic functions subject to simple constraints (lower and upper bounds). This minimization is achieved by the use of a method which combines the conjugate gradient method and an active constraints strategy.

In recent years many methods oriented for solving dynamic linear problems (DLP) have been developed. Most of those methods consists of adaptation of the simplex method for problems with a special structure of constraints. In HYBRID, a different approach is applied. A DLP, which should be defined together with a state equation, is solved through the use of adjoint equations and by reduction of gradients to control subspaces (more exactly, to a subspace of independent variables). The method exploits the sparseness of the matrix structure. The simple constraints (lower and upper bounds for non–slack variables) for control variables are not violated during optimization and the resulting sequence of multipliers is feasible for the dual problem. The global constraints (i.e constraints other then those defined as simple constraints) may be violated, however, and therefore the algorithm can be started from any point that satisfies the simple constraints.

The solution technique can be also used to solve single-criteria quadratic problems with virtually no changes in the algorithm. However, a routine to input and handle the relevant data and a corresponding standard for data input have yet to be designed and implemented. The solution method for multi-criteria quadratic problems requires modification of the algorithm. However the necessary modifications will be based on HYBRID 3.01 (cf sec.7 for details).

In order to provide general information about capabilities of HYBRID, the main options are listed below. HYBRID offers the following features:

- Input of data and the formulation of an LP problem follow the MPS standard. Additional rules (that concern only sequencing of some rows and columns) should be observed in order to take advantage of the structure of a dynamic problem. An experienced user may speed up computations by setting certain options and/or parameters (cf the HYBRID User Manual).
- Solution is available in the standard MPS format and optionally in a user file which contains all data that might be useful for postoptimal analysis and reports.
- A main storage area, called the *communication region*, contains all the information corresponding to a run. The communication region is stored on disk in certain situations to allow continuation of computations from failed (or interrupted) runs or to run a modified problem while using previously obtained information without the necessity of reading and processing the MPS input file.
- The multicriteria problem is formulated and solved as a sequence of parametric optimization problems modified in interactive way upon analysis of previous results.
- For static or dynamic problem, the solution technique can be chosen.
- The problem can be modified at any stage of its solution (i.e., by changing the matrix of coefficients, introducing or altering right-hand sides, ranges or bounds).
- A special problem scaling is implemented (as described by the authors in [4] and briefly discussed in Section 3.8).
- A comprehensive diagnostics is implemented, including the checking of parallel rows, the detection of columns and rows which are empty or contain only one entry, the splitting of columns, the recognition of inconsistencies in right-hand sides, ranges and bounds, and various other features that are useful in debugging the problem formulation.
- The package supports a display of a matrix by rows (printing the nonzero elements and names of the corresponding columns, right-hand sides and ranges), as well as a display of a matrix by columns (analogous to displaying by rows).
- A check of the feasibility of a problem prior to its optimization is optionally performed.
- The optimization problem solver uses a regularization of the problem (see Section 3.7).
- More detailed information for an infeasible or unbounded problem is optionally provided by the package.

1.3. Remarks on implementation

HYBRID 3.01 is an extended version of HYBRID 2.1 documented in [27]. Therefore there are only small changes in the methodological guide in comparison to the methodology presented in [27], because the solution techniques are basically the same. However, there are some important methodological innovations:

- A modification of the problem formulation and of the solution technique as well as resulting changes in the algorithm allow for solving dynamic problems with delays in both control and state variables.
- Instead of state equations for a dynamic problem, the user may specify state inequalities.
- The optimization algorithm has been improved by an automatic evaluation of some parameters, a different technical implementation of scaling, some changes in control flow, which results in its faster execution.
- The code has been modified in a way that allows for implementation on a personal computer (compatible with IBM PC/XT). A new approach to data handling provides for easier use of the package.
- Diagnostics have been improved and several observed bugs have been removed.

2. STATEMENT OF OPTIMIZATION PROBLEMS

2.1. Formulation of an LP problem

We will consider a linear programming problem (P) in the following standard form (see, e.g., [9]):

$$\min \ cx \tag{2.1}$$

$$b - r \leq Ax \leq b \tag{2.2}$$

$$l \leq x \leq u \tag{2.3}$$

where $x, c, l, u \in R^n$, $b, r \in R^m$ and A is an $m \times n$ matrix.

The constraints are divided into two groups: general constraints (2.2) and simple constraints (2.3). In the input data file (MPS file) the vectors b is called RHS and the vector r - RANGES. The vector l and u are called LOWER and UPPER BOUNDS, respectively. Obviously, some of bounds and/or ranges may have an infinite value. Therefore HYBRID may be used for solving any LP problem formulated in the way accepted by most of commercial packages.

2.2. Classical formulation of a Dynamic LP problem (CDLP)

Before discussing a formulation of a dynamic problem that can be solved by HYBRID 3.01., let us first consider a classical formulation of a dynamic linear programming problem (CDLP) (cf [17]) in the following form:

Find a control trajectory

$$u = (u_1, \ldots, u_T)$$

and a state trajectory

$$x = (x_1, \ldots, x_T)$$

satisfying the state equations with initial condition x_0

$$x_t = A_{t-1}x_{t-1} + B_t u_t - c_t \tag{2.4}$$

and constraints

$$d_{t-1} - r_{t-1} \leq F_{t-1}x_{t-1} + D_t u_t \leq d_{t-1} \quad t=1,\ldots,T \tag{2.5}$$

$$e_t \leq u_t \leq f_t \quad t=1,\ldots,T \tag{2.6}$$

$$F_T x_T \leq d_T \tag{2.7}$$

which minimize the performance index

$$\sum_{t=1}^{T} (a_t x_t + b_t u_t) \tag{2.8}$$

where:

- t=1,...T denote periods of time

- state variables x_t, control variables u_t, both for each period, are elements of Euclidian spaces of appropriate dimensions;
- matrices A_t,B_t,D_t,F_t are assumed to be given,
- RHS vectors c_t and d_t, as well as range vector r_t and bounds for control variables e_t and f_t are given,
- initial condition x_0 is given.

The above given formulation has been chosen for the purpose of simplification of presentation only. Actually, the following modifications are accepted:

1. Instead of inequality (2.5), equality constraints can be used;
2. Since no constraints of bounds type (2.6) are allowed for state variables x, such constraints may be specified in columns section of MPS file, thus formally are handled as inequality constraints of type (2.5);
3. Performance index (goal function) can either be specified as single objective or will be replaced by a dummy goal function that is defined by the transformation of a multicriteria problem to a parametric LP problem;

The structure of an CDLP problem (formulated above as in [17]) may be illustrated by the following diagram (example for T = 3, u_1,u_2,u_3,x_0,x_1,x_2,x_3 are vectors, slack variables are not shown)

$$
\begin{array}{cccccccccr}
u_1 & u_2 & u_3 & x_0 & x_1 & x_2 & x_3 & rhs & & \text{var.} \\
B_1 & 0 & 0 & A_0 & -I & 0 & 0 & c_1 & & \text{state eq.} \\
0 & B_2 & 0 & 0 & A_1 & -I & 0 & c_2 & & \text{state eq.} \\
0 & 0 & B_3 & 0 & 0 & A_2 & -I & c_3 & & \text{state eq.} \\
D_1 & 0 & 0 & F_0 & 0 & 0 & 0 & d_0 & & \text{constr.} \\
0 & D_2 & 0 & 0 & F_1 & 0 & 0 & d_1 & & \text{constr.} \\
0 & 0 & D_3 & 0 & 0 & F_2 & 0 & d_2 & & \text{constr.} \\
0 & 0 & 0 & 0 & 0 & 0 & F_3 & d_3 & & \text{final state} \\
b_1 & b_2 & b_3 & 0 & a_1 & a_2 & a_3 & - & & \text{goal}
\end{array}
$$

where I is identity matrix and 0 is a matrix composed of zero elements

2.3. Formulation of a Dynamic Problem (DLP)

The formulation of CDLP has been chosen for the purpose of simplification of presentation only. Actually HYBRID 3.01 is capable to solve problems of more general class, which will be referred to as Dynamic Linear Programming problems (DLP). Namely, the matrices $B = diag(B_i)$, $D = diag(D_i)$, $F = diag(F_i)$ need no longer be block diagonal matrices. Also matrices below identity matrices need no longer have any specific structure. Therefore the CDLP is a specific example of DLP. One of main generalizations – from a practical point of view – is that a problem with delays for control variables (which is not CDLP–class problem) may be solved by HYBRID. In fact, HYBRID accepts also problems with delays for both state and control variables, provided that state variables for periods "before" initial state do not enter state equations. A choice of criteria for CDLP–class problem is also limited in comparison with that for DLP (cf sec.4.3).

All variables are divided into two groups: decision variables u and state variables x_t, the latter are specified for each period of time

Find a trajectory x_t and decision variables u such that both:

state equations

$$-H_t x_t + \sum_{i=0}^{t-1} A_{t-1,i} x_i + B_t u = c_t, \qquad t=1,\dots,T \tag{2.9}$$

with given initial condition x_0

and constraints

$$d - r \le \sum_{t=0}^{T} F_t x_t + Du \le d \tag{2.10}$$

$$e \le u \le f \tag{2.11}$$

are satisfied and the following function is minimized:

$$\sum_{t=1}^{T} a_t x_t + bu \tag{2.12}$$

The following two symbols can be used in the specification file for definition of DLP:

NT – number of periods (stands for T in the above formulation)

NSTV - number of state variables in each period (the dimension of vectors x_t)

The user can define state inequalities instead of state equations (2.9). The slack variables for such inequalities are generated by HYBRID. For the sake of the presentation simplicity only the state equation will be considered further on.

The structure of an DLP problem may be illustrated by the following diagram: (corresponding to an example analogous to the above example for CDLP)

$$\begin{array}{lllllll}
u & x_0 & x_1 & x_2 & x_3 & rhs & \text{var.} \\
B_1 & A_{00} & -H_1 & 0 & 0 & c_1 & \text{state eq.} \\
B_2 & A_{10} & A_{11} & -H_2 & 0 & c_2 & \text{state eq.} \\
B_3 & A_{20} & A_{21} & A_{22} & -H_3 & c_3 & \text{state eq.} \\
D & F_0 & F_1 & F_2 & F_3 & d & \text{constr.} \\
b & 0 & a_1 & a_2 & a_3 & - & \text{goal}
\end{array}$$

where H_t is diagonal matrix and 0 is a matrix composed of zero elements

2.4. Multicriteria optimization

2.4.1. General remarks

The specification of a single-objective function, which adequately reflects preferences of a model user is perhaps the major unresolved difficulty in solving many practical problems as a relevant optimization problem. This issue is even more difficult in the case of collective decision making. Multiobjective optimization approaches make this problem less difficult, particularly if they allow for an interactive redefinition of the problem.

The method adopted in HYBRID 3.01 is the reference point approach introduced by Wierzbicki [21]. Since the method has been described in a series of papers and reports and has been applied to DIDAS (cf [1],[20]), we give only general outline of the approach applied. This approach may be summarized in form of following stages:

1. The user of the model (referred to further as the decision maker - DM) specifies a number of criteria (objectives). For static LP problem a criterion is a linear combination of variables. For DLP problems one may also use other types of criteria (cf sec. 2.4.2). The definition of criteria in HYBRID can be performed in an easy way described in the User Guide to HYBRID.
2. The DM specifies an aspiration level $\bar{q} = \{\bar{q}_1,\dots,\bar{q}_{NC}\}$, where $\bar{q}_i$ are desired values for each criterion. Aspiration level is called also a reference point.
3. The problem is transformed into an auxiliary parametric LP (or DLP) problem. Its solution gives a Pareto-optimal point. If 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 then $\bar{q}$. Properties of the Pareto-optimal point depend on the localization of the reference point (aspiration level) and on weights associated with criteria.
4. The DM explores various Pareto-optimal points by changing either the aspiration level $\bar{q}$ or/and weights attached to criteria or/and other parameters related to the definition of the multicriteria problem.
5. The procedure described in points 3 and 4 is repeated until satisfactory solution is found.

To give more formal presentation, let us introduce following notation:

NC	is the number of criteria
q_i	is the i-th criterion
$\bar{q}_i$	is the aspiration level for i-th criterion
w_i	is a weight associated with i-th criterion (whereas the user specifies its absolute value which is internally changed to negative depending on the type of criteria - cf sec. 2.4.3).
ϵ_m	is a given non-negative parameter.

A Pareto-optimal solution can be found by the minimization of the achievement scalarizing function in the form

$$\max_{i=1,\dots,NC} (w_i*(q_i-\bar{q}_i)) + \epsilon_m * \sum_{i=1}^{NC} w_i * q_i \rightarrow \min$$

This form of achievement function is a slight modification of a form suggested by A.Lewandowski [20] and by A.Wierzbicki [23]. Note that for ϵ_m=0 only weakly Pareto-optimal points can be guaranteed as minimal points of this function. Therefore, the use of very small ϵ_m will result in practice (except of situations in which reference point has some specific properties) in almost weakly Pareto-optimal solution. On the other hand, too big values of ϵ_m could drastically change properties associated with the first part of the scalarizing function.

2.4.2. Types of criteria

A user may define any number of criteria. To facilitate the definition 6 types of criteria are available and a user is requested to declare chosen types of criteria before their actual definition. Two types of criteria are simple linear combination of variables and those criteria may be used for both static and dynamic problems. Four other types of criteria correspond to various possible performance indices often used for dynamic problems. Since the latter criteria implicitly relate to the dynamic nature of the problem, they may be used only for dynamic problems.

For the sake of simplicity, only the variables of the type x_i (which otherwise is used in this paper to distinguish a state variable in DLP) are used in the following formulae, but in fact one can use in the definition of criteria both control and/or state variables. The only exception is the type DER of criteria, which may be defined by state variables only. Note that $x_i = \{x_{it}\}, \quad t=1,\dots T$.

An k-th criterion q_k is defined in one of following ways, for static and dynamic LP:

Type MIN

$$q_k = \sum_{t=1}^{T} \sum_{i=1}^{n} a_{it} x_{it} \rightarrow \min$$

where n is number of (state and control) variables, T is number of periods; T=1 is assumed for static LP.

Type MAX

$$q_k = \sum_{t=1}^{T} \sum_{i=1}^{n} a_{it} x_{it} \rightarrow \max$$

and exclusively for dynamic LP:

Type SUP

$$q_k = \max_{t=1,..T} (x_{it} - \bar{x}_{it}) \rightarrow \min$$

where x_i is a selected state or control variable, $\bar{x}_i$ - its reference trajectory

Type INF

$$q_k = \min_{t=1,..T} (x_{it} - \bar{x}_{it}) \rightarrow \max$$

Type FOL

$$q_k = \max_{t=1,..T}(abs(x_{it} - \bar{x}_{it})) \rightarrow \min$$

Type DER

$$q_k = \max_{t=1,..T}(abs(x_{it} - x_{it-1})) \rightarrow \min$$

which applies only to state variables.

2.4.3. Transformation of multicriteria problem to an auxiliary LP

The transformation is done by HYBRID 3.01, therefore its description here has only informative purpose. This description may be useful in case of using the MPS file (optionally created after modifications and transformation of a problem) as input for another LP package.

Following notation is used throughout this subsection:

v - name of the auxiliary variable v

w_i - weight coefficient for i-th criterion

cn_i - name of i-th criterion

ch_t - string (2-characters) which identifies t-th period of time

$\bar{q}_i$ - reference point (aspiration level) for i-th criterion

q_i - linear combination of variables that defines a criterion of the type MAX or MIN

' ' - delimiters of a string

T - number of time periods

$x_j = \{x_{jt}\}, t=1,...,T$ is a variable that defines a criterion of a type SUP,INF,FOL or DER.

Transformation will be discussed for each type of criteria:

Type : **MIN**

additional row (with name which is concatenation of following three strings: '<',cn_i,'· · ·' is generated in form:

$$-v + w_i q_i \leq w_i \bar{q}_i$$

Type : **MAX**

is transformed in the way similar to type MIN, with additional (internal, for computations only) change of the signs of w_i to negative.

Type : **SUP**

additional T rows (with names which are concatenations of strings '<', cn_i, '.' ch_t, where $t=1,...,T$) are generated in forms:

$$-v + w_i x_{jt} \leq w_i \bar{x}_{jt} + w_i \bar{q}_i$$

Type : **INF**

is transformed in the way similar to type SUP, with additional (internal, for computations only) change of the signs of w_i to negative.

Type : **FOL**

additional T columns (with names which are concatenations of strings '+', cn_i, '.', ch_t, where $t=1,...,T$) are generated ; in the following formulae this name is replaced by c_{it}^{+}

additional T columns (with names which are concatenations of strings '−', cn_i, '.', ch_t, where $t=1,...,T$) are generated ; in the following formulae this name is replaced by c_{it}^{-}

additional T rows (with names which are concatenation of strings '=', cn_i .', ch_t, where $t=1,...,T$) are generated in form :

$$c_{it}^{+} - c_{it}^{-} - x_{jt} = - \bar{x}_{jt}$$

additional T rows (with names which are concatenations of strings '<', cn_i, '.', ch_t, where $t=1,...,T$) are generated in the form:

$$-v + w_i*(c_{it}^{+} + c_{it}^{-}) \leq w_i\bar{q}_i$$

Type : **DER**

additional $2 \times T$ columns are generated in the same way as described for a criterion of the type FOL;

additional T rows (with names with are concatenations of strings '=', cn_i, '.', ch_t, where $t=1,...,T$) are generated in form :

$$c_{it}^{+} - c_{it}^{-} - B_t^j u - (A_{t-1} - I)^j x_{t-1} = -c_{jt}$$

where A_{t-1}, B_t, c_{jt} are parameters of the state equations (cf sec.3.3.3), I is the identity matrix and B_t^j and $(A_{t-1}-I)^j$ denote the j-th row of matrices B_t and $(A_{t-1}-I)$ respectively;

additional T rows (with names which are concatenations of strings '<', cn_i, '.', ch_t) are generated in form :

$$-v + w_i*(c_{it}^{+} + c_{it}^{-}) \leq w_i\bar{q}_i$$

Auxiliary goal function, which is to be minimized, is generated in the following form:

$$v + \epsilon_m*(\sum_i w_i q_i + \sum_t(\sum_j w_j x_{jt} + \sum_k w_k(c_{kt}^{+} + c_{kt}^{-})))$$

where summation is done over corresponding sets of respective criteria, i.e. indices i, j, k correspond to criteria of type: MIN or MAX, SUP or INF and FOL or DER, respectively; ϵ_m is given parameter.

The name of auxiliary variable v is '..dummy.', whereas the name of auxiliary goal function is '.dummy..'.

Value of ϵ_m may be changed by the command MEPS in a routine for modification of multicriteria parameters.

3. THEORETICAL FOUNDATIONS AND METHODOLOGICAL PROBLEMS

3.1. General remarks

The most popular methods for solving linear programming problems are based on the simplex algorithm. However, a number of other iterative non-simplex approaches have recently been developed [5-7]. HYBRID belongs to this group of non-simplex methods. The solution technique is based on the minimization of an augmented Lagrangian penalty function using a modification of the conjugate gradient method. The Lagrange multipliers are updated using a modified version of the multiplier method [8] (see Sections 3.2 and 3.4).

This method is useful not only for linear programming problems but also for other purposes, as described in Section 1.2. In addition, the method may be used to solve problems with non-unique solutions (as a result of regularization - see Section 3.7).

The following notation will be used:

a_i denotes the i-th row of matrix A

x_j denotes the j-th component of vector x

$\|x\|$ denotes the Euclidian norm of vector x

$(u)_+$ denotes the vector composed of the non-negative elements of vector u (where negative elements are replaced by zeros)

A^T denotes transposition of matrix A

3.2. The multiplier method

We shall first explain how the multiplier method may be applied directly to LP problems.

Consider the problem (PO), which is equivalent to the problem (P):

$$\min \; cx$$

$$Bx \leq d \tag{PO}$$

where $d \in R^p$, B is a $p \times n$ matrix, and $m \leq p \leq 2(m+n)$. To apply the multiplier method to this problem we proceed as follows:

Select initial multipliers y^0 (e.g., $y^0 = 0$) and $\rho \in R$,$\rho > 0$. Then for $k = 0,1,\ldots$, determine successive values of x^{k+1}, y^{k+1} where

$$x^{k+1} = \underset{x}{argmin}\; L(x,y^k)$$

and

$$y^{k+1} = (y^k + \rho(Bx_{k+1} - d))_+ \tag{3.1}$$

where

$$L(x,y^k) = cx + (\|(y^k + \rho(Bx-d))_+\|^2 - \|y^k\|^2)/(2\rho) \tag{3.2}$$

until a stopping criterion is satisfied.

The method has the following basic properties:

1. A piecewise quadratic differentiable convex function is minimized at each iteration.
2. The algorithm terminates in a finite number of iterations for any positive ρ.
3. There exists a constant $\bar{\rho}$ such that for any $\rho \geq \bar{\rho}$ the algorithm terminates in the second iteration.

Note that it is assumed above that the function $L(\cdot,y^k)$ is minimized exactly and that the value of the penalty parameter ρ is fixed. Less accurate minimization may be performed provided that certain conditions are fulfilled (see, e.g., [7,8]). For numerical reasons, a non-decreasing sequence of penalty parameters $\{\rho^k\}$ is generally used instead of a fixed ρ.

3.3. The conjugate gradient method for the minimization of an augmented Lagrangian penalty function

The augmented Lagrangian function for a given vector of multipliers y will be called the augmented Lagrangian penalty function [22]. For minimization of that function the conjugate gradient method has been modified to take advantage of the formulation of the problem. The method may be understood as an modification of the techniques developed by Polyak [10], O'Leary [11] and Hestenes [12] for minimization of a quadratic function on an interval using the conjugate gradient method.

The problem (P) may be reformulated as follows:

$$\begin{aligned} &\min \ cx \\ &Ax + z = b \\ &l \leq x \leq u \\ &0 \leq z \leq r \end{aligned} \tag{PS}$$

where $z \in R^m$ are slack variables.

Formulation (PS) has a number of advantages over the initial formulation (PO):

1. The dimension of matrix A in (PS) is usually much smaller than that of matrix B in (PO).
2. The problem is one of minimization of a quadratic function in (PS), and of minimization of a piecewise quadratic in (PO).
3. Some computations only have to be performed for subsets of variables. Note that slack variables are introduced only for ease of interpretation and do not have to be computed.

In (PS) the augmented Lagrangian is defined by

$$L(x,z,y) = cx + (\|y + \rho(Ax+z-b)\|^2 - \|y\|^2)/(2\rho) \ . \tag{3.3}$$

We shall first discuss the problem of minimizing $L(x,z,y)$ for given $y,\rho > 0$, subject to lower and upper bounds for x and z. Let us consider the following augmented Lagrangian penalty function

$$F(x,z) = (c/\rho)x + (\|y/\rho + Ax - b + z\|^2 - \|y/\rho\|^2)/2. \tag{3.4}$$

The gradient of F is defined by

$$\frac{\partial F}{\partial x} = c/\rho + A^T(z-g)$$

$$\frac{\partial F}{\partial z} = z - g$$

where

$$g = -y/\rho - Ax + b \ .$$

From the Kuhn–Tucker optimality condition, the following relations hold for the minimum point (x^*, z^*):

$$\frac{\partial F^*}{\partial x_j} \geq 0 \text{ if } x_j^* = l_j \ , \ \frac{\partial F^*}{\partial x_j} \leq 0 \text{ if } x_j^* = u_j \ ,$$
$$\frac{\partial F^*}{\partial z_i} \geq 0 \text{ if } z_i^* = 0 \ , \ \frac{\partial F^*}{\partial z_i} \leq 0 \text{ if } z_i^* = r_i \ ,$$

and

$$\frac{\partial F^*}{\partial x_j} = 0 \text{ if } l_j < x_j^* < u_j$$
$$\frac{\partial F^*}{\partial z_i} = 0 \text{ if } 0 < z_i^* < r_i \ .$$

For any given point such that $l \leq x \leq u$ it is possible to determine slack variables $0 \leq z \leq r$ in such a way that the optimality conditions with respect to z are obeyed. Variables z are defined by

$$z_i = \begin{cases} 0 & \text{if } g_i \leq 0 \quad (\partial F/\partial z_i > 0) \\ r_i & \text{if } g_i \geq r_i \quad (\partial F/\partial z_i < 0) \\ g_i \text{ if } r_i > g_i > 0 & (\partial F/\partial z_i = 0) \ . \end{cases} \tag{3.5}$$

We shall use the following notation and definitions. The vector of variables x with indices that belong to a set J will be denoted by x^J, and analogous notation will be used for variables g. We shall let q denote minus the gradient of the Lagrangian penalty function reduced to x-space ($q = -(\partial F/\partial x)$). The following sets of indices are defined for a given point x:

The set of indices I of violated constraints, i.e.,

$$I = \{i : g_i \geq r_i\} \bigcup \{i : g_i \leq 0\} \ .$$

$\bar{I}$ is the complement of I, i.e.,

$$\bar{I} = \{1,2,....,m\} \backslash I \ .$$

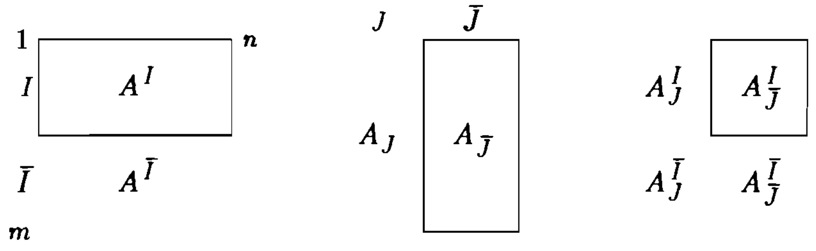
The set of indices I can be also interpreted as a set of active simple constraints for z. The set of indices J of variables that should be equal to either the upper or the lower bound, i.e.,

$$J = \{j : x_j = l_j \text{ and } q_j \leq 0\} \bigcup \{j : x_j = u_j \text{ and } q_j \geq 0\} \ .$$

$\bar{J}$ is the complement of J, i.e.,

$$\bar{J} = \{1,2,.....,n\} \backslash J \ .$$

For the sake of illustration the matrix A may be schematically split up in the following three ways (see the Figure below): first according to active rows, second according to basic columns and third with illustrate the part of the matrix A for which augmented Lagrangian penalty function is computed. The contents of the matrix $A_{\bar{J}}^{I}$ (for which the augmented Lagrangian penalty function is computed) changes along with computations.



In essence, the augmented Lagrangian penalty function is minimized using the conjugate gradient method with the following modifications:

1. During the minimization process x and z satisfy simple constraints and z enters the augmented Lagrangian in the form defined by (3.5).
2. The conjugate gradient routine is run until no new constraint becomes active, i.e., neither set I nor set J increases in size. If this occurs, the computed step length is shortened to reach the next constraint, the corresponding set (I or J) is enlarged and the conjugate gradient routine is re-entered with the direction set equal to minus the gradient.
3. Sets J and I are defined before entering the procedure discussed in point 2 and may be only enlarged before the minimum is found. When the minimum with respect to the variables with indices in sets $\bar{J}$ and I has been found, sets J and I are redefined.
4. Minimization is performed subject only to those components of variables x whose indices belong to set $\bar{J}$, i.e., variables that are not currently equal to a bound value.
5. Minimization is performed subject only to those components of variables z whose indices do not belong to set I, i.e., slack variables that correspond to non-active simple constraints for z. Note that, formally, this requires only the use of different formulae for z. In actual fact it is sufficient to know only the set I, which defines the minimized quadratic function.

4. SOLUTION TECHNIQUE

4.1. Algorithm for minimization of augmented Lagrangian

We may now present the algorithm for minimization of the augmented Lagrangian penalty function in a more formal way. The algorithm consists of the following steps:

1. For given y and $\rho>0$ choose a point x such that $l \le x \le u$
2. Compute $g = -y/\rho - Ax + b$
3. Determine sets I and $\bar{I}$

   $I = \{i: g_i > r_i\} \bigcup \{i: g_i < 0\},$

   $\bar{I} = \{1,...,m\} \backslash I$

4. Redefine $g$as follows:

   $$g_i := \begin{cases} g_i - r_i & \text{if } g_i - r_i > 0 \\ g_i & \textit{otherwise} \end{cases}$$

5. Compute the minus gradient:

   $q = -c/\rho + (A^I)^T g^I$

6. Determine sets J and $\bar{J}$

   $J = \{j: x_j = l_j \text{ and } q_j \le 0\} \bigcup \{j: x_j = u_j \text{ and } q_j \ge 0\}$

   $\bar{J} = \{1,...,n\} \backslash J$

7. If $q_j = 0$ for all $j \in \bar{J}$ then x is a minimum point of the augmented Lagrangian penalty function
8. Set $p^{\bar{J}} = q^{\bar{J}}$
9. Compute

   $s = A_{\bar{J}} p^{\bar{J}}$

   $h = \|q^{\bar{J}}\|^2$

   $d = \|s^I\|^2$

   $\alpha(1) = h/d$

   Note that $\alpha(1)$ is the conjugate gradient step length in direction $p^{\bar{J}}$

10. Find the step length that would violate the nearest non-active constraint, i.e., for $i \in \bar{I}$,

    $\alpha(2) = \min_{i \in K} \{g_i/s_i\}, \quad K = \{i : i \in \bar{I}, s_i > 0\}$

    $\alpha(3) = \min_{i \in K} \{(g_i - r_i)/s_i\}, \quad K = \{i : i \in \bar{I}, s_i < 0\}$

11. Find the step length that would enable a variable to reach a bound, i.e.,

$$\alpha(4) = \min_{j\in K} (l_j - x_j)/p_j\ ,\quad K = \{j : j\in\bar{J},\ p_j<0\}$$

$$\alpha(5) = \min_{j\in K} (u_j - x_j)/p_j\ ,\quad K = \{j : j\in\bar{J},\ p_j>0\}$$

12. Determine step length $\alpha = \min_{i=1,\dots,5} (\alpha(i))$. If $\alpha = min(\alpha(2),\alpha(3))$ add the row index for which this condition holds to set I and remove that index from set $\bar{I}$. If $\alpha = min(\alpha(4),\alpha(5))$ add the column index for which this condition holds to set J and remove that index from set $\bar{J}$.

13 Compute the new point $x^{\bar{J}} := x^{\bar{J}} + \alpha p^{\bar{J}}$ and the minus gradient at that point:

$$g_i := g_i - \alpha s_i$$

$$q^{\bar{J}} = (A_{\bar{J}}^{I})^T g^I - c^{\bar{J}}/\rho$$

14. If $q^{\bar{J}} = 0$. go to step 2

15. If $\alpha = \alpha(1)$ continue with the conjugate gradient step, i.e.

$$\beta = \|q^{\bar{J}}\|^2/h$$

$$p^{\bar{J}} := q^{\bar{J}} + \beta p^{\bar{J}}$$

and go to step 9

16. Go to step 8

Note that the condition $q^{\bar{J}} = 0$ is in practice replaced by $\|q^{\bar{J}}\| \le \epsilon^k/\rho$ The value of ϵ^k may be quite large in the first few iterations; it then decreases as the number of iterations increases.

4.2 Adaptation of the multiplier method

Let the violation of i-th constraint in a point x^k be defined in the following way:

$$v_i^k = \max\{a_i x^k - b_i,\ - a_i x^k + b_i - r_i,\ 0\ \}$$

and $\|v^k\|$ denotes the norm of violated constraints. The multiplier method will be presented in algorithmic form.

1. Compute an initial vector of multipliers on the basis of the particular option chosen (i.e., either $y^0 = 0$ or y^0 corresponding to the constraints violated at starting point x)
2. Find x^{k+1} which minimizes the augmented Lagrangian penalty function (see Section 3.3.) with accuracy ϵ^k. It is assumed that

   $$\epsilon^k := \min\ (\epsilon^k, \|v^k\|\epsilon^k))$$

   where the sequence $\epsilon^k \to 0$. In addition, $\epsilon_{mi} \ge \epsilon^k \ge \epsilon_{mx}$, where ϵ_{mi}, ϵ_{mx} is the assumed minimum and maximum accuracy, respectively.
3. If $\|q^{\bar{J}}\| \le \epsilon^k$ and the last step has been a multiplier update go to step 6 (where $\|q^{\bar{J}}\|$ is the norm of the gradient of the augmented Lagrangian penalty function).

4. If $\|q^J\| \leq \epsilon^k$ and the last iteration has been a multiplier update set $\rho^k := \min(\rho^k \rho_s, \rho_m)$ (where $\rho_m$is the assumed maximum value of the penalty parameter and ρ_s is assumed to be constant)
5. If $\rho^k = \rho_m$ then set $\epsilon^k := \max(\epsilon^k \epsilon_s, \epsilon_{mx})$ where ϵ_s and ϵ_m are assumed parameters.

   If $\rho^k = \rho_m$ and $\epsilon^k = \epsilon_{mx}$ go to step 6. Otherwise go to step 2
6. Compute new multipliers

$$y_i^{k+1} := \begin{cases} y_i^k + \rho^k(a_i x^{k+1} - b_i) & \text{if } y^k + \rho^k(a_i x^{k+1} - b_i) \geq 0 \\ y_i^k + \rho^k(a_i x^{k+1} - b_i + r_i) & \text{if } y^k + \rho^k(a_i x^{k+1} - b_i + r_i) \leq 0 \\ 0 & \textit{otherwise} \end{cases}$$

7. If $\|y^{k+1} - y^k\| > \epsilon_d$ then set $\rho^{k+1} = \min(\rho^k \rho_s, \rho_{mx})$, set $\epsilon^{k+1} = \epsilon^k \epsilon_s$, $k := k + 1$ and go to step 2
8. Check the feasibility of the current point. If it is feasible, $\|v^k\| \leq FEAS$, minimize the augmented Lagrangian penalty function with the vector of multipliers fixed at y^{k+1} and with accuracy ϵ^{k+1}, and then stop
9. If the point tested at step 8 was infeasible and $\rho^k < \rho_{mx}$ then set $\rho^{k+1} = \min(\rho^k \rho_s, \rho_{mx})$, set $k := k + 1$ and go to step 2
10. If step 9 was omitted, check the feasibility of the problem by minimizing the square Euclidian norm of the violated constraints. If the problem is infeasible, then stop.
11. Take the feasible solution found in step 10 as the current point, set $k := k + 1$, update $\epsilon^k = \max(\epsilon^k \epsilon_s, \epsilon_{mx})$ and go to step 2.

The list of parameters which are referred to in the User Guide to HYBRID and their relative symbols used above is as follows (index k is omitted):

RO - ρ, ROST - ρ_s , ROMX - ρ_{mx} , EPS - ϵ, EPSS - ϵ_s, EPSM - ϵ_{mx}, EPSD - ϵ_d.

4.3. Solution technique for DLP

We will not repeat reasoning given in the first part of sec. 2.3. Instead, let us point out basic differences between the algorithms for static LP and DLP:

1. Minimization is reduced to a subspace of decision variables. Gradient of Lagrangian penalty function is computed for variables that belong to a subspace of decision variables. This (together with arguments already presented in sec. 3.3.) shows advantages due to the use of dynamic structure of DLP problem in comparison with presentation of such a problem as a large LP.

2. The structure of matrices $B_1, \ldots\ldots, B_T$ and $F_0, \ldots, F_T$ has no impact for the algorithm nor affects the technique of storage of data, because super-sparse technique is applied (cf sec.1.4.). It should be also pointed out that the method of transforming a multicriteria problem to a parametric LP one introduces constraints (cf sec.2.4.3.) that - for the proposed (cf sec.2.4.2.) types of criteria - do not fit to the staircase structure of CDLP (cf [17]). Therefore, any technique that would exploit the staircase structure of DLP would also imply a reduction of a number of criteria types. The alternative is then to treat a problem as a large LP static one or to apply a technique that does not exploit the classical DLP structure.

3. State equations are solved (for given decision variables u) recursively and are fulfilled in any stage of computations. Therefore any single constraints for state variables have to be treated as general constraints and included into the matrix. Gradient need not to be computed for those variables, but state equation is solved twice (for state variables and variations).
4. A conjugate trajectory Ψ is computed from conjugate equation and has an interpretation of dual variables for state equations. No other variables associated with those rows (defined in sec. 3.3, i.e. Lagrange multipliers, shifted constraints g) are computed for state equations rows.
5. The general structure of the algorithm for DLP is similar to that presented in sec. 3.4. To sum up basic differences one may observe that
   - we consider a problem that is equivalent to a static LP but reduced to the subspace of decision variables and is solved in the way similar to that described in sec. 3.3. and 3.4.
   - state equations are solved for control variables and for variations
   - a conjugate trajectory Ψ is computed.

4.4. Algorithm for minimization of augmented Lagrangian for DLP

Now we may present the algorithm for minimization of the augmented Lagrangian function for DLP in a more formal way. In each iteration of multiplier method, the following optimization problem is solved: minimize the augmented Lagrangian penalty function

$$F(x,u,z) = \sum_{t=1}^{T}(a_t/\rho)x_t + (b/\rho)u +$$
$$+ (\|y/\rho + \sum_{t=0}^{T} F_t x_t + Du - d + z\|^2 - \|y/\rho\|^2)/2.$$

subject to

$$-H_t x_t + \sum_{i=0}^{t-1} A_{t-1,i}x_i + B_t u = c_t \qquad t=1,\dots T$$

with a given initial condition x_0 and

$$e \le u \le f$$
$$0 \le z \le r$$

where z is a vector of slack variables, which - as discussed in sec. 3.3. - are not used in the algorithm. The algorithm consists of the following steps:

1. For given y and ρ choose a point u such that $e \le u \le f$
2. Solve the state equation

   $$H_t x_t = \sum_{i=0}^{t-1} A_{t-1,i}x_i + B_t u - c_t \quad t=1,\dots,T$$

   with given initial condition x_0

3. Compute shifted constraints for constraints (2.10.)

$$g = -y/\rho - \sum_{t=0}^{T} F_t x_t - Du + d$$

and determine sets I, $\bar{I}$

$$I = \{i\colon g_i > r_i\} \bigcup \{i\colon g_i < 0\}$$

while $\bar{I}$ is the complement of I.

4. Redefine g as follows :

$$g_i := \begin{cases} g_i - r_i & \text{if } g_i > r_i \\ g_i & \textit{otherwise} \end{cases}$$

5. Find the conjugate trajectory by solving backwards the conjugate equations

$$H_t^T \Psi_t = \sum_{i=t}^{t-1} A_{i,t}^T \Psi_{t+1} + (F_t^I)^T g^I - a_t/\rho \ , \quad t = T-1,...,1$$

with boundary condition

$$\Psi_T = (F_T^I)^T g^I - a_T/\rho$$

6. Compute the minus gradient reduced to subspace of decision variables

$$q = -b/\rho + (D^I)^T g^I + \sum_{t=1}^{T} B_t^T \Psi_t$$

7. Determine sets J and $\bar{J}$

$$J = \{j : u_j = e_j \text{ and } q_j \le 0\} \bigcup \{j : u_j = f_j \text{ and } q_j \ge 0\}$$

while $\bar{J}$ is the complement of J

8. If $q_j = 0$ for all $j \in \bar{J}$ then u is a minimum point of the augmented Lagrangian penalty function

9. Set $p^{\bar{J}} = q^{\bar{J}}$

10. Solve state equation in variations

$$\sigma_t = A_{t-1}\sigma_{t-1} + B_t^{\bar{J}} p^{\bar{J}} \qquad t=1,...,T$$

with boundary condition $\sigma_0 = 0$

11. Compute

$$s = D^{\bar{J}} p^{\bar{J}} + \sum_{t=0}^{T} F_t \sigma_t$$

$$h = \|q^{\bar{J}}\|^2$$

$$v = \|s^I\|^2$$

$$\alpha(1) = h/v$$

Note that $\alpha(1)$ is the conjugate gradient step length in direction $p^{\bar{J}}$

12. Find the step length that would violate the nearest non-violated constraint, i.e.,

$$\alpha(2) = \min_{i \in K} \{g_i / s_i\}, \quad K = \{i : i \in \bar{I} \text{ and } s_i > 0\}$$

$$\alpha(3) = \min_{i \in K} \{(g_i - r_i)/s_i\}, \quad K = \{i : i \in \bar{I} \text{ and } s_i < 0\}$$

13. Find the step length that would enable a variable to reach a bound, i.e.,

$$\alpha(4) = \min_{j \in K} \{(e_j - u_j)/p_j\}, \quad K = \{j : j \in \bar{J} \text{ and } p_j < 0\}$$

$$\alpha(5) = \min_{j \in K} \{(f_j - u_j)/p_j\}, \quad K = \{j : j \in \bar{J} \text{ and } p_j > 0\}$$

14. Determine step length $\alpha = \min_{i=1,\dots,5} (\alpha(i))$

If $\alpha = min(\alpha(2),\alpha(3))$ add the index for which this condition holds to set I and remove that index from set $\bar{I}$. If $\alpha = \min(\alpha(4),\alpha(5))$ add the index for which this condition holds to set J and remove that index from set $\bar{J}$.

15. Compute :

$$u^{\bar{J}} := u^{\bar{J}} + \alpha p^{\bar{J}}$$

$$x_t := x_t + \alpha \sigma_t$$

$$g_i := g_i - \alpha s_i$$

16. For the new g^I solve the conjugate equation (as in step 5)

17. Compute the minus gradient :

$$q^{\bar{J}} = -b^{\bar{J}}/\rho + (D^I_{\bar{J}})^T g^I + \sum_{t=1}^{T} (B_t)^T_{\bar{J}} \Psi_t$$

18. If $q^{\bar{J}} = 0$, then go to 2

19. If $\alpha = \alpha(1)$ continue with the conjugate gradient step, i.e.

$$\beta = \|q^{\bar{J}}\|^2 / h$$

$$p^{\bar{J}} = q^{\bar{J}} + \beta p^{\bar{J}}$$

and go to step 10

20. Go to step 9

Note that the condition $q^{\bar{J}} = 0.$ is in practice replaced by $\|q^{\bar{J}}\| \leq \epsilon^k$ The value of ϵ^k may be quite large in the first few iterations; it then decreases as the number of iterations increases.

4.5. Regularization

It is possible that a linear programming problem may have nonunique optimal solutions. Although this is theoretically rare, in practice many problems actually have a large set of widely varying basic solutions for which the objective values differ very little [7]. In some cases, the simplex algorithm will stop when a basic solution is recognized as optimal for a given set of tolerances. For problems with a nonunique optimum, the first optimal solution found is accepted, so that one may not even be aware of the non-uniqueness of the solution reported as optimal.

Thus we are faced with the problem of choosing an optimal (or, in most cases, to be more accurate, a suboptimal) solution that possesses certain additional properties required by the user. This problem may be overcome by applying an approach called *regularization*. Regularization (Tikhonov's type) is a means of finding the optimal solution with either minimum Euclidian norm or minimum distance from a given reference point. The second of these options has not yet been implemented; the first may be activated by a REGZERO statement in the specification file (see the User Guide to HYBRID).

The minimum norm solution is obtained by carrying out a sequence of minimizations of regularized augmented Lagrangians rather than one minimization of an "ordinary" augmented Lagrangian [16]. Thus minimization of $L(\cdot,y^k)$ in problem (PO) is replaced by

$$x^{k+1} = \underset{x}{argmin}\ L(x,y^k) + \| x \|^2/(2\eta^k)$$

where

$$\eta^k \rightarrow \infty \ , \quad \sum_{i=1}^{\infty} (\rho^k/\eta^k))^{1/2} < \infty \ , \ \eta^{k+1} = min(\eta^k,\ \eta_s,\ \eta_m)$$

η_0 , η_s and η_m are given parameters.

The list of parameters which are referred to in the User Manual to HYBRID and their relative symbols used above is as follows :

RETA - η_0, RSETA - η_s, RMETA - η_m

4.6. Scaling

It is generally agreed that the choice of an appropriate scaling of a problem being solved can be a critical issue for numerical stability. There are obviously two approaches to deal with that problem. First, suggested by Tomlin ([15]), assume that an experienced model builder, who uses sensible units may avoid unnecessarily large or small matrix elements. This is true, but requires a lot of time consuming preparations, which are reliable source of frustrating bugs. Therefore, we have followed the second approach, suggested by Curtis and Reid ([14]) for solving the scaling problem. This approach is nowadays widely accepted (e.g. the new version of MINOS has also scaling option, which has removed many problems typical for older versions of MINOS).

Our approach is discussed in details in [4], therefore only short description follows. For the sake of simplicity we consider a problem of scaling on an example of a problem in a form

$$\begin{aligned} &Ax = b \\ &d \leq x \leq q \end{aligned} \tag{3.6.}$$

where $A \in R^m \times n$

According to Curtis and Reid (1972) matrix A is considered as well-scaled if

$$\sum_{i=1}^{m} \sum_{j \in J_i} (log(abs(a_{ij})))^2 \leq v \tag{3.7.}$$

for some acceptable v. J_i are sets of indices of columns with non-zero elements in i-th row.

Therefore, instead of solving a badly conditioned problem a of type (3.6.), one can solve an equivalent problem in form

$$(RAC)y = Rb$$
$$C^{-1}d \leq y \leq C^{-1}q$$
$$x = Cy$$

Here $R = diag(r_1,....,r_m)$ and $C = diag(c_1,....,c_n)$ are two diagonal matrices with positive components. In other words, an equivalent problem is formed by multiplying i-th row by r_i and j-th column by c_j.

The problem of scaling boils down to finding coefficients r_i and c_j such that

$$\sum_{i=1}^{m} \sum_{j \in J_i} (log(r_i c_j abs(a_{ij})))^2 \rightarrow \min$$

It is easy to observe that the above stated problem has no unique solution (although the optimally scaled matrix may be unique). Therefore we minimize the following performance index:

$$\sum_{i=1}^{m} \sum_{j \in J_i} (log(r_i c_j abs(a_{ij})))^2 + \beta * \sum_{i \in K} (log(abs(r_i rhs_i)))^2 + \qquad (3.8.)$$
$$\gamma * \sum_{i \in L} (log(abs(c_j bnd_i)))^2 \rightarrow \min$$

where rhs and bnd are non-zero elements of RHS and bounds, respectively, sets of indices K and L contain indices of rows with non-zero rhs and columns with non-zero bounds, respectively.

For the numerical reasons the base of logarithms is 2 and obtained coefficients are rounded to nearest integer number.

For this formulation of the scaling problem, it was possible to design a specialized algorithm based on conjugate gradient method. Since an excessive accuracy is not required, the scaling algorithm is very efficient (usually it takes less then 10 iterations regardless of dimension of a problem). Therefore the scaling option (which is the default) should not be suppressed except if special requirements apply. The values of performance indices (3.7.) and (3.8.) are displayed both before and (if active) after scaling.

Usually there is no need to change default parameters. Should a change of parameters be desired, it may be done by entering respective values in specification file (SBETA stands for β and SETA stands for γ). Two stopping criteria are used, which may be controlled by parameters SEPS and SEP1. Let v^k be a value of the performance index (3.8.). The scaling routine is ended, if $v^k/v^{k-1} \geq$ SEPS or if the norm of gradient is less then SEP1. In addition the number of scaling iterations in constrained by ITSCAL (cf the User Guide to HYBRID).

Scaling coefficients are displayed as additional column in MPS-type output of results. This has only informative purpose, since all results are rescaled internally.

5. TESTING EXAMPLES

HYBRID has been tested on number of examples. For the sake of illustration of the package capabilities 3 known examples have been selected: two dynamic and one static.

5.1. Econometric growth model (Manne)

This model is a linear multicriteria version of Manne's model described in [26].

The variables are the following:

t	time period, $t = 1,2,...,T$
c_t	consumption
I_t	investment,
K_t	capital in time period t.

$$\max \sum_{t=1}^{T} \beta_t c_t \qquad \text{(MAX)}$$

$$\max K_T \qquad \text{(MAX)}$$

$$\min \max_{t=1,2,...,T} |c_t - \bar{c}_t| \qquad \text{(FOL)}$$

The state equation:

$$K_t = K_{t-1} + I_t, \qquad t=1,2,...,T$$

with K_0 given.

Linear constraints for $t = 1,2,...,T$

$$c_t + I_t \le \alpha_{t-1} K_{t-1}$$

$$K_t \ge K_0 + I_0$$

Bounds:

$$C_t \ge C_0$$

$$I_t \le (1.04)^t I_0$$

Parameters:

$$\beta_t = 0.95^t, \ b=0.25, \ g=0.03, \ c_0=0.65,$$

$$I_0=0.16, \ K_0=3.0, \ \alpha_t=\alpha(1+g)^{(1-b)t}$$

where $\alpha=(C_0+I_0)/K_0$.

In the table 1 the test examples which refers to the modified Manne problem are denoted by MannT, where T corresponds to a number of periods.

5.2. Flood control problem.

The problem is a model (cf [25]) of the water system which consists of three general purpose reservoirs supplying water to the main river reach. The goal of the system dispatcher is to operate the reservoirs in such a way that the flood peak on the main river do not coincide. It is assumed that inflow forecast for each reservoir is known.

The model consists of water balance equations for selected points and for each time period. The capacities of reservoirs are also constraint. Various types of criteria are examined:

FOL – corresponds to following given trajectories of water flow in selected points,

DER – corresponds to minimization water flow changes (in consecutive time periods) in selected points,

MAX – corresponds to minimization of maximal (over time) flow in selected points.

In the table 1 the test examples which refers to the multicriteria flood control problems are denoted by FloodT, where T corresponds to a number of periods.

5.3. Full dense LP problem.

This problem is a modification of the Mangasarian example [5] and has been generated for verification of the package for fully dense LP problems. Computations are performed for one criterion and elements of matrix are equal to 1.0 with exception of diagonal elements for which values of 10.0 are selected.

In the table 1 the test examples which refers to the modified Mangasarian example are denoted by MangT, where T corresponds to a dimension of LP matrix.

5.4. Discussion of test results.

Testing problems have been solved on a PC compatible with IBM/AT with 80287 coprocessor. The algorithm was implemented with double precision arithmetic (the machine precision about 2.22e-16). The default values of all parameters (this includes initial multipliers equal to zero) were assumed in all runs.

The results of some tests are summarized in the following table.

Problem	Number of crit.	Rows	Cols	Dens. [%]	Time (min.)	Mult. iter.	Outer iter.	Total steps
Manne05	3	29	27	12	0.4	2	13	24
Manne10	3	54	52	7	0.6	2	23	28
Manne20	2	103	102	3	3.0	2	41	72
Manne30	2	153	152	2	5.0	2	64	112
Manne40	2	203	202	2	9.5	2	84	154
Flood03	6	55	55	6	5.0	10	87	230
Flood05	3	77	79	4	4.5	2	36	172
Mang20	1	20	20	100	2.0	2	4	49
Mang30	1	30	30	100	5.0	2	4	76

Numbers of rows and columns correspond to a single criterion LP problem, which were obtained by transformation of relevant multicriteria problems. The numbers of outer iterations and of total steps correspond to execution of step 2 and step 3 of the algorithm (cf sec. 4.1.).

Due to super sparse matrix technique applied for storing data, rather long computation time is required for fully dense matrix problems. For dynamic sparse problems better performance of the algorithm was observed. One should also note that the Flood problem is badly conditioned and is reported by many LP packages as infeasible.

6. CONCLUSIONS

First version of HYBRID was made operational in 1982. This version is documented in [13]. Then we had improved and extended the package for dynamic linear programming problems (DLP) and for multicriteria problems (both static and dynamic). The later version in documented in [27].

HYBRID 3.01 is still a pilot-type of software that requires a lot of testing. It is true that for some problems HYBRID 3.01 performs worse than the commercial packages FMPS and MINOS but for some other problems HYBRID performs better, especially if a problem is defined as a dynamic one. If HYBRID is used not only for one run but for scenario analysis (solving the problem with change of multicriteria parameters, matrix elements, RHS etc.) its performance is much better. The reason being so is not only due to the fact that MPS file is processed only in a first run but mainly because in consecutive runs (which uses *communication region*) only update of affected coefficients is made (the problem is generated only for the first run) and because a solution is usually obtained much faster then for the first run (HYBRID – contrary to simplex approach – uses the same solution technique for any possible modification of a problem being solved).

HYBRID provides very useful diagnostics for any LP problem and therefore is also useful for a problem verification. It could be used for that purpose as "stand alone" package, and - also after possible modification of a problem in interactive way - one may output MPS-format file to be used by other packages. The same approach may be used for transformation of multicriteria problem to equivalent single-criteria LP.

The further development of HYBRID will proceed in following directions:

1. Modification of the way in which the user communicates with the package. The modification will exploit capabilities of PC compatible with IBM/XT and will remarkably ease the use of the package.
2. Extensions of capabilities of HYBRID by introduction of new options for definition and handling of multicriteria problem (new types and more flexible definition of criteria, introduction of both aspiration and reservation levels, data base for previous runs etc).
3. Further improvement of the algorithm and its computer code (automatic evaluation of some parameters, experiments with possible modification of the algorithm) that will result in faster execution.

We hope that, despite the reservations outlined above, HYBRID 3.01. will eventually be a useful tool with many practical applications. We would be grateful for any criticisms and comments that would help us to improve the package.

7. REFERENCES

1. M. Kallio, A. Lewandowski and W. Orchard-Hays. An implementation of the reference point approach for multiobjective optimization. WP-80-35. International Institute for Applied Systems Analysis, Laxenburg, Austria, 1980.
2. M. Makowski and J. Sosnowski. A decision support system for planning and controlling agricultural production with a decentralized management structure. In: Plural Rationality and Interactive Decision Processes. Eds. M. Grauer, M. Thompson, A.P. Wierzbicki. Springer - Verlag, 1985.
3. A.P. Wierzbicki. A methodological guide to multiobjective decision making. WP-79-122. International Institute for Applied Systems Analysis, Laxenburg, Austria, 1979.
4. M. Makowski and J. Sosnowski. Implementation of an algorithm for scaling matrices and other programs useful in linear programming. CP-81-37. International Institute for Applied Systems Analysis, Laxenburg, Austria, 1981.
5. O.L. Mangasarian. Iterative solution of linear programs. *SIAM Journal for Numerical Analysis*, 18(4): 606–614, 1981.
6. B.T. Polyak and N.V. Tretiyakov. An iterative method for linear programming and its economic interpretation. *Economic and Mathematical Methods*, 8: 740–751, 1972 (in Russian).
7. J.S. Sosnowski. Linear programming via augmented Lagrangian and conjugate gradient methods. In S. Walukiewicz and A.P. Wierzbicki (Eds.), *Methods of Mathematical Programming*, Proceedings of a 1977 Conference in Zakopane. Polish Scientific Publishers, Warsaw, 1981.
8. D.P. Bertsekas. Multiplier methods: a survey. *Automatica*, 12: 133–145, 1976.
9. B.A. Murtagh and M.A. Sanders. MINOS – A large-scale nonlinear programming system (for problems with linear constraints). User guide. Technical Report, Systems Optimization Laboratory, Stanford University, 1977.
10. B.T. Polyak. The conjugate gradient method in extremal problems. *Computational Mathematics and Mathematical Physics*, 9: 94–112, 1969.
11. D.P. O'Leary. A generalized conjugate gradient algorithm for solving a class of quadratic problems. *Linear Algebra and its Applications*, 34: 371–399, 1980.
12. M.R. Hestenes. *Conjugate Gradient Methods in Optimization*. Springer Verlag, Berlin, 1980.

13 M.Makowski, J. Sosnowski *Hybrid: A mathematical programming package* , IIASA, 1984, CP-84-9.

14 A.R. Curtis, J.K. Reid *On the automatic scaling of matrices for Gaussian elimination* , Journal of Mathematics and its applications , 1972, no 10, pp.118–124.

15 J.A.Tomlin, *On scaling linear programming problems* , Mathematical Programming Study 4. North Holland Publishing Company, 1972, Amsterdam

16 J.S. Sosnowski, Dynamic optimization of multisectorial linear production model. Systems Research Institute, Warsaw, Ph.D. Thesis, (in Polish), 1978.

17 A. Propoi, Problems of Dynamic Linear Programming, IIASA, RM-76-78

18 R. Fourer, Solving staircase linear programs by the simplex method, 1,2. Mathematical Programming 23 (1982) 274-314, 25(1983) 3.01-292

19 J.K.Ho, A.S. Hanne, Nested decomposition for dynamic models. Mathematical Programming 6 (1974) 121-140

20 A. Lewandowski and Grauer M., The reference point optimization approach - methods of efficient implementation. CP-12-S12, IIASA Collaborative Proceedings Series: Multiobjective and Stochastic Optimization Proceedings of an IIASA Task Force Meeting

21 A.Wierzbicki ,A mathematical basis for satisficing decision making, WP-80-90, IIASA, 1980).

22 R.Flecher, Practical methods of optimization, vol II, Constrained optimization, Wiley, New York, 1981

23 A.Wierzbicki, On the use of penalty functions in multi- objective optimization, Institute of Automatics, Technical University of Warsaw, 1978.

24 B.A. Murtagh, Advanced Linear Programming: Computation and Practice, Mc Graw-Hill, New York, 1982.

25 Kreglewski, T., Lewandowski, A. and Rogowski, T. (1984) Dynamic Extension of the DIDAS system and its Application in Flood Control. In: Plural Rationality and Interactive Decision Processes. Eds. M. Grauer, M. Thompson, A.P. Wierzbicki. Springer - Verlag, 1985.

26 Murtagh B.A. and Sanders M.A., A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints, Mathematical Programming Study 16 (1982), 84-117

27 Makowski M., Sosnowski J., HYBRID 2.1: A mathematical programming package for multicriteria dynamic problems. In: A.Lewandowski, A. Wierzbicki eds., Theory Software and Testing Examples for Decision Support Systems, IIASA, Laxenburg, September 1985.

IAC-DIDAS-L A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Linear and Dynamic Linear Models on Professional Microcomputers

Tadeusz Rogowski, Jerzy Sobczyk, Andrzej P. Wierzbicki

Institute of Automatic Control, Warsaw University of Technology

ABSTRACT

This paper presents introductory documentation and a theoretical manual for two, professional microcomputer based, versions of decision analysis and support systems of DIDAS family. These versions have been developed in 1986, in the Institute of Automatic Control, Warsaw University of Technology, under a contracted study agreement with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis, and differ from previous DIDAS versions in several aspects. Both are implemented on professional microcomputers compatible with IBM-PC-XT (with a hard disk, Hercules or color graphics card and, preferably, a co-processor) and both support graphical representation of results in interactive analysis. However, the first version: IAC-DIDAS-L1, uses a linear programming solver written in FORTRAN, which results in relatively fast execution of optimization runs during interactive analysis but requires the preparation of the substantive model being analysed in the system, in the MPS-format. The second version: IAC-DIDAS-L2, is written in PASCAL and supports also an interactive definition and edition of the substantive model by the user, in a user-friendly format of a spreadsheet. Both versions are designed to work with substantive models of linear programming and dynamic linear programming type, that is, to perform and to graphically represent the results of interactive multiobjective analysis of such models.

A. INTRODUCTORY DOCUMENTATION

A1. EXECUTIVE SUMMARY

In many situations of complex decisions involving economic, environmental and technological decisions as well as in the cases of complex engineering design, the decision maker needs help of an analyst, or a team of analysts, to learn about possible decision options and their predicted results. The team of analysts frequently summarizes its knowledge in the form of a *substantive model* of the decision problem that can be formalized mathematically and computerized.

While such a model can never be perfect and cannot encompass all aspects of the problem, it is often a great help to the decision maker in the process of learning about novel aspects of the decision situation and of gaining expertise in handling problems of a given class. Even if the final decisions are typically made judgmentally - that is, are based on holistic, deliberative assessments of all available information without performing a calculative analysis of this information, see S.Dreyfus (1985) - the interaction of a decision maker with the team of analysts and the substantive models prepared by them can be of great value.

In organizing such interaction, many techniques of optimization, multicriteria decision analysis and other tools of mathematical programming can be used. To be of value for a holistically thinking decision maker, however, all such techniques must be used as supporting tools of interactive analysis rather than as means for proposing unique optimal decisions and thus replacing the decision maker. The decision analysis and support systems of DIDAS family - that is, Dynamic Interactive Decision Analysis and Support systems, see e.g. Lewandowski et al. (1984) - are especially designed to support interactive work with a substantive model while using multicriteria optimization tools, but they stress the learning aspects of such work, such as the right of a decision maker to change his priorities and preferences when learning new facts. DIDAS systems can be used either by analysts who want to analyse their substantive models, or by teams of analysts and decision makers, or even by decision makers working alone with a previously defined substantive model; in any case, we shall speak further about the user of the system.

There are several classes of substantive models that require special technical means of support. The IAC-DIDAS-L1 and -L2 versions are designed to support models of linear programming type; specifically, multiobjective linear programming models, often with dynamic structure. If a model has a multiobjective dynamic structure, the *objectives* (called also criteria, outcomes, results, etc.) of decisions form trajectories, which might be interpreted as graphs of the dependence of an objective on time or another variable of similar type; these trajectories are evaluated by the user as a whole, complex objective. The decisions can also have the form of trajectories.

Models of multiobjective linear programming type specify, firstly, the bounds on admissible *decision variables*, in the form of linear equations or inequalities called constraints (including,for models of dynamic type, also special constraints called state equations of the model) and, secondly, the attainable decision outcomes, in the form of linear equations for *outcome variables* among which the user can select his objectives. Actually, the distinction between constraints and outcome variables is not necessarily sharp (if the value of a constraint can be changed, it becomes an outcome variable) and the user might select his objectives also among constraint variables.

There are many examples of decision problems that can be analysed by means of a substantive model of multiobjective linear programming type; for example, DIDAS-type systems with multiobjective, dynamic linear programming models have been used in planning energy policies (see Strubegger, 1985, Messner, 1985), agricultural policies (see Makowski and Sosnowski, 1983) as well as in analysing various environmental or technological problems (see Kaden, 1985, Gorecki et al., 1983). As demonstrative or tutorial examples, IAC-DIDAS-L1 and -L2 use a multiobjective linear programming model for a problem of diet composition (see Appendix), where the decision variables correspond to various dishes and the constraints or outcomes correspond to the amount of vitamins, minerals, the cost and subjectively defined taste and stimulus of the diet; another example might be a dynamic multiobjective linear programming model for flood control with several tributaries of a river and several reservoirs, where the decisions are time sequences - trajectories - of outflows of reservoirs and the outcomes are trajectories of flows in various points on the river. The user can also define substantive models of multiobjective (possibly dynamic) linear programming type for his own problems and analyse them with the help of IAC-DIDAS-L1 or -L2.

A typical procedure of working with a DIDAS-type system consists of several phases.

In the first phase, a user - typically, an analyst - defines the substantive model and edits it on the computer. In earlier versions of DIDAS-type systems (which were mostly implemented on bigger mainframe computers) this phase has not been explicitly supported in the system and the user had to separately prepare (define and edit) his model in the MPS format. This is a typical format for single-objective linear programming problems and can be also used for multiobjective problems; however, working with MPS format requires some knowledge of linear programming and thus limits the use of such DIDAS systems to rather experienced analysts. On the other hand, there are many existing linear programming models in the MPS format that could be analysed multiobjectively with the help of a DIDAS system. Therefore the version IAC-DIDAS-L1 has been designed to work with substantive models in the MPS format while the user-friendliness of professional microcomputers compatible with IBM-PC-XT is exploited only in the graphical representation of results of multiobjective analysis.

The second version: IAC-DIDAS-L2, exploits the user-friendliness of such microcomputers also by supporting the definition and edition of a substantive model in an easy format of a spreadsheet, where the decision variables (and, possibly, some model parameters) are represented by the columns, the constraints and outcome variables - by the rows of the spreadsheet, and the coefficients of all linear functions defining the model are entered in the corresponding cells of the spreadsheet. Therefore, the user can define, review and edit his model easily; when analysing his model in further phases of work with IAC-DIDAS-L2, he can also return to the model definition phase and modify his model if necessary. The user of IAC-DIDAS-L2 can also have several substantive models recorded in a special model directory, use old models from this directory to speed up the definition of a new model, etc., while the system supports automatically the recording of all new or modified models in the directory. The easiness of model definition and edition has, however, its price: models defined in the spreadsheet format should not be too large and the number of their variables (decision variables, constraints and outcome variables, while counting separately variables for each time instant in dynamic models) should not be too large (not greater than a hundred).

In the second phase of work with DIDAS-type systems, the user - here typically an analyst working together with the decision maker - specifies a multiobjective analysis problem related to his substantive model and participates in an initial analysis of this problem. There might be many multiobjective analysis problems related to the same substantive model: the specification of a multiobjective problem consists in designating outcome and constraint variables in the model that become objectives (or objective trajectories in a dynamic case) and defining whether an objective (or objective trajectory) should be minimized or maximized, or kept close to a given level. For a given definition of the multiobjective analysis problem, the decision and outcomes in the model are subdivided into two categories: those that are *efficient* with respect to the multiobjective problem (that is, such that no objective can be improved without deteriorating some other objective) and those that are inefficient. It is assumed that the user is interested only in efficient decisions and outcomes (this assumption is reasonable provided that the user has listed all objectives of his concern; if he has not, or if some objectives of his concern are not represented in the model he can still modify the sense of efficiency by adding new objectives, or by requiring some objectives to be kept close to given levels, or by returning to the model definition phase and modifying the model).

One of the main functions of a DIDAS-type system is to compute efficient decisions and outcomes - following interactively various instructions of the user - and to present them for analysis. This is done by solving a special parametric linear programming problem resulting from the specification of the multiobjective analysis problem; for this purpose, IAC-DIDAS-L contains a specialized linear programming algorithm called *solver*.

Usually, however, the definition of a multiobjective problem admits many efficient decisions and outcomes; therefore the user should first learn about ***bounds on efficient outcomes***. This is the main function of IAC-DIDAS-L in the initial analysis phase. The user can request the system to optimize any objective separately; however, there are also two special commands in the system, related to this function. The first, called *"utopia"*, results in subsequent computations of the best possible outcomes for all objectives treated separately (such outcomes are practically never attainable jointly, hence the name "utopia" for the point in outcome space composed of such outcomes; in dynamic cases, only approximate joint bounds for entire trajectories are computed). The second, called *"nadir"*, results in an estimation of the worst possible among the efficient outcomes (defining precisely the worst possible efficient outcome is a very difficult computational task; in some simple cases, the "utopia" computations give enough information to determine the worst possible among the efficient outcomes, but for more general cases this information is not reliable and a more reliable way of estimating the worst possible efficient outcome is implemented in IAC-DIDAS-L).

The "utopia" and "nadir" computations give important information to the user about reasonable ranges of decision outcomes; in order to give him also information about a reasonable compromise efficient solution, a *neutral efficient solution* can be also computed in the initial analysis phase following a special command. The neutral solution is an efficient solution situated "in the middle" of the range of the efficient outcomes, while the precise meaning of being "in the middle" is defined by the distances between the utopia and the nadir point. After analysing the utopia point, the nadir point and a neutral solution (which all can be represented graphically for the user), the initial analysis is completed and the user has already learned much about the ranges of the attainable efficient objectives and the possible trade-offs between these objectives. Each change of the definition of the substantive model or of the multiobjective analysis problem, however, necessitates actually a repetition of the initial analysis phase; on the other hand, the user can omit this repetition if he judges that the changes in the model or in multiobjective analysis definition have been small.

The third phase of work with DIDAS-type systems consists in interactive scanning of efficient outcomes and decisions, guided by the user through specifying *aspiration levels* for each objective (or *aspiration trajectories* , in a dynamic case; called also *reference points* or *trajectories*). The user has already reasonable knowledge about the range of possible outcomes and thus he can specify the aspiration levels that he would like to attain. IAC-DIDAS-L utilizes the aspiration levels as a parameter in a special achievement function, coded in the system, uses its solver to compute the solution of a linear programming problem, equivalent to maximizing this achievement function, and responds to the user with an attainable efficient solution and outcomes (or outcome trajectories) that strictly correspond to the user-specified aspirations.

If the aspirations are "too high" (better than attainable), then the response of the system is a solution with attainable, efficient outcomes that are uniformly as close to the aspirations as possible. If the aspirations are "too low" (if they correspond to attainable but inefficient outcomes that can be improved), then the response of the system is a solution with outcomes that are uniformly better than the aspirations. The precise meaning of the uniform approximation or improvement depends on *scaling units* for each objective that can be either specified by the user or defined automatically in the system as the differences between the utopia point and the current aspiration point. This second, automatic definition of scaling units has many advantages to the user who is not only relieved of specifying scaling units but also has a better control of the selection of efficient outcomes by changing aspiration levels in such a case.

After scanning several representative efficient solutions and outcomes controlled by changing aspirations, the user usually learns enough to select either an actual decision, subjectively, (which needs not to correspond to the decisions proposed in the system, since even the best substantive model might differ from real decision situation) or an efficient decision and outcome proposed in the system as a basis for actual decisions.

Rarely, the user might be still uncertain about what decision to choose; for such a case, several additional options can be included in a system of DIDAS type. Such options include two more sophisticated scanning options: ***multidimensional scanning***, resulting from perturbing current aspiration levels along each coordinate of objective space, ***directional scanning***, resulting from perturbing current aspiration levels along a direction specified by the user (see Korhonen, 1985). Another option is ***forced convergence***, that is, such changes of aspiration levels along subsequent directions specified by the user that the corresponding efficient decisions and outcomes converge to a final point that might represent the best solution for the preferences of the user. However, not all these additional options are implemented in IAC-DIDAS-L, since the experience of working with DIDAS-type systems shows that these options are rarely used.

A2. SHORT PROGRAM DESCRIPTION

The IAC-DIDAS-L1 and -L2 systems (Institute of Automatic Control, Dynamic Interactive Decision Analysis and Support, Linear versions 1 and 2) are decision support systems designed to help in the analysis of decision situations where a mathematical model of substantive aspects of the situation can be formulated in the form of a multiobjective linear programming problem, possibly of dynamic structure.

The IAC-DIDAS-L1 and -L2 systems are recorded on two separate diskettes that should be installed on an IBM-PC-XT or a compatible computer with a hard disk, Hercules or a color graphic card and, preferably, a coprocessor. Both diskettes contain compiled codes, partly in FORTRAN and partly in PASCAL for IAC-DIDAS-L1, and entirely in PASCAL for IAC-DIDAS-L2. After installing them in the users directory, they can be activated (by the command didasl or didas2 Cr) and used in a program system. Both systems support the following general functions:

1) The definition and edition of a substantive model of the decision situation, in a linear programming form. IAC-DIDAS-L1 uses the MPS format of linear programming for this purpose, while IAC-DIDAS-L2 supports model definition and edition in a user-friendly format of a spreadsheet.
2) The specification of a multiobjective decision analysis problem related to the substantive model. This is performed by several commands from the main menu of IAC-DIDAS-L1, and by specific features of spreadsheet edition in IAC-DIDAS-L2.
3) The initial multiobjective analysis of the problem, resulting in estimating bounds on efficient outcomes of decisions and in learning about some extreme and some neutral decisions. In both IAC-DIDAS-L1 and -L2, these functions are supported by some specific commands from the main menu.
4) The interactive analysis of the problem with the stress on learning by the user of possible efficient decisions and outcomes, organized through systems' response to user-specified aspiration levels or reference points for objective outcomes. In both IAC-DIDAS-L1 and -L2, the system responds with efficient solutions and objective outcomes obtained through the maximization of an achievement function that is

parameterized by the user-specified reference points. The maximization is performed through a linear programming algorithm called solver, written in FORTRAN for IAC-DIDAS-L1 and in PASCAL for IAC-DIDAS-L2. In both systems, the interactive analysis is supported by specific commands from the main menu, including commands that might help in convergence to the most preferred solution; however, the main function of both systems is helping the user to learn about novel aspects of the decision situation, not necessarily forcing him to converge to one, most preferred solution.

The main menu of commands in IAC-DIDAS-L1 is the following:

1)Problem setting phase

? Cr - displays help.

MAX | MIN | GUI | FLO | REM objectivename Cr - includes new objectives (from the list of names of outcome and decision variables of the model), changes status (to maximized, minimized, guided - that is, corresponding to an equality constraint, or floating - that is, displayed only for information purposes) or removes an objective from the definition of the multiobjective analysis problem.

UPP | LOW | FIX objectivename value Cr - sets bounds for objective values (UPP for upper bounds, LOW for lower bounds, FIX for equality constraints of GUI type; all objectives except of GUI and FLO types must have specified bounds in this phase; defaults are zero and rhs or bounds - as specified in the model).

SCA objectivename value Cr - sets user-specified scaling units for an objective (all objectives except of GUI and FLO types must have specified scaling units in this phase; default is 1).

RAS binary (0 or 1) Cr - sets off or on automatic utopia- reference scaling (after computing utopia point, see further commands, the user-supplied scaling can be replaced by a more convenient type of scaling).

EPS value Cr - sets the value of parameter $0<eps<1$ in the achievement function.

XRH value Cr - sets the value of parameter $\rho>1$ in the achievement function.

EPS | XRH Cr - displays the value of parameter eps or rho.

2) Initial analysis phase

FOR objectivename Cr - results in the calculation and graphical display of an extreme solution, that is, the optimal solution for a given, single objective.

UTO Cr - calculates and displays graphically utopia and approximate nadir points (that is, upper and lower bounds for efficient decision outcomes).

NAD Cr - improves and displays graphically the approximation of nadir point.

NEU Cr - calculates and displays graphically a neutral solution using scaling coefficients based on utopia-nadir differences.

GRA Cr - graphic displays.

REU Cr - changes scale of graphical displays to utopia-nadir relative.

REB Cr - changes scale of graphical displays to relative to bounds. *3) Interactive analysis phase*

RFP | REF objectivename value (%) Cr - sets reference point for an objective (if the option % is used, this point is given in % of current graphical display scale).

GO Cr - calculates and displays graphically an efficient solution related to the last specified reference point.

DIS BOU | UTO | SOL | Cr - displays numerically bounds, or utopia and nadir points, or the last solution.

SCN value Cr - starts the SCAN procedure with the step d = 'value'.

ACC objname Cr - accepts the solution obtained during the SCAN process, when the reference point component corresponding to 'objname' was perturbed, as a new reference point.

PRI Cr - writes the last results on the file RESULTS.

PSC Cr - writes the results of the last scan on the file RESULTS.

BAS Cr - makes possible manipulating with the data base for solution (up to 10 items). After invoking this command the following menu appears at the screen:

(1) save (2) load (3) remove (4) list (5) quit.

The user ought to select the option number:

- option (1) save - at this point the program asks:
  save as ?:
  and the user gives a name to the last solution to be saved in the data base,
- option (2) load - at this point the user gives the names of the data and the solution to be retrieved from the data base,
- option (3) remove - removes a name from the data base,
- option (4) list - lists the names saved in the data base,
- option (5) quit - returns to the main menu.

STOP Cr - ends work with the system.

The main menu of IAC-DIDAS-L2 performs also all the above functions, with the distinction that most of the functions of phase 1) and 2) are specific commands of spreadsheet edition: the decision variables are defined as columns of the spreadsheet, the outcome variables are defined as rows, model coefficients are entered in the corresponding cells, there are special rows and columns for scaling units, lower and upper bounds, for defining objective outcomes and their type, for reference points, utopia and nadir points, for solutions corresponding to the reference points. The data for tutorial example, contained in the Appendix, is illustrated by several screen outprints that are related to various functions of model edition in IAC-DIDAS-L2. The functions of other phases are executed by macrocommands using various controlling keys; the user can get various help displays that suggest in an easy fashion the commands useful in a current phase of work with the system.

IAC-DIDAS-L1 and -L2 systems have been developed in the Institute of Automatic Control, Warsaw University of Technology, Warsaw, Poland, in a contracted study agreement "Theory, Software and Testing Examples for Decision Support Systems" with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis, Laxenburg, Austria, which has the copyright for these systems.

B. THEORETICAL MANUAL

The standard form of a multiobjective linear programming problem is defined as follows:

$$\textit{maximize}\ (q{=}Cx);\ X = \{x{\in}R^n{:}\ Ax{=}b, x{\geq}0\} \tag{1}$$

where $x \in R^n$, $b \in R^p$, A is a $m \times n$ matrix, C is a $p \times n$ matrix and the maximization of the vector q of p objectives is understood in the Pareto sense: $\hat{x},\hat{q}$ are solutions of (1) iff $\hat{q}=C\hat{x}$, $\hat{x}\in X$ and there are no such x,q, with $q=Cx$, $x\in X$ that $q \geq \hat{q}$, $q \neq q$. Such solutions $\hat{x}$ and $\hat{q}$ of (1) are called an efficient decision $\hat{x}$ and the corresponding efficient outcome $\hat{q}$, respectively. If, in the above definition, it were only required that there would be no x and q, with $q=Cx$, $x\in X$, such that $q>\hat{q}$, then the solutions $\hat{x}$, $\hat{q}$ would be called *weakly efficient*. Equivalently, if the set of all attainable outcomes is denoted by

$$Q=\{q\in R^p:\ q=Cx,\ x\in X\} \tag{2}$$

and so called *positive cones* $D=R^p_+$, $\tilde{D}=R^p_+\backslash\{0\}$ and $\tilde{\tilde{D}}=intR^p_+$ are introduced (thus, $q\geq\hat{q}$ can be written as $q-\hat{q}\in D, q\geq\hat{q}, q\neq\hat{q}$ as $q-\hat{q}\in\tilde{D}$ and $q>\hat{q}$ as $q-\hat{q}\in\tilde{\tilde{D}}$ then the sets of efficient outcomes $\hat{Q}$ and of weakly efficient outcomes $\hat{Q}^w$ can be written as:

$$\hat{Q} = \{\hat{q}\in Q:\ (\hat{q}+\tilde{D})\cap Q=\phi\} \tag{3}$$

$$\hat{Q}^w = \{\hat{q}\in Q:\ (\hat{q}+\tilde{\tilde{D}})\cap Q=\phi\} \tag{4}$$

The set of weakly efficient outcomes is larger and contains the set of efficient outcomes; in many practical applications, however, the set of weakly efficient outcomes is decisively too large. For multiobjective linear programming problems, the efficient outcomes are always *properly efficient*, that is, they have bounded *tradeoff coefficients* that indicate how much an objective outcome should be deteriorated in order to improve another objective outcome by a unit.

The *abstract problem of multiobjective linear programming* consists in determining the entire sets $\hat{Q}$ or $\hat{Q}^w$, or at least all vertices or basic solutions of the linear programming problem that corresponds to efficient decisions and outcomes.

The *practical problem of multiobjective decision support*, using linear programming models, is different and consists in computing and displaying for the decision maker (or, generally, for the user of the decision support system) some selected efficient decisions and outcomes. This selection of efficient decisions and outcomes should be easily controlled by the user and should result in any efficient outcome in the set Q he might wish to attain, in particular, also in efficient outcomes that are not necessarily basic solutions of the original linear programming problem; moreover, weakly efficient outcomes are not of practical interest for the user.

Before turning to some theoretical problems resulting from these practical requirements, observe first that the standard formulation of multiobjective linear programming is not the most convenient for the user. Although many other formulations can be rewritten to the standard form by introducing proxy variables, such reformulations should not bother the user and should be automatically performed in the decision support system. Therefore, we present here another basic formulation of the multiobjective linear programming problem, more convenient for typical applications.

A *substantive model* of multiobjective linear programming type consists of the specification of vectors of n decision variables $x\in R^n$ and of m outcome variables $y\in R^m$, together with linear *model equations* defining the relations between the decision variables and the outcome variables and with *model bounds* defining the lower and upper bounds for all decision and outcome variables:

$$y=Ax;\ x^{lo}\leq x\leq x^{up};\ y^{lo}\leq y\leq y^{up} \tag{5}$$

where A^c is a $m\times n$ matrix of coefficients. Among the outcome variables, some might be chosen as corresponding to *equality constraints*; let us denote these variables by

$y^c \in R^{m'} \subset R^m$ and the constraining value for them - by b^c and let us write the additional constraints in the form:

$$y^c = A^c x = b^c;\ y^{c,lo} \leq b^c \leq y^{c,up} \tag{6}$$

where A^c is the corresponding submatrix of A. The outcome variables corresponding to equality constraints will be called ***guided outcomes*** here. Some other outcome variables can be also chosen as optimized objectives or ***objective outcomes***. Denote the vector of p objective outcomes by $q \in R^p \subset R^m$ (some of the objective variables might be originally not represented as outcomes of the model, but we can always add them by modifying this model) to write the corresponding objective equations in the form:

$$q = Cx \tag{7}$$

where C is another submatrix of A. Thus, the set of attainable objective outcomes is again $Q = CX$, but the set of admissible decisions X is defined by:

$$X = \{x \in R^n:\ x^{lo} \leq x \leq x^{up};\ y^{lo} \leq Ax \leq y^{up};\ A^c x = b^c\} \tag{8}$$

Moreover, the objective outcomes are not necessarily minimized; some of them might be minimized, some maximized, some stabilized or kept close to given ***aspiration levels*** (that is, minimized if their value is above aspiration level and maximized if their value is below aspiration level). All these possibilities can be summarized by introducing a different definition of the positive cone D:

$$D = \{q \in R^p:\ q_i \geq 0,\ i=1,..,p';\ \ q_i \leq 0,\ i=p'^{+1},..,p'';\ \ q_i = 0, i=p'',..,p\} \tag{9}$$

where the first p' objectives are to be maximized, the next, from p'^{+1} to p'', are to be minimized, and the last, from $p''+1$ to p, are to be stabilized. Actually, the user needs only to define what to do with subsequent objectives; the concept of the positive cone D is used here only in order to define comprehensively what are efficient outcomes for the multiobjective problem. Given some aspiration levels for stabilized objectives and the requirement that these objectives should be minimized above and maximized below aspiration levels, the set of efficient outcomes can be defined only relative to the aspiration levels.

However, since the user can define aspiration levels arbitrarily, of interest here is the union of such relative sets of efficient outcomes. Let $\tilde{D} = D \setminus \{0\}$; then the outcomes that might be efficient for arbitrary aspiration levels for stabilized objectives can be defined, as before, by the relation (3). The weakly efficient outcomes are of no practical interest in this case, since the cone D, typically, has empty interior which implies that weakly efficient outcomes coincide with all attainable outcomes.

The stabilized outcomes in the above definition of efficiency are, in a sense, similar to the guided outcomes; however, there is an important distinction between these two concepts. Equality constraints must be satisfied; if not, then there are no admissible solutions for the model. Stabilized objective outcomes should be kept close to aspiration levels, but they can differ from those levels if, through this difference, other objectives can be improved. The user of a decision support system should keep this distinction in mind and can modify the definition of the multiobjective analysis problem by taking, for example, some outcomes out of the guided outcome category and putting them into the stabilized objective category.

By adding a number of proxy variables and changing the interpretation of matrix A, the substantive model formulation (5), (6), (7), (8) together with its positive cone (9) and the related concept of efficiency could be equivalently rewritten to the standard form of multiobjective linear programming (1); this, however, does not concern the user. More important is the way of user-controlled selection of an efficient decision and outcome from the set (3). For stabilized objective outcomes, the user can change the related aspiration levels in order to influence this selection; it is assumed here that he will use, for all objective outcomes, the corresponding aspiration levels in order to influence the selection of efficient decisions. The aspiration levels are denoted here $\bar{q}_i$ or, as a vector, $\bar{q}$ and called also, equivalently, *reference points.*

A special way of parametric scalarization of the multiobjective analysis problem is utilized for the purpose of influencing the selection of efficient outcomes by changing reference points. This parametric scalarization is obtained through maximizing the following *order-approximating achievement function* (see Wierzbicki 1983, 1986):

$$s(q,\bar{q})=min\left[\min_{1\leq i\leq p} z_i(q_i,\bar{q}_i),\ \frac{1}{\rho p}\sum_{i=1}^{p} z_i(q_i,\bar{q}_i)\right]+\frac{\epsilon}{p}\sum_{i=1}^{p} z_i(q_i,\bar{q}_i) \tag{10}$$

where the parameter ϵ should be positive, even if very small; if this parameter would be equal to zero, then the above function would not be order-approximating any more, but *order-representing*, and its maximal points could correspond to weakly efficient outcomes. The parameter ρ should be $\rho\geq 1$; the interpretation of both these parameters is given later.

The functions $z_i(q_i,\bar{q}_i)$ are defined as follows:

$$z_i(q_i,\bar{q}_i)=\begin{cases}(q_i-\bar{q}_i)/s_i, & \text{if } 1\leq i\leq p', \\ (\bar{q}_i-q_i)/s_i, & \text{if } p'^{+1}\leq i\leq p'', \\ \min(z'^i,z_i''), & \text{if } p''+1\leq i\leq p\end{cases} \tag{11}$$

where

$$z'^i=(q_i-\bar{q}_i)/s'^i,\quad z_i''=(\bar{q}_i-q_i)/s_i'' \tag{12}$$

The coefficients s_i,s'^i and s_i'' are scaling units for all objectives, either defined by the user (in which case $s'^i=s_i''$, the user does not need to define two scaling coefficients for a stabilized objective outcome) or determined automatically in the system (see further comments).

The achievement function $s(q,\bar{q})$ is maximized with $q=Cx$ over $x\in X$; its maximization in the system is converted automatically to an equivalent linear programming problem, different than the original one, and having more basic solutions that depend on the parameter $\bar{q}$. If the coefficient $\epsilon>0$, then the achievement function has the following properties (see Wierzbicki, 1986):

a) For an arbitrary aspiration level or reference point $\bar{q}$, not necessarily restricted to be attainable or not attainable, each maximal point $\hat{q}$ of the achievement function $s(q,\bar{q})$ with $q=Cx$ over $x\in X$ is a D_ϵ -efficient solution,that is, a properly efficient solution with tradeoff coefficients bounded approximately by ϵ and $1/\epsilon$.

b) For any properly efficient outcome $\hat{q}$ with trade-off coefficients bounded by ϵ and $1/\epsilon$, there exist such reference points $\bar{q}$ that the maximum of the achievement function $s(q,\bar{q})$ is attained at the properly efficient outcome $\hat{q}$. In particular, if the user (either by chance or as a result of a learning process) specifies a reference point $\bar{q}$ that in itself is such properly efficient outcome, $\bar{q}=\hat{q}$, then the maximum of the therefore, it is called the ***utopia point*** $\hat{q}^{uto}$.

However, this way of computing the 'upper' bound for efficient outcomes is not practical for problems of dynamic structure (see further comments); thus, IAC-DIDAS-L1 and -L2 use a different way of estimating the utopia point. This way consists in subsequent maximizations of the achievement function $s(q,\bar{q})$ with suitably selected reference points. If an objective should be maximized and its maximal value must be estimated, then the corresponding component of the reference point should be very high, while the components of this point for all other maximized objectives should be very low (for minimized objectives - very high; stabilized objectives must be considered as floating in this case that is, should not enter the achievement function). If an objective should be minimized and its minimal value must be estimated, then the corresponding component of the reference point should be very low, while other components of this point are treated as in the previous case. If an objective should be stabilized and both its maximal and minimal values must be estimated, then the achievement function should be maximized twice, first time as if for a maximized objective and the second time as if for minimized one. Thus, the entire number of optimization runs in utopia point computations is $p''+2(p-p'')$. It can be shown that, for problems with static structure (no trajectory objectives), this procedure gives a very good approximation of the utopia point $\hat{q}^{uto}$, whereas the precise meaning of 'very high' reference should be interpreted as the upper bound for the objective plus, say, twice the distance between the lower and the upper bound, while the meaning of 'very low' is the lower bound minus twice the distance between the upper and the lower bound.

During all these computations, the lower bound for efficient outcomes can be also estimated, just by recording the lowest efficient outcomes that occur in subsequent optimizations for maximized objectives and the highest efficient outcomes for minimized objectives (there is no need to record them for stabilized objectives, where the entire attainable range is estimated anyway). However, such a procedure results in the accurate, tight 'lower' bound for efficient outcomes - called ***nadir point*** $\hat{q}^{nad}$ - only if $p''=2$; for larger numbers of maximized and minimized objectives, this procedure can give misleading results, while an accurate computation of the nadir point becomes a very cumbersome computational task.

Therefore, IAC-DIDAS-L1 and -L2 offer an option of improving the estimation of the nadir point in such cases. This option consists in additional p'' maximization runs for achievement function $s(q,\bar{q})$ with reference points $\bar{q}$ that are very low, if the objective in question should be maximized, very high for other maximized objectives, and very low for other minimized objectives, while stabilized objectives should be considered as floating. If the objective in question should be minimized, then the corresponding reference component should be very high, while other reference components should be treated as in the previous case. By recording the lowest efficient outcomes that occur for maximized objectives in subsequent optimizations (and are lower than the previous estimation of nadir component) and the highest efficient outcomes for minimized objectives (higher that the previous estimation of nadir component), a better estimation $\hat{q}^{nad}$ of the nadir point is obtained.

Once the approximate bounds $\hat{q}^{uto}$ and $\hat{q}^{nad}$ are computed and known to the user, they can be utilized in various ways. One way consists in computing a ***neutral efficient solution***, with outcomes situated approximately 'in the middle' of the efficient set. For this purpose, the reference point $\bar{q}$ is situated at the utopia point $\hat{q}^{uto}$ (only for maximized or minimized outcomes; for stabilized outcomes, the user-supplied reference component $\bar{q}_i$ must be included here) and the scaling units are determined by:

$$s_i=|\hat{q}_i^{uto}-\hat{q}_i^{nad}|,\ 1\leq i\leq p \tag{13a}$$

for maximized or minimized outcomes, and:

$$\begin{aligned} s_i'&=\bar{q}_i-\hat{q}_i^{nad}-0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}).\\ s_i''&=\hat{q}_i^{uto}+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad})-\bar{q}_i,\ \ p''+1\leq i\leq p \end{aligned} \tag{13b}$$

for stabilized outcomes, while the components of the utopia and the nadir points are interpreted respectively as the maximal and the minimal value of such an objective; the correction by $0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad})$ ensures that the scaling coefficients remain positive, if the user selects the reference components for stabilized outcomes in the range $\hat{q}_i^{nad}\leq q_i\leq \hat{q}_i^{nad}$ (if he does not, the system automatically projects the reference component on this range). By maximizing the achievement function $s(q,\bar{q})$ with such data, the neutral efficient solution is obtained and can be utilized by the user as a starting point for further interactive analysis of efficient solutions.

In further interactive analysis, an important consideration is that the user should be able to influence easily the selection of the efficient outcomes $\hat{q}$ by changing the reference point $\bar{q}$ in the maximized achievement function $s(q,\bar{q})$. It can be shown (see Wierzbicki, 1986) that best suited for this purpose is the choice of scaling units determined by a difference between the slightly displaced utopia point and the current reference point:

$$s_i=\begin{cases}\hat{q}_i^{uto}+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad})-\bar{q}_i, & \text{if } 1\leq i\leq p'\\ \bar{q}_i-\hat{q}_i^{uto}-0.01(\hat{q}_i^{nad}-\hat{q}_i^{uto}), & \text{if } p'+1\leq i\leq p'',\end{cases} \tag{14a}$$

for maximized or minimized outcomes. For stabilized outcomes, the scaling units are determined somewhat differently than in (13b):

$$\begin{aligned} s_i'&=\hat{q}_i^{uto}+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad})-\bar{q}_i,\\ s_i''&=\bar{q}_i-\hat{q}_i^{uto}-0.01(\hat{q}_i^{nad}-\hat{q}_i^{uto})\ \text{ if } p'+1\leq i\leq p'', \end{aligned} \tag{14b}$$

It is assumed now that the user selects the reference components in the range $\hat{q}_i^{nad}\leq q_i\leq \hat{q}_i^{uto}$ or $\hat{q}_i^{uto}\leq q_i\leq \hat{q}_i^{nad}$ (if he does not, the system automatically projects the reference component on these ranges) for all objectives. Observe that, similarly as in the case of the neutral solution, the scaling units are determined automatically once the utopia, nadir and reference points are known; the user is not bothered by their definition. The interpretation of the above way of setting scaling units is that the user attaches implicitly more importance to reaching a reference component if he places it close to the known utopia component; in such a case, the corresponding scaling unit becomes smaller and the corresponding objective component is weighted stronger in the achievement function $s(q,\bar{q})$. Thus, this way of scaling, *relative to utopia-reference difference*, is taking into account the implicit information, given by the user, specified by the relative position of the reference point.

When the relative scaling is utilized, the user can easily obtain - by moving suitably reference points - efficient outcomes that are either situated close to the neutral solution, in the middle of efficient outcome set $\hat{Q}$, or in some remote parts of the set $\hat{Q}$, say, close to various extreme solutions.

Typically, several experiments of computing such efficient outcomes give enough information for the user to select an actual decision - either some efficient decision suggested by the system, or even a different one, since even the best substantive model cannot encompass all aspects of a decision situation. However,there might be some cases in which the user would like to receive further support - either in analysing the sensitivity of a selected efficient outcome, or in converging to some best preferred solution and outcome.

For analysing the sensitivity of an efficient solution to changes in the proportions of outcomes, a *multidimensional scan* of efficient solutions is implemented in IAC-DIDAS-L1 and -L2. This operation consists in selecting an efficient outcome, accepting it as a base $\bar{q}^{bas}$ for reference points, and performing p″ additional optimization runs with the reference points determined by:

$$\bar{q}_j = \bar{q}_j^{bas} + \gamma(\hat{q}_j^{uto} - \hat{q}_j^{nad}); \quad \bar{q}_i = \bar{q}_i^{bas} \ i \neq j, \ 1 \leq j \leq p'' \tag{15}$$

where γ is a coefficient determined by the user, $-1 \leq \gamma \leq 1$; if the relative scaling is used and the reference components determined by (15) are outside the range $\hat{q}_j^{nad}$, $\hat{q}_j^{uto}$, they are projected automatically on this range. The reference components for stabilized outcomes are not perturbed in this operation (if the user wishes to perturb them, he might include them, say, in the maximized category). The efficient outcomes, resulting from the maximization of the achievement function $s(q,\bar{q})$ with such perturbed reference points, are typically also perturbed, mostly along their subsequent components, although other their components might also change.

For analysing the sensitivity of an efficient solution when moving along a direction in the outcome space - and also as a help in converging to a most preferred solution - a directional scan of efficient outcomes is implemented in IAC-DIDAS-L1 and -L2. This operation consists again in selecting an efficient outcome, accepting it as a base $\bar{q}^{bas}$ for reference points, selecting another reference point $\bar{q}$, and performing a user-specified number K of additional optimizations with reference points determined by:

$$\bar{q}(k) = \bar{q}^{bas} + \frac{k}{K}(\bar{q} - \bar{q}^{bas}), \ 1 \leq k \leq K \tag{16}$$

The efficient solutions $\hat{q}(k)$, obtained through maximizing the achievement function $s(q,q(k))$ with such reference points, constitute a cut through the efficient set $\hat{Q}$ when moving approximately in the direction $\bar{q} - \bar{q}^{bas}$. If the user selects one of these efficient solutions, accepts it as a new $\bar{q}^{bas}$ and performs the next directional scans along some new directions of improvement, he can converge eventually to his most preferred solution (see Korhonen, 1985). Even if he does not wish the help in such convergence, the directional scans can give him valuable information.

Another possible way of helping in convergence to the most preferred solution is choosing reference points as in (16) but using a harmonically decreasing sequence of coefficients (such as $1/j$, where j is the iteration number) instead of user-selected coefficients k/K. This results in convergence even if the user makes stochastic errors in determining next directions of improvement of reference points, or even if he is not sure about his preferences, and learns about them during this analysis (see Michalevich, 1986). Such a convergence, however, is rather slow and is thus not implemented in IAC-DIDAS-L1 and -L2.

A separate problem is multiobjective decision analysis and support based on substantive models of dynamic structure. A useful standard of defining a substantive model of multiobjective linear dynamic programming type is as follows.

The model is defined on $T+1$ discrete time periods t, $0 \leq t \leq T$ (where t is a discrete time variable counted in days, years or any other time units; models of dynamic structure can also have other interpretations of the variable t, such numbers of subsequent operations, etc). The decision variable x, called in this case *control trajectory*, is an entire sequence of decisions:

$$x = \{x(0),x(1),....,x(T-1)\}\in R^{nT},\ x(t)\in R^{n} \tag{17b}$$

and a special type of outcome variables, called *state variables*, $w(t)\in R^{m'}$, is also considered. The entire sequence of state variables, or *state trajectory*:

$$w = \{w(0),w(1),....,w(T-1)\}\in R^{m'}(T+1) \tag{17b}$$

is actually one time period longer than x; the initial state $w(0)$ must be specified as given data, while the decision $x(T)$ in the final period is assumed to influence the state $w(T+1)$ only, thereby of no interest for the interval $\{0,...,T\}$. This is because the fundamental equations of a substantive dynamic model have the form of *state equations*:

$$w(t+1) = A(t)w(t)+B(t)x(t);\ t=0,1,...T-1,\ w(0)\ -\ given \tag{18a}$$

The model *outcome equations* have, then, the form:

$$y(t)=C(t)w(t)+D(t)x(t),\ t=0,1,...,T-1;\ y(T)=C(T)w(T)\in R^{m''} \tag{18b}$$

and define the sequence of outcome variables, or *outcome trajectory*:

$$y=\{y(0),...,y(t),...,y(T-1),y(T)\}\in R^{m''}(T+1) \tag{17c}$$

The decision, state and outcome variables can all have their corresponding lower and upper bounds (each understood as an appropriate sequence of bounds):

$$x^{lo}\le x\le x^{up},\ w^{lo}\le w\le w^{up},\ y^{lo}\le y\le y^{up} \tag{18c}$$

The matrices $A(t)$, $B(t)$, $C(t)$ and $D(t)$, of appropriate dimensions, can dependent on or can be independent of time t; in the latter case, the model is called *time invariant* (actually, in a fully time-invariant model, the bounds should also be independent of time t, that is, they should be constant for all time periods). This distinction is important, in multiobjective analysis of such models only in the sense of model edition: time-invariant models can be defined easier by automatic, repetitive edition of model equations and bounds for subsequent time periods.

Some of the outcomes might be chosen to be equality constrained, or *guided* along a given trajectory:

$$y^{c}(t)=e^{c}(t)\in R^{m'''}\subset R^{m''},\ t=0,1,...,T;\ e^{c} = \{e^{c}(0),...,e^{c}(T)\} \tag{19}$$

The optimized (maximized, minimized or stabilized) objective outcomes of such a model can be actually selected among both state variables and outcome variables (or even decision variables) of this model; in any case, they form an entire *objective trajectory*:

$$q=\{q(0),..,q(t),..,q(T-1),q(T)\}\in R^{p(T+1)},\ q(t)\in R^{p} \tag{20}$$

Various positive cones could be defined to specify the sense of efficiency of such objective trajectory; however, it is assumed here that the sense of efficiency cannot change along the trajectory, that is, a component $q_i(t)$ that will be maximized in one period t must be also maximized in other time periods, etc. (however, not necessarily in all time periods: if the user wishes to maximize,minimize or stabilize some outcome only in one or several time periods, he can always change suitably the definition of objective outcomes). Thus, assume that the first components $q_i(t)$, for $1\le i\le p'$, are to be maximized, next, for $p'+1\le i\le p''$, are to be minimized, and the last components, for $p''+1\le i\le p$, are to be stabilized. The achievement function $s(q,\bar{q})$ in such a case takes the form:

$$s(q,\bar{q})=\min\left\{\min_{0\leq t\leq T}\min_{1\leq i\leq p} z_i(t),\ \frac{1}{\rho(T+1)p}\sum_{t=0}^{T}\sum_{i=1}^{p} z_i(t)\right\}+\frac{\epsilon}{(T+1)p}\sum_{t=0}^{T}\sum_{i=1}^{p} z_i(t) \quad (21)$$

where the functions $z_i(t)=z_i[q_i(t),\bar{q}_i(t)]$ are defined by:

$$z_i(t)=\begin{cases}[q_i(t)-\bar{q}_i(t)]/s_i(t), & \text{if } 1\leq i\leq p' \\ [\bar{q}_i(t)-q_i(t)]/s_i(t), & \text{if } p'+1\leq i\leq p'' \\ \min[z_i'(t),z_i''(t)], & \text{if } p''+1\leq i\leq p,\end{cases} \quad (22)$$

where

$$z_i'(t)=\frac{q_i(t)-\bar{q}_i(t)}{s_i'(t)},\quad z_i''(t)=\frac{\bar{q}_i(t)-q_i(t)}{s_i''(t)}, \quad (23)$$

The user does not need to define time-varying scaling units $s_i(t)$ nor two different scaling units $s_i'(t),s_i''(t)$ for a stabilized objective: the time-dependence of scaling units and separate definitions of $s_i'(t),s_i''(t)$ are needed only in the case of automatic, relative scaling.

The estimation of utopia and nadir points in the space of objective trajectories would create, in the dynamic case, major computational difficulties ($p(T+1)$ subsequent optimization runs) if exact estimates were needed; moreover, even if the utopia point in itself is not attainable, it can be better interpreted if each of its components - in this case, each objective component trajectory - is attainable for the model. These considerations indicate that the way of estimating utopia point by p (or by $p''+2(p-p'')$, when stabilized objectives are included) subsequent maximizations of the achievement function (21) with suitably 'very high' or 'very low' components of *reference trajectories*:

$$\bar{q}=\{\bar{q}(0),\bar{q}(1),\ldots\ldots,\bar{q}(T)\}\in R^{p(T+1)},\quad \bar{q}(t)\in R^p \quad (24)$$

is much more adequate for the dynamic case than an exact computation of the utopia point. Denote the results of such maximizations with subsequent reference trajectories $\bar{q}^{(i)}$ by $\hat{q}^{(i)},i=1,\ldots,p$, (we do not include here stabilized outcomes for the simplicity of denotations); then the components of an approximate utopia trajectory can be determined as:

$$\hat{q}_i^{uto}(t)=\hat{q}_i^{(i)}(t),\quad t=0,1,\ldots,T;\quad i=1,2,\ldots,p \quad (25a)$$

whereas the components of an approximate nadir trajectory (in the case of maximized trajectories, with obvious modifications in the minimized case) should be determined as:

$$\hat{q}_i^{nad}(t)a=\min_{1\leq j\leq p}\hat{q}_i^{(j)}(t),\quad t=0,1,.,,,p \quad (25b)$$

Unfortunately, the components of such nadir approximation cannot be interpreted as attainable trajectories for the model (since the minimization in (25b) can result in different j for various t); however, this is less important than in the utopia trajectory case. A more precise approximation of nadir point can be obtained, similarly as in the static case, by additional p (or only p'', if stabilized objectives are included in the model) maximizations of achievement function (21) with yet other reference trajectories $\bar{q}^{(j)},j=p+1,\ldots,2p$, and by extending the minimization in (25b) to $1\leq j\leq 2p$.

Once the approximations of utopia and nadir trajectories are determined, a neutral solution as well as the automatic relative scaling can be defined similarly as in the static case. Other aspects of interactive multiobjective analysis of dynamic models are similar to the static case; naturally, the graphical representation of results of analysis is in some cases more straightforward (for single optimization runs) or, in other cases, more involved (for repetitive runs, as in utopia, nadir and scanning computations) than in the static case.

REFERENCES

Dreyfus, S. (1984): Beyond rationality. In M.Grauer, M.Thompson, A.P.Wierzbicki(eds), Plural Rationality and Interactive Decision Processes, Proceedings Sopron 1984. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 248).

Kaden, S. (1985): Decision support system for long-term water management in open-pit lignite mining areas. In G.Fandel, M.Grauer, A.Kurzhanski and A.P.Wierzbicki (eds), Large Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Korhonen, P. (1985): Solving discrete multiple criteria decision problems by using visual interaction. In G.Fandel, M.Grauer, A.Kurzhanski and A.P.Wierzbicki(eds), Large Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Lewandowski, A., M.Grauer, A.P.Wierzbicki (1983): DIDAS - theory, implementation and experiences. In M.Grauer, A.P.Wierzbicki (eds), Interactive Decision Analysis, Proceedings Laxenburg 1983. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 229).

Makowski, M. and J.Sosnowski (1984): A decision support system for planning and controlling agricultural production with a decentralized management structure. In M.Grauer, M.Thompson, A.P.Wierzbicki (eds), Plural Rationality and Interactive Decision Processes, Proceedings Sopron 1984. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 248).

Messner, S. (1985): Natural gas trade in Europe and interactive decision analysis. In G.Fandel, M.Grauer, A.Kurzhanski and A.P.Wierzbicki (eds), Large Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach 1985.Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Michalevich, M. (1986): Stochastic approaches to interactive multicriteria optimization problems. IIASA WP-86-10. International Institute for Applied Systems Analysis, Laxenburg, Austria. Wierzbicki, A.P. (1983): A mathematical basis for satisficing decision making. Mathematical Modelling 3, 391-405.

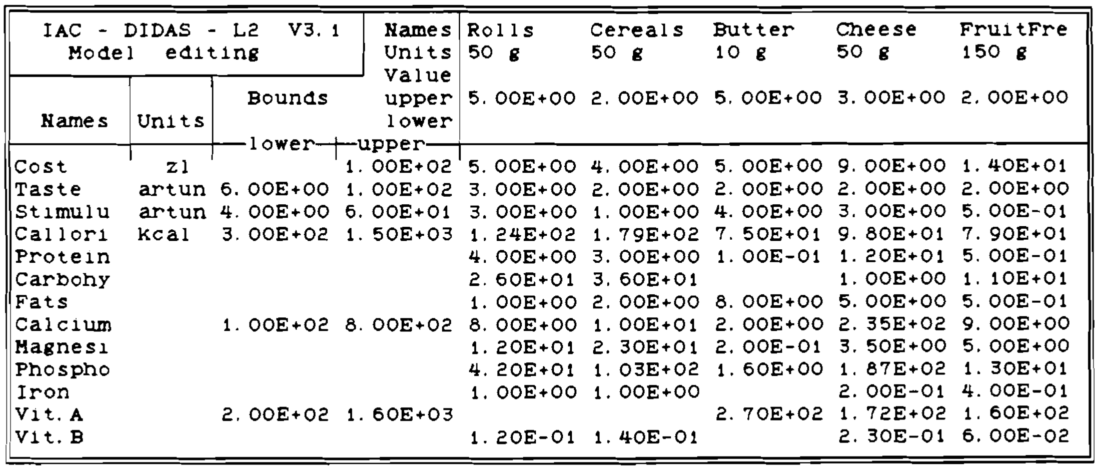
Wierzbicki, A.P. (1986): On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spektrum 8, 73-87.

APPENDIX

A shortened spreadsheet format of the tutorial model of multiobjective diet selection.

Dish Unit	Lo/Up	Rolls 50 g	Cereals 50 g	Butter 10 g	Cheese 50 g	Fruitfre 150 g	Milk 250 g	Coffee 1 cup
Lo.bound		0.0	0.0	0.0	0.0	0.0	0.0	0.0
Up.bound		5	2	5	3	2	3	3
Cost	0/100	5	4	5	9	14	6	18
Taste	6/100	2	2	2	2	2	1	2
Stimulus	4/60	3	2	4	3	0.5	5	10
Calorie	300/1500	124	179	75	98	79	137	0.0
Proteins		4	3	0.1	12	0.5	7	0.0
Carbohyd.		26	36	0.0	1	11	10	0.0
Fats		1	2	8	5	0.5	7	0.0
Calcium	100/800	8	10	2	235	9	295	0.0
Magnesium		12	23	0.2	3.5	5	30	0.0
Phosphor.		42	103	1.6	187	13	213	0.0
Iron		1	1	0.0	0.2	0.4	0.25	0.0
Vit.A	200/1600	0.0	0.0	270	172	160	277	0.0
Vit.B		0.12	0.14	0.0	0.23	0.06	0.73	0.0
Vit.C		0.0	0.0	0.0	0.0	30	2.5	0.0
Vit.PP		0.4	1.0	0.01	0.05	0.23	0.25	0.0

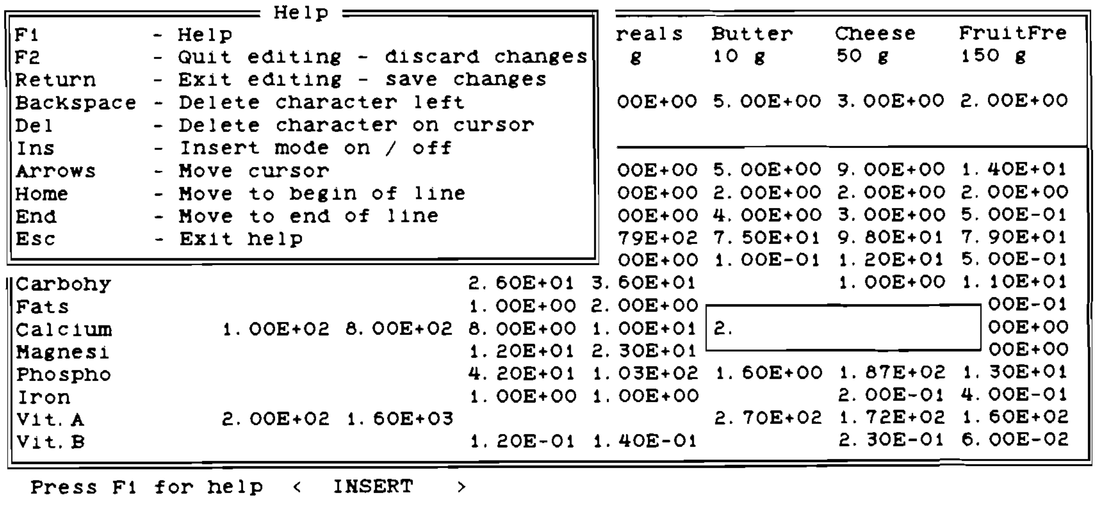
The following example is spreadsheet format of IAC-DIDAS-L2 (a screen print, other parts of the data accessible through scrolling).



```
 IAC - DIDAS - L2  V3.1        Names  Rolls      Cereals    Butter     Cheese     FruitFre
   Model  editing              Units  50 g       50 g       10 g       50 g       150 g
                               Value
                   Bounds      upper  5.00E+00   2.00E+00   5.00E+00   3.00E+00   2.00E+00
 Names    Units                lower
                  lower     upper
Cost        zl              1.00E+02  5.00E+00   4.00E+00   5.00E+00   9.00E+00   1.40E+01
Taste     artun  6.00E+00   1.00E+02  3.00E+00   2.00E+00   2.00E+00   2.00E+00   2.00E+00
Stimulu   artun  4.00E+00   6.00E+01  3.00E+00   1.00E+00   4.00E+00   3.00E+00   5.00E-01
Callori   kcal   3.00E+02   1.50E+03  1.24E+02   1.79E+02   7.50E+01   9.80E+01   7.90E+01
Protein                               4.00E+00   3.00E+00   1.00E-01   1.20E+01   5.00E-01
Carbohy                               2.60E+01   3.60E+01              1.00E+00   1.10E+01
Fats                                  1.00E+00   2.00E+00   8.00E+00   5.00E+00   5.00E-01
Calcium          1.00E+02   8.00E+02  8.00E+00   1.00E+01   2.00E+00   2.35E+02   9.00E+00
Magnesi                               1.20E+01   2.30E+01   2.00E-01   3.50E+00   5.00E+00
Phospho                               4.20E+01   1.03E+02   1.60E+00   1.87E+02   1.30E+01
Iron                                  1.00E+00   1.00E+00              2.00E-01   4.00E-01
Vit.A            2.00E+02   1.60E+03                        2.70E+02   1.72E+02   1.60E+02
Vit.B                                 1.20E-01   1.40E-01              2.30E-01   6.00E-02
```

Press F1 for help

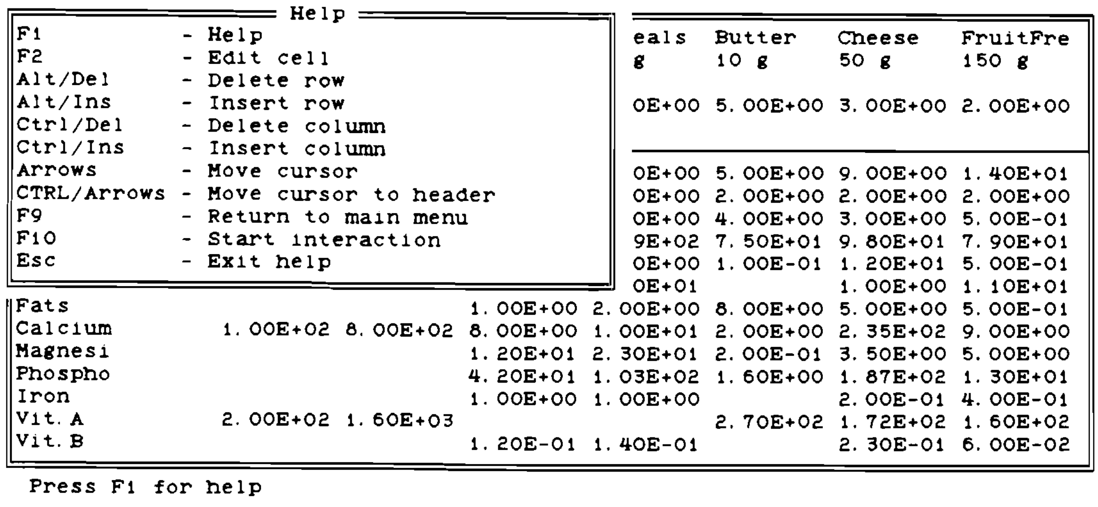
Editing help during model editing in IAC-DIDAS-L2



```
                  Help
F1          - Help                              reals   Butter    Cheese    FruitFre
F2          - Quit editing - discard changes    g       10 g      50 g      150 g
Return      - Exit editing - save changes
Backspace   - Delete character left             00E+00  5.00E+00  3.00E+00  2.00E+00
Del         - Delete character on cursor
Ins         - Insert mode on / off
Arrows      - Move cursor                       00E+00  5.00E+00  9.00E+00  1.40E+01
Home        - Move to begin of line             00E+00  2.00E+00  2.00E+00  2.00E+00
End         - Move to end of line               00E+00  4.00E+00  3.00E+00  5.00E-01
Esc         - Exit help                         79E+02  7.50E+01  9.80E+01  7.90E+01
                                                00E+00  1.00E-01  1.20E+01  5.00E-01
Carbohy                       2.60E+01 3.60E+01           1.00E+00  1.10E+01
Fats                          1.00E+00 2.00E+00                       00E-01
Calcium     1.00E+02 8.00E+02 8.00E+00 1.00E+01 2.                    00E+00
Magnesi                       1.20E+01 2.30E+01                       00E+00
Phospho                       4.20E+01 1.03E+02 1.60E+00  1.87E+02  1.30E+01
Iron                          1.00E+00 1.00E+00           2.00E-01  4.00E-01
Vit.A       2.00E+02 1.60E+03                   2.70E+02  1.72E+02  1.60E+02
Vit.B                         1.20E-01 1.40E-01           2.30E-01  6.00E-02

Press F1 for help  <  INSERT   >
```

Further editing help in IAC-DIDAS-L2



```
                  Help
F1            - Help                                eals    Butter    Cheese    FruitFre
F2            - Edit cell                           g       10 g      50 g      150 g
Alt/Del       - Delete row
Alt/Ins       - Insert row                          0E+00   5.00E+00  3.00E+00  2.00E+00
Ctrl/Del      - Delete column
Ctrl/Ins      - Insert column
Arrows        - Move cursor                         0E+00   5.00E+00  9.00E+00  1.40E+01
CTRL/Arrows   - Move cursor to header               0E+00   2.00E+00  2.00E+00  2.00E+00
F9            - Return to main menu                 0E+00   4.00E+00  3.00E+00  5.00E-01
F10           - Start interaction                   9E+02   7.50E+01  9.80E+01  7.90E+01
Esc           - Exit help                           0E+00   1.00E-01  1.20E+01  5.00E-01
                                                    0E+01             1.00E+00  1.10E+01
Fats                          1.00E+00 2.00E+00 8.00E+00  5.00E+00  5.00E-01
Calcium     1.00E+02 8.00E+02 8.00E+00 1.00E+01 2.00E+00  2.35E+02  9.00E+00
Magnesi                       1.20E+01 2.30E+01 2.00E-01  3.50E+00  5.00E+00
Phospho                       4.20E+01 1.03E+02 1.60E+00  1.87E+02  1.30E+01
Iron                          1.00E+00 1.00E+00           2.00E-01  4.00E-01
Vit.A       2.00E+02 1.60E+03                   2.70E+02  1.72E+02  1.60E+02
Vit.B                         1.20E-01 1.40E-01           2.30E-01  6.00E-02

Press F1 for help
```

A Solver for the Transshipment Problem with Facility Location

Wlodzimierz Ogryczak, Krzysztof Studzinski, Krystian Zorychta
Institute of Informatics, Warsaw University.

ABSTRACT

This paper describes the initial results of research, development and implementation of the Dynamic Interactive Network Analysis System (DINAS) which will make opportunity for solving various multiobjective transshipment problems with facility location on IBM PC/XT microcomputers. The main result of this stage is the development and implementation of the TRANSLOC solver which provides the DINAS with solutions to single-objective problems. It is based on the branch and bound scheme with a pioneering implementation of the simplex special ordered network (SON) algorithm with implicit representation of the VUB & SUB constraints. The paper describes in details backgrounds of techniques used in the TRANSLOC solver. A real example of the transshipment problem with facility location is also discussed and an outline of the designed procedure for handling multiple objectives in the DINAS is given.

1. Introduction.

The distribution - location type problems belong to the class of most significant problems directly leading to real life applications of mathematical programming methods. Steadily rising costs and inflation as well as legal and political considerations, competition, fuel scarcity and many other factors have led, in recent years, many organizations to examine more closely their present and planned distribution patterns or facility locations. For instance, the impact of the energy crisis in the 70-th caused real impetus for re-evaluation of existing and often outmoded distribution patterns and methods.

Suppose we have a number of facilities and a number of customers or customer zones. Finding the distribution pattern is a fairly straightforward mathematical programming problem, e.g. transportation problem. When we add the possibility of removing or adding a number of facilities with their associated fixed costs, we have a more complex facility location problem which is in general an integer programming problem. Many real world problems in industry, business, government and nonprofit organizations include a variety of conflicting goals and objectives as functions of their distribution patterns and facility locations. Adding these functions as the criteria of optimization we expand our problem into a multicriteria transportation and facility location problem. However, real life situations create even more complex problems. Therefore the problem considered in the paper will be precisely described and formulated once more in the next sections.

Due to the multiple objective formulation and to the integrity of location variables, the problem is complicated and computationally complex. Hence the method designed for solving the problem should be stable and fast in order to produce a correct result or its acceptable approximation in a reasonable time.

This paper describes the initial results of research, development and implementation of the Dynamic Interactive Network Analysis System (DINAS) which is being developed with the purpose of solving various multiobjective transshipment problems with facility location on IBM PC/XT microcomputers. The main result of this stage is the development and implementation of the TRANSLOC solver which provides the DINAS with solutions to single-objective problems. It is based on the branch and bound scheme with a pioneering implementation of the simplex special ordered network (SON) algorithm with implicit representation of the VUB & SUB constraints. The paper describes in details backgrounds of techniques used in the TRANSLOC solver. A real example of the transshipment problem with facility location is also discussed and an outline of the designed procedure for handling multiple objectives in the DINAS is given.

2. An example

As an illustration of the transshipment-location type problem mentioned in the previous section, the problem of location of depots in a sugar-beet distribution system is considered. The problem was studied by Jasinska & Wojtych in [7]. They were dealing with a real-life problem concerning a sugar enterprise in Lower Silesia, in Poland.

There are 1588 villages in the considered region. Each of them is treated as a farm that produces the sugar-beet. Every farm is characterized by its total supply in the sugar-beet harvesting period. The sugar-beet is supplied to sugar-mills directly or through some depots. There are 12 sugar-mills in the region. Each sugar-mill is characterized by two amounts: the total storing capacity and the total production capacity in one production season.

A sugar production season in Poland lasts about three months. The total amounts of the sugar-beet must be shipped between the farms and the sugar-mills in this period. There are three types of shipping: between the farms and depots, between the depots and sugar mills, and directly between the farms and the sugar-mills. Each of the types is characterized by a unit cost of the shipping.

Climatic conditions and poor storage facilities may cause losses of sugar-beet volume or sugar content in the sugar beet. To avoid the losses, the deliveries from farms should be carried out within the harvesting season (the beginning phase of the sugar production season). However, the sugar-mills stores have limited capacities and cannot take all the amount of the sugar-beet in the short time. Therefore, a part of the sugar-beet supply has to be delivered to depots and stored there temporarily. But the technological and economic analysis indicates that the density of the existing network of small depots is insufficient in the case of an increased supply. Hence, some existing depots should be modernized to increase their throughputs and some new depots should be built.

The sugar industry decision maker chose 49 possible depot locations in the considered region. Each location is characterized by the lower bound (20 000 tons) and the upper bound (55 000 tons) of throughput. Every potential depot is considered as two separate depots: the basic one with the throughput within the interval [20 000, 34 000] and the additional one with the throughput belonging to [0, 20 000]. The additional depot can be opened at the same site if the basic depot reaching its upper throughput limit is opened there.

Thus in the given site:

- no depot need to be located;
- the basic depot may be located;
- both the basic depot and the additional one may be located provided the basic depot reaches the upper bound of its throughput.

Each depot location is evaluated by the operating and the investment costs. The investment cost is defined as the annual fixed charge of the basic or additional depot.

The problem is to determine the number, location and sizes of the depots to be selected from the candidate set and to find the corresponding sugar-beet flows from farms to sugar-mills directly or through depots so as to minimize the total transportation and depot investment and operating cost (provided the total amount of sugar-beet is delivered from farms to sugar-mills).

As reported in the quoted paper [7], the problem could not be solved in a reasonable time due to its large size. Fortunately, the size can be reduced by an aggregation of farms into supply zones. The farms located in the neighborhood of the same depot or sugar-mill or situated along the same route were aggregated. In consequence, instead of 1588 farms 128 zones were generated and a reduced problem was solved using the MPSX and MIP systems.

The problem described above represents a class of transshipment problem with facility location. It is a single-objective optimization problem. However, the single-objective optimization is insufficient in real-life circumstances and additional objectives should be taken into consideration. For instance, the total amount of the sugar-beet flow through depots should be minimized. This criteria seems to be very important because of the direct flows from farms to sugar-mills are technologically most efficient. As another objective, minimization of the total amount of the sugar-beet delivered by rail or maximization of the sugar production volume can be considered. The objectives need not be, in general, comparable; therefore our problem should be considered as a multicriteria optimization problem. The multicriteria optimization approach to the transshipment-location type problem will be developed more precisely in next sections.

3. The generalized network model

In the previous section, we have introduced a class of transshipment problems with facility location. In this section, we define the mathematical model of such problems more precisely.

A network model of the problem consist of nodes that are connected by a set of direct flow arcs. The set of nodes is partitioned into two subsets: the set of fixed nodes and the set of potential nodes. The fixed nodes represent "fixed points" of the transportation network, i.e., points which cannot be changed. Each fixed node is characterized by two quantities: supply and demand. The potential nodes are introduced to represent possible locations of new points in the network. Some groups of the potential nodes represent different versions of the same facility to be located (e.g., different sizes of a warehouse). For this reason, potential nodes are organized in the so-called selections, i.e., sets of nodes with the multiple choice requirement. Each selection is defined by the list of included potential nodes as well as by a lower and upper number of nodes which have to be selected (located). Each potential node is characterized by a capacity which bounds maximal flow through the node. The capacities are also given for all arcs but not for the fixed nodes.

Several linear objective functions are considered in the problem. The objective functions are introduced into the model by given coefficients associated with several arcs and potential nodes. They will be called cost coefficients independently of their real character in the objective functions. The cost coefficients for potential nodes are, however, understood in different way than for arcs. The cost coefficient connected to an arc is treated as the unit cost of the flow along the arc whereas the cost coefficient connected to a potential node is considered as the fixed cost associated with using (locating) of the node rather than as the unit cost.

We assume two restrictions on the network structure in our model:

(1) there are no arcs that directly connect two potential nodes;

(2) each potential node belongs to one or two selections.

Both the restriction are not very strong. The first one does not imply any loss of generality since every two of potential nodes can be separated by an introduction of an artificial fixed node if necessary. The second requirement, in general, restricts the class of problems. However, each potential node in practical models usually belongs to exactly one selection or sometimes to two selections in more complex problems.

For simplification of the model and the solution procedure, we transform the potential nodes into artificial arcs. The transformation is performed by duplication of all potential nodes. After the duplication all the nodes can be considered as fixed and each potential node is replaced by an artificial arc which leads from the node to its copy. Due to the transformation we get a network with fixed structure since all the nodes are fixed. Potentiality of artificial arcs does not imply any complication because each arc in the network represents a potential flow. Moreover, all the bounds on flows (i.e., capacities) are connected to arcs after this transformation. Additional nonstandard discrete constraints on the flow are generated only by the multiple choice requirements associated with the selections. Cost coefficients are connected only to arcs, but the coefficients connected to artificial arcs represent fixed costs.

A mathematical statement of this transformed problem takes the form of the following generalized network model:

$$minimize \quad \sum_{(i,j)\in A\setminus A_a} f^p_{ij}x_{ij} + \sum_{(i,j)\in A_a} f^p_{ij}y_{ij} \qquad p=1,2,\ldots,nobj \tag{3.1}$$

subject to

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i \qquad i=1,2,\ldots,nnode \tag{3.2}$$

$$0 \le x_{ij} \le c_{ij}, \quad (i,j)\in A \tag{3.3}$$

$$0 \le x_{ij} \le c_{ij}y_{ij}, \quad (i,j)\in A_a \tag{3.4}$$

$$g_k \le \sum_{(i,j)\in S_k} y_{ij} \le h_k, \quad k=1,2,\ldots,nsel \tag{3.5}$$

$$y_{ij}=0 \; or \; 1, \quad (i,j)\in A_a \tag{3.6}$$

where

nobj	number of objective functions,
nnode	number of nodes (including copies of potential nodes),
nsel	number of selections,

A	set of arcs (including artificial arcs),
A_a	set of artificial arcs,
f^p_{ij}	cost coefficient of the p-th objective associated with the arc (i,j),
b_i	supply-demand balance at the node i (supply is denoted as a positive quantity and demand as negative),
c_{ij}	capacity of the arc (i,j)
g_k, h_k	lower and upper number of (artificial) arcs to be selected in the k-th selection,
S_k	set of (artificial) arcs that belong to the k-th selection,
x_{ij}	decision variable that represents flow along the arc (i,j),
y_{ij}	decision variable equal 1 for selected arc and 0 otherwise.

The generalized network model of this form includes typical network constraints (3.2) with simple upper bounds (3.3) as well as a special discrete structure (3.5)-(3.6) connected to the network structure by variable upper bounds (3.4). While solving the model we have to take advantages of all these structures.

Taking into consideration an artificial arc, we notice that its capacity limits not only the flow along this arc but also many other flows. Let (i_o, j_o) be an artificial arc. Then (i_o, j_o) is the only arc which emanates from the node i_o and only arc which reaches the node j_o. Due to this fact we can introduce additional bounds on flow along each arc which reaches the node i_o or emanates from the node j_o. In such a way we get additional inequalities:

$$x_{ti} \leq c_{ij} y_{ij} \text{ and } x_{jt} \leq c_{ij} y_{ij} \quad (i,j) \in A_a, (t,i) \in A, (j,t) \in A \tag{3.7}$$

which makes the constraints of our model tighter and improves effectiveness of the solution process.

4. Interactive procedure for handling multiple objectives

There are many different concepts for handling multiple objectives in mathematical programming. We decided to use the so-called reference point approach. The reference point approach introduced by Wierzbicki (see [16]) was developed in many papers (see [9]) and was used as a basis for construction of the software package DIDAS (Dynamic Interactive Decision Analysis and Support system). The DIDAS package developed at IIASA proved to be useful in analyzing conflicts and assisting in decision making situations (see [4],[5]).

The reference point approach is a generalization of the well-known goal programming method (see [6]) and of the method of displaced ideals (see [18]). The basic concept of this approach is as follows:

(1) the decision-maker (DM) forms his requirements in terms of aspiration levels, i.e., he specifies acceptable values for given objectives;

(2) the DM works with the computer in an interactive way so that he can change his aspiration levels during the sessions of the analysis.

In our system, we extend the DIDAS approach. The extension depends on additional use of reservation levels which allow the DM to specify necessary value for given objectives (see [17]).

Consider the multi-objective program associated with the generalized network model:

$$\textit{minimize} \quad q$$

subject to

$$q = F(x,y)$$

$$(x,y) \in Q$$

where

q represents the vector,

F is the linear vector-function defined by (3.1),

Q denotes the feasible set of the generalized network model, i.e., the set defined by conditions (3.2)-(3.7).

The reference point technique works in two stages. In the first stage the DM is provided with some initial information which gives him an overview of the problem. The initial information is generated by minimization of all the objectives separately. More precisely, a sequence of single objective programs is solved defined as follows:

$$\min\{f^p(x,y) + \frac{\rho}{nobj}\sum_{i=1}^{nobj} f^i(x,y) \,:\, (x,y)\in Q\}, \quad p=1,2,...,nobj \tag{4.1}$$

where f^p denotes the p-th objective function and ρ is an arbitrarily small number.

The so-called decision-support matrix (or pay-off matrix) $D=(q_{pj})$ $p=1,...,nobj$; $j=1,...,nobj$ which yields information on the range of numerical values of each objective is then constructed. The p-th row of the matrix D corresponds to the vector (x^p,y^p) which solves the p-th program (4.1). Each quantity q_{pj} represents a value of the j-th objective at this solution (i.e., $q_{pj}=f^j(x_p,y_p)$). The vector with elements q_{pp}, i.e., the diagonal of D, defines the utopia (ideal) point. This point, denoted further by q^u, is usually not attainable but it is presented to the DM as a lower limit to the numerical values of the objectives.

When analysing a column j of the matrix D, we notice that the minimal value in the column is $q_{pp}=q^{n_p}$. Let q^{n_j} be the maximal value, i.e.,

$$q^{n_j} = \max_{1\leq p<n=obj} q_{pj}$$

The point q^n is called the nadir point and may be presented to the DM as an upper guideline to the values of the objectives. Thus, for each objective f^p a reasonable but not necessarily tight upper bound q^n and a lower bound q^u are known after the first stage of the analysis.

In the second stage, an interactive selection of efficient solutions is performed. The DM controls the selection by two (vector-) parameters: his aspiration level q^a and his reservation level q^r, where

$$q^u \leq q^a \leq q^r \leq q^n$$

The support system searches for the satisfying solution while using an achievement scalarizing function as a criterion in single-objective optimization. Namely, the support system computes the optimal solution to the following problem:

$$\textit{minimize} \quad \max_{1\leq p\leq nobj} \mu_p(q,q^a,q^r) + \frac{\rho}{nobj}\sum_{p=1}^{nobj} \mu_p(q,q^a,q^r) \tag{4.2}$$

subject to

$$q=F(x,y)$$
$$(x,y)\in Q$$

where ρ is an arbitrarily small number and μ_p is a function which measures the deviation of results from the DM's expectations with respect to the p-th objective, depending on given aspiration level q^a and reservation level q^r.

The computed solution is an efficient (Pareto-optimal) solution to the original multiobjective model. It is presented to the DM as a current solution. The DM is asked whether he finds this solution satisfactory or not. If the DM does not accept the current solution he has to enter new aspiration and/or reservation levels for some objectives. Depending on this new information supplied by the DM, a new efficient solution is computed and presented as a current solution. The process is repeated as long as necessary.

The function $\mu_p(q,q^a,q^r)$ is a strictly monotone function of the objective vector q with value $\mu_p=0$ if $q=q^a$ and $\mu_p=1$ if $q=q^r$. In our system, we intend to use a piecewise linear function μ_p defined as follows:

$$\mu_p(q,q^a,q^r)=\begin{cases}\beta_p(q_p-q_p^a)/(q_p^r-q_p^a), & \text{if} q_p<q_p^a\\ (q_p-q_p^a)/(q_p^r-q_p^a), & \text{if} q_p^a\leq q_p\leq q_p^r\\ \gamma_p(q_p-q_p^r)/(q_p^r-q_p^a)+1, & \text{if} q_p^r<q_p\end{cases}$$

where β_p and γ_p $(p=1,2,...,nobj)$ are given positive parameters. In particular, the parameters β_p and γ_p may be defined (similarly as in [17]) according to the formulae

$$\beta_p=(q_p^r-q_p^a)/(q_p^a-q_p^u)*\beta$$
$$\gamma_p=(q_p^r-q_p^a)/(q_p^n-q_p^r)*\gamma$$

with two arbitrarily given positive parameters β and γ.

If the parameters β_p and γ_p satisfy inequalities $\beta_p<1$ and $\gamma_p>1$, then the achievement functions μ_p are convex. Minimization of the function μ_p is then equivalent to minimization of a variable μ_p defined as follows:

$$\mu_p=v_p+\gamma_p v_p^- - \beta_p v_p^+ \tag{4.3}$$
$$v_p-v_p^+ + v_p^- =(q_p-q_p^a)/(q_p^r-q_p^a) \tag{4.4}$$
$$0\leq v_p\leq 1 \tag{4.5}$$
$$v_p^+\geq 0, \quad v_p^-\geq 0 \tag{4.6}$$

To provide for a special treatment of the equalities (4.3) in the single objective solver, we perform substitutions:

$$\beta_p v_p^+=d_p^+ \quad \textit{and} \quad \gamma_p v_p^- =d_p^-$$

Finally, we form the problem (4.2) in terms of linear programming as the following program:

$$\min \ z+\frac{\rho}{nobj}\sum_{p=1}^{nobj}\mu_p \tag{4.7}$$

subject to

$$\mu_p < z, \quad p=1,2,...,nobj \tag{4.8}$$

$$\mu_p = v_p + d_p^- - d_p^+, \quad p=1,2,...,nobj \tag{4.9}$$

$$v_p - \frac{1}{\beta_p} d_p^+ + \frac{1}{\gamma_p} d_p^- = (q_p - q_p^a)/(q_p^r - q_p^a), \quad p=1,2,...,nobj$$

$$0 \le v_p \le 1, \quad p=1,2,...,nobj \tag{4.11}$$

$$d_p^+ > 0, \quad d_p^- > 0, \quad p=1,2,...,nobj \tag{4.12}$$

$$q = F(x,y) \tag{4.13}$$

$$(x,y) \in Q \tag{4.14}$$

5. General concept of the TRANSLOC solver

The TRANSLOC solver has been prepared to provide the multiobjective analysis procedure with solutions to single-objective problems. According to the interactive procedure described in Section 4 the TRANSLOC solver has to be able to solve two kinds of single-objective problems: the first one associated with calculation of the decision support matrix (problems (4.1)) and the second one associated with minimization of the scalarizing achievement function (problems (4.2)). Both kinds of the problems have, however, the same main constraints which represent the feasible set of the generalized network model. Moreover, the other constraints of both kinds of problems can be expressed in very similar ways. So, we can formulate a general single-objective problem for the TRANSLOC solver as follows:

$$\max \ s \tag{5.1}$$

subject to

$$\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i \qquad i=1,2,...,nnode \tag{5.2}$$

$$w_k + \sum_{(i,j)\in S_k} y_{ij} = h_k \qquad k=1,2,...,nsel \tag{5.3}$$

$$\mu_p - v_p + d_p^+ - d_p^- = 0 \qquad p=1,2,...,nobj \tag{5.4}$$

$$v_p - \frac{1}{\beta_p} d_p^+ + \frac{1}{\gamma_p} d_p^- - \sigma_p\Big(\sum_{(i,j)\in A\setminus A_a} f_{ij}^p x_{ij} + \sum_{(i,j)\in A_a} f_{ij}^p y_{ij}\Big) = \tag{5.5}$$

$$= \delta_p \quad p=1,2,...,nobj$$

$$\mu_o - \sum_{p=1}^{nobj} \mu_p = 0 \tag{5.6}$$

$$s + z + \frac{\rho}{nobj}\mu_o = 0 \tag{5.7}$$

$$0 \le x_{ij} \le c_{ij} \qquad (i,j)\in A \tag{5.8}$$

$$0 \le w_k \le h_k - g_k \qquad k=1,2,...,nsel \tag{5.9}$$

$$x_{ij} \le c_{ij} y_{ij},\ x_{ti} \le c_{ij} y_{ij},\ x_{jt} \le c_{ij} y_{ij} \quad (i,j)\in A_a,\ (t,i)\in A, (j,t)\in A \tag{5.10}$$

$$\mu_p \le z \qquad p=1,2,...,nobj \tag{5.11}$$

$$y_{ij} = 0 \ or \ 1 \qquad (i,j)\in A_a \tag{5.12}$$

and depending on the kind of optimization:

$$d_p^+ = 0 \ , \ d_p^- = 0 \qquad\qquad p=1,2,...,nobj \tag{5.13}$$

for the utopia point calculation or

$$d_p^+ \geq 0 \ , \quad d_p^- \geq 0, \quad 0 \leq v_p \leq 1 \quad p=1,2,...,nobj \tag{5.14}$$

for the achievement scalarizing function optimization, respectively, where: $\sigma_p = 1$ and $\delta_p = 0$ during utopia point calculation, $\sigma_p = \dfrac{1}{q_p^r - q_p^a}$ and $\delta_p = \dfrac{-q_p^a}{q_p^r - q_p^a}$ during the minimization of the achievement scalarizing function, whereas all the other quantities are the same as in Sections 3 and 4.

The above single-objective problem is a typical mixed integer linear program, i.e., it is a typical linear program with integrity conditions for some variables (namely y_{ij}). Mixed integer linear programs are usually solved by branch and bound approach with utilization of the simplex method. The TRANSLOC solver also uses this approach. Fortunately, only very small group of decision variables is required to be integer in our model. Therefore, we can use a simple branch and bound scheme in the solver. Background of this scheme is described in Section 6.

Even for a small transshipment problem with facility location the corresponding linear program (5.1) - (5.11) has rather large size. For this reason it, cannot be solved directly with the standard simplex algorithm. In order to solve the program on IBM PC/XT microcomputers, it is necessary to take advantages of its special structure.

Note that the main group of equality constraints (5.2) represents typical network relations. Similarly, the equalities (5.3) and (5.4) include only variables with unit coefficients. All the rows (5.2) - (5.4) can be handled in the simplex method as the so-called special ordered network (SON) structure. Basic rules of the SON technique used in the TRANSLOC solver are developed in Section 7.

The inequalities (5.8) - (5.9) and (5.13) or (5.14) are standard simple upper bounds (SUB) which are usually processed out of the linear programming matrix. Similarly, inequalities (5.10) and (5.11) can be considered as the so-called variable upper bounds (VUB) and processed out of the matrix due to a special technique. Basic rules of the technique for SUB & VUB processing are developed in Section 8.

Thus, only a small number of inequalities (5.5) - (5.7) has to be considered as typical rows of linear program. While taking advantage of this fact, the TRANSLOC solver can process transshipment problems of quite large dimensions. As a proper size of problems for IBM PC/XT microcomputers we regard:

- a few objective functions,
- about one hundred of fixed nodes,
- a few hundreds of arcs,
- several potential nodes (artificial arcs) organized in a few selections.

Initial experiences with the TRANSLOC solver show that such problems can be solved on IBM PC/XT microcomputers in reasonable time.

6. The branch and bound scheme

6.1. A basic concept

As mentioned in Section 5, the TRANSLOC solver uses the branch and bound approach for handling the discrete structure of the single-objective programs (5.1) – (5.14). The branch and bound technique was introduced by Land and Doig (see [8]) and further developed by many authors (see e.g. [1]). It is implemented in most of the commercial codes for solving mixed integer linear programs. The branch and bound approach proved to be the most effective for linear programs with large number of continuous variables and relatively small number of integer variables. The single-objective programs (5.1) - (5.14) have just such a structure.

The problem (5.1) - (5.14) can be shortly written in the form:

$$P_o\text{: max } f(x,y)\text{: } (x,y)\in G,\ 0 \leq y_{ij} \leq 1,\ y_{ij}\in Z \text{ for } (i,j)\in A_a$$

where:

x represents a vector of all the continuous variables,

G is the feasible set defined by conditions (5.2) - (5.11) and (5.13) or (5.14), respectively,

Z denotes the set of integer numbers.

A basic concept of the branch and bound approach to solving the problem P_o may be summarized as follows (see [10]). At first, the continuous variable problem associated with P_o is solved. If one or more variables y_{ij} turn out to be noninteger, i.e., $0 < y_{ij} < 1$ in the optimal solution, then one such variable is chosen as the so-called branching variable to generate two subproblems P_L and P_R :

P_L : problem P_o with the added constraint $y_{ij} = 0$;
P_R : problem P_o with the added constraint $y_{ij} = 1$.

The same procedure can be, obviously, repeated with respect to new problems P_L or/and P_R. In such a way, we construct a general process which generates a binary tree of (sub)problems.

Consider a parent problem P_N from which two subproblems of types L and R are generated. Let C_k denote the optimum value of the objective function of the continuous problem corresponding to the problem P_k, and let I_k denote the value of the objective function at an optimal integer solution to P_k. The following relations are easily seen to be true:

$$C_k \geq I_k \qquad k{=}N,L,R$$

$$I_N = \max I_L, I_R$$

$$C_N \geq C_L,\ \ C_N \geq C_R$$

Let T denote the so-called candidate set of problems, that is the set of indices identifying all the terminal nodes at any stage of the development of the tree. Then

$$\max_{i\in T} C_i \geq I_o \geq I_k$$

where I_k is the value of the objective function of any $P_k, k\in T$, such that $C_k{=}I_k$ and I_o corresponds to the optimal integer solution sought. Thus by recurring the process of branching and solving continuous problems the optimal integer solution will be finally found as an optimal solution to a continuous problem corresponding to one of the subproblems from the candidate set.

6.2. The branch and bound algorithm

At each stage of the branch and bound process it is necessary to know whether or not an integer (feasible) solution to P_0 has been found so far. If it has, it is also necessary to know the value of the objective function of the best of these. Existence is recorded in the algorithm by an integer solution marker and the best solution value so far found by the so-called cutting-value V. A basic branch and bound algorithm may be written as follows:

(0) *Initial step.* Define the integer solution marker and the cutting-value according to the best known integer solution or clear the marker and initialize the cutting-value to a large negative value. Solve the continuous LP problem corresponding to P_0. If an optimal solution exists put P_0 into the candidate set and go to (1). Otherwise go to (4).

(1) *Subproblem selection.* If the candidate set is empty, go to (4). Otherwise choose a subproblem from the candidate set (deleting it from the set) and test whether the objective value of the continuous optimal solution is greater than the cutting-value V. If it is not, enter (1) again.

(2) *Branching.* Select a branching variable y_{rs} such that $0<y_{rs}<1$ for optimal solution to the continuous subproblem and generate two subproblems P_L and P_R.

(3) *Subproblems solution.* Solve the continuous problems corresponding to the P_L and P_R subproblems using a simplex algorithm. If some subproblem has no feasible solution then it is regarded as fathomed. If the objective function value is less than or equal to V, then the corresponding subproblem is regarded as fathomed. If the solution satisfies the integrity conditions (i.e., $y_{ij}=0$ *or* 1 for $(i,j)\in A_a$), then the corresponding subproblem is regarded as fathomed, the integer solution marker is set and the cutting-value V is updated. Add the subproblems which are not fathomed to the candidate set and go to (1).

(4) *Exit.* If the integer solution marker is not set, then no integer (feasible) solution exists to the original problem. Otherwise output the best integer solution as the optimal solution.

Effectiveness of an implementation of the branch and bound algorithm depends on techniques used for putting through the following tasks: a) solving of continuous LP subproblems, b) subproblem selection, c) choice of branching variable.

Solving of continuous subproblems should be organized in such a way that the optimal basis to the parent subproblem is restored as an initial basis for optimization of the current subproblem. Usually the dual simplex method is used for performing the LP optimization (see [19]). It was impossible, however, to use the dual simplex method in our code due to the VUB structure processed out of the basis. For this reason we decided to use the so-called composite primal simplex algorithm (see [11]) which also is very effective in reoptimization of slightly modified linear programs.

There are many heuristic strategies for subproblem selection and choice of branching variable. They use some special statistical estimations (the so-called pseudo-costs) and are reported to be very effective while solving extremely large integer programs. However, our solver was designed for solving rather small problems on microcomputers. Therefore, we chose techniques which intend to restrict the growth of the tree. We use the so-called LIFO (Last In - First Out) strategy for subproblem selection, i.e., we always select the most recently created subproblem for branching. This strategy implies that the candidate set is added to and subtracted from in a linear fashion. As a branching variable we choose the closest one to the integer value, i.e., a variable which minimizes $\min(y_{ij}, 1-y_{ij})$.

Such a strategy used together with the LIFO rule for subproblem selection aims at minimizing the number of the so-called backtrackings in the branch and bound process. In result we get a simple branch and bound algorithm which appears to be rather effective in solving problems of type (5.1) - (5.14) on IBM PC/XT microcomputers.

7. The simplex SON algorithm

7.1. Graph representation

The simplex special ordered network (SON) procedure was developed by Glover & Klingman in [2],[3]. It is a partitioning method for solving LP problems with embedded network structure. The procedure is based on the network topology of the basis embodied in a specially constructed master basis tree. The main features of this approach are fast multiplication algorithms with basis inverse, accelerated labelling algorithms for modifying the master basis tree in an efficient manner and a compact form of the basis inverse occupying a small memory space only.

Let A denote the matrix of an auxiliary LP problem (excluding the SUB and VUB constraints). Without loss of generality it will be assumed that A has full row rank. As it was discussed in Section 5 the matrix A has a special form

$$\begin{array}{|c|c|}\hline \text{ANN} & \text{ANL} \\ \hline \text{ALN} & \text{ALL} \\ \hline \end{array}$$

where ANN (m $\times$ n) denotes the matrix corresponding to a pure network problem.

Thus each column of the submatrix ANN contains at most one +1, one -1 end zeros elsewhere. It will be assumed that there is no column with all zeros in the matrix ANN. The other submatrices ANL (m$\times$p), ALN (q$\times$n), ALL (q$\times$p) consist of any real elements.

The matrix ANN defines a digraph G(V,E) as follows. Each constraint represented by a row of ANN corresponds to a node of the graph and will be referred to as node constraint or simply node. Each variable represented by a column of ANN corresponds to an arc of the graph and will be referred to as arc variable or simply arc. There are two classes of arcs: ordinary arcs which have exactly two non-zero entries in ANN and slack arcs which have exactly one non-zero entry in ANN. The -1 entry in a column indicates the node where the arc begins and the +1 entry in a column indicates the node where the arc ends. For a slack arc (column with only one non-zero entry) the endpoints of the arc are incident at the same node. Such arc is called simple loop.

7.2. Basis structure

The SUB and VUB simplex algorithms use a basis B which is composed of a linearly independent set of column vectors selected from the matrix A. Any basis B will be a nonsingular matrix of order (m+q)×(m+q) and may be partitioned as follows

$$\begin{array}{cc} X_{B1} & X_{B2} \\ \hline B_{11} & B_{12} \\ B_{21} & B_{22} \end{array}$$

where B_{11} is a nonsingular submatrix of ANL and $x_B=(x_{B1},x_{B2})$ denotes the basic part of x. It appears to be better for the effectiveness of the algorithm if rank of B_{11} is as large as possible.

Thus the basic variables x_{B1} are exclusively arc variables.The basic variables x_{B2} may also contain arc variables. Analogous principle holds for rows of B. The rows of B_{11} are exclusively node rows. The matrix (B_{21},B_{22}) may also contain node rows.

The basis inverse B^{-1} may be written as follows

$$\begin{array}{|c|c|} \hline B_{11} + B_{11}^{-1}B_{12}V^{-1}B_{21}B_{11} & -B_{11}B_{12}V^{-1} \\ \hline -V^{-1}B_{21}B_{11}^{-1} & V^{-1} \\ \hline \end{array}$$

where $V=B_{22} - B_{21}B_{11}^{-1}B_{12}$.

7.3. Tree representation of B_{11}

Define the so called master basis tree (MBT) associated with a given basis. The set of nodes of the tree contains all nodes of our LP/embedded network problem plus another node called the master root. Thus MBT always contains m+1 nodes

$$N = 0,1,...,m$$

where 0 is the master root, and m arcs. The nodes of MBT that correspond to rows of B_{21} are called externalized roots (ER's).

All of the ordinary arcs in B_{11} belong to MBT. There may be two types of simple loops associated with B_{11}. If the simple loop in B is a slack arc of ANN then the simple loop is replaced by an arc between the master root and its unique node. If the simple loop in B_{11} is an ordinary arc in ANN, it is replaced by an arc between its nodes in ANN (one of the endpoints of the arc is ER node). Each ER is connected to the master root by an externalized arc (EA).

The arcs in the master basis tree have a natural orientation defined as follows: if the edge (u,v) belongs to MBT and node u is nearer the master root than v, then u is called the predecessor of v and v is called the immediate successor of u. Thus we will refer to a basis arc as conformable if its ANN direction agrees with its MBT orientation, and refer to the arc as nonconformable otherwise.

7.4. Representation of MBT

The master basis tree is represented by the following node functions.

(1) *PRED.* The values of the function are defined as follows PRED[i] = predecessor of node i in MBT. For convenience PRED[0] = -1.

(2) *THREAD.* The function defines a connecting link (thread) which passes through each node exactly once. If i is a node on the thread, then THREAD[i] is the next one. The alternation of the nodes on the thread is defined by using the preorder method of tree passage.

(3) *RETHREAD.* It is a pointer which points in the reverse order of the thread. That is, if THREAD[i] = j then RETHREAD[j] = i.

(4) *DEPTH.* The value DEPTH[i] specifies the number of arcs in the predecessor path of node i to the master root.

(5) *CARD.* Let T(i) denotes the subtree of MBT associated with node i. (The node i is the root of the subtree and each node j such that predecessor path from j to the master root contains i, belongs to the subtree.) CARD[i] indicates the number of nodes contained in T(i).

(6) *LAST.* The value LAST[i] specifies the node in the subtree T(i) that is the last node of this subtree in thread order.

(7) *CONF.* Each node i in MBT represents the predecessor arc of the node. If the arc is conformable then CONF[i] = +1, otherwise CONF[i] = -1.

7.5. Finding the representation of the entering vector

Let $P = (P_1,P_2)^T$ denote the column vector selected to enter the basis matrix (P_1 specifies the part of P associated with B_{11} and P_2 the part associated with B_{21}). Similarly $\alpha = (\alpha_{B1},\alpha_{B2})^T$ denotes the representation of P in terms of B.

We have $\alpha = B^{-1}P$ and hence using the partitioning formula for B^{-1} we obtain the following system of equations

$$\alpha_{B2} = V^{-1}(-B_{21}B_{11}^{-1}P_1 + P_2)$$
$$\alpha_{B1} = B_{11}^{-1}(P_1 - B_{12}\alpha_{B2})$$

The execution of the formulas may be decomposed into five steps

(1) $G_1 = B_{11}^{-1}P_1$

(2) $G_2 = P_2 - B_{21}G_1$

(3) $G_3 = V^{-1}G_2$

(4) $G_4 = P_1 - B_{12}G_3$

(5) $G_5 = B_{11}^{-1}G_4$

Suppose that the matrix $D = V^{-1}$ is attained in the explicit form (it is the right down corner part of the matrix B^{-1}). The first and fiveth steps consist in computing the multiplication $x = B_{11}^{-1}G$ where either $G = P_1$ or $G = G^4$. Consider a multiplication procedure which uses the master basis tree structure for B_{11}. In the procedures the system of linear equations $B_{11}x = G$ is solved instead of computing the multiplication $x = B_{11}^{-1}G$. The system $B_{11}x = G$ is transformed via the MBT structure to an equivalent system $\tilde{B}_{11}^{-1}\tilde{x} = \tilde{G}$ with an upper triangular matrix $\tilde{B}_{11}$. The unknowns and equations are ordered in the triangular system as the arcs and nodes on the thread line, respectively.

Let x[k] denote the value of the basis variable represented by the node k in MBT. The result $x = (x[1],...,x[m]) = B_{11}^{-1}G$ may be computed by the algorithm:

Algorithm A.

```
(0)  k := LAST[0];
     While k = 0 do:
     (1)  if q := PRED[k] is not ER then G[q]:=G[q]+G[k];
     (2)  x[k] := CONF[k]*G[k];
     (3)  k := RETHREAD[k];
     End of while loop.
```

The cost of the algorithm is proportional to the number of nodes in MBT. There are other versions of the algorithm for special cases when a column of G contains either two non-zero entries or one non-zero entry only.

7.6. Finding the dual vector

Let $c_B = (c_{B1}, c_{B2})$ denote the vector of basis cost coefficients. The dual vector $w = (w_1, w_2) = c_B B^{-1}$ is needed at the pricing step of the simplex method and may be computed as follows:

$$w_2 = (c_{B2} - c_{B1}B_{11}^{-1}B_{12})\, V^{-1}$$

$$w_1 = (c_{B1} - w_2 B_{21}^{-1}) B_{11}$$

The formulas may be executed by decomposition into following steps:

(1) $H_1 = c_{B1}B_{11}^{-1}$

(2) $H_2 = c_{B2} - H_1 B_{12}$

(3) $H_3 = H_2 V^{-1}$

(4) $H_4 = c_{B1} - H_3 B_{21}$

(5) $H_5 = H_4 B_{11}^{-1}$

The multiplication at the step 3 may be directly computed since the matrix $D = V^{-1}$ is assumed to be kept in the explicit form. The steps 1 and 5 use the matrix B_{11}^{-1} for the multiplication $w = HB_{11}^{-1}$, where either $H = c_{B1}$ or $H = H_4$. The multiplication can be executed using the master basis tree structure for B_{11}.

Computing the dual vector is equivalent to solving the system of equations $wB_{11} = H$. By using the MBT structure we can solve an equivalent lower triangular system $\tilde{w}\tilde{B}_{11} = \tilde{H}$. In the last system, unknowns and equations are ordered as the nodes and the arcs on the thread line, respectively.

Let w[k] denote the component of w corresponding to the node k, and H[k] denote the component of H that corresponds to the arc represented by node k. Computation of the vector w is performed by the following algorithm:

Algorithm B.

```
(0)  i := THREAD[0];
     While i = 0 do:
     (1)  q := PRERD[i];
     (2)  if q is not ER,then w[i] := w[q]+CONF[i]*H[i];
     (3)  i := THREAD[i];
     End of while loop.
```

The cost of the algorithm depends linearly on the number of nodes. Another version of the algorithm solves the special case when a row vector of H contains one non-zero entry only.

7.7. Exchange rules

Consider a single step of the simplex method. When the incoming and outgoing variables are chosen, then the whole basis representation has to be changed and adjusted to the new situation. Thus the problem arises how to change, in a single simplex iteration, the matrix D and the functions describing the master basis tree.

Let x_s and x_r denote the incoming and outgoing variables, respectively, and let x_t be the so called transfer variable that belongs to x_{B2} and replaces x_r in x_{B1}, if it is possible.

At each iteration, the variables can alter by the transitions:

- Incoming variable x_s : $x_N \rightarrow x_{B1}$ or x_{B2}
- Outgoing variable x_r : x_{B1} or $x_{B2} \rightarrow x_N$
- Transfer variable x_t : $x_{B2} \rightarrow x_{B1}$, or no change
- Transfer ER nodes : $x_{B1} \rightarrow x_{B2}$ (one ER more), or $x_{B2} \rightarrow x_{B1}$ (one ER less), or no change.

If an arc is added to the master basis tree, then a loop is closed. In order to have a tree in the next iteration also, the loop must be cut and exactly one arc from the loop must be deleted. It is the fundamental exchange rule for the master basis tree. Now we will discuss in details all cases that can be met in the exchange.

When x_{B1} is maximal relative to x_{B2}, exactly one of the seven types of basis exchange steps, listed below, will occur, and their updating prescriptions will maintain x_{B1} maximal.

Type 1. The outgoing variable x_r leaves the x_{B2} part of the basis and either x_s is not an arc or its lop in MBT does not contain an EA. The master basis tree is not changed. D is transformed by elimination (the pivot row x_r lies in the x_{B2} part of B^{-1}).

Type 2. The outgoing variable x_r leaves the x_{B2} part of x_B, x_s is an arc and the loop for the arc contains an EA. D is transformed by elimination and the pivot row lying in the x_{B2} part of B^{-1} is deleted from D. The column of D corresponding to the ER representing the mentioned EA is dropped also. The x_s arc is added to MBT and the EA is deleted from the x_r loop. The ER representing the deleted EA changes

into an ordinary node.
Remark: In all of the remaining types of basis exchange the outgoing variable leaves the x_{B1} part of the basis. Hence the pivot row is always generated outside of D.

Type 3. The incoming variable x_s is not an arc or else its loop contains neither an EA nor the arc x_r. There exists an arc x_t in x_{B2} whose loop contains x_r.
When the elimination is finished, the D part of the pivot row is added to D and x_t row of D is dropped. The x_t arc is added to MBT and the x_r arc is deleted.

Type 4. Variable x_s is not an arc or else its loop contains neither an EA nor the arc x_r. No arc x_t exists in x_{B2} whose loop contains x_r.
Let n_r be the node representing the x_r arc. Generate the D part of the column of B^{-1} that corresponds to LAST[n_r] and add it to D. Add the pivot row to D after execution of elimination. Add the arc (0,LAST[n_r]) to MBT as a new EA and change the ordinary node LAST[n_r] into an ER.

Type 5. Variable x_s is an arc and its loop contains the x_r arc. If the loop contains an EA also, there is no x_t in x_{B2} whose loop contains x_r.
Execute the elimination of D. Add x_s to MBT and delete (from its loop) x_r.

Type 6. The incoming variable x_s is an arc and its loop contains an EA. There is an arc x_t in x_{B2} whose loop contains the outgoing arc x_r.
Elimination of D is executed. The x_t row of D is dropped. The column of D associated with the node which represents the mentioned EA is also dropped. The master basis tree is twice transformed:

(1) x_s is added and the EA is deleted;

(2) x_t is added and x_r is deleted. The ER associated with the deleted EA changes into an ordinary node.

Type 7. Variable x_s is an arc and its loop contains an EA, but does not contain the arc x_r. There is no an arc x_t lying on a loop containing x_r.
Let n_r denotes the node representing the outgoing arc x_r. Add to D the D part of the column of B^{-1} associated with the node LAST[n_r] and delete the column associated with ER representing deleted EA. Execute the elimination of D. Add the arc x_s to MBT and remove an EA lying in the loop for x_s. The ER associated with the EA changes into an ordinary node. The arc (0,LAST[n_r]) is added also to MBT as a new EA and the x_r arc is deleted. The ordinary node LAST[n_r] is changed into an ER.

8. Implicit representation of VUB & SUB constraints

8.1. A basic concept

The single-objective program (5.1) - (5.14) includes many inequalities of special simple forms. They can be partitioned into two groups. The first one consists of the so-called simple upper bounds (SUB), i.e., inequalities of the form

$$0 \leq x_j \leq c_j \quad \text{for } \textit{some variables } x_j \text{ and } \textit{constants } c_j,$$

such as conditions (5.8), (5.9), (5.14) with respect to variables v_p, and continuous form of (5.12). The second one includes the so-called variable upper bounds (VUB), i.e., inequalities of the form

$$x_j \leq c_j x_k \quad \text{for } \textit{some variables } x_j,\ x_k \text{ and } \textit{constants } c_j,$$

such as conditions (5.10) and (5.11).

SUB constraints are usually implicitly represented in commercial simplex codes (see [19]). Schrage (see [12]) proposed some technique for implicit representation of VUB constraints. The technique was further developed and led to effective implementations (see [13],[14]).

The techniques presented in the literature deals, however, only with a simple form of VUB constraints. Namely, it is assumed that $c_j=1$ in all VUBs and there are no upper bounds on x_k variables. The restriction of consideration to only unit variable upper bounds usually does not imply any loss of generality since it can be attained by a proper scaling of the problem. Unfortunately, in our model such scaling techniques cannot be used without destroying of the special SON structure (see Section 7). Therefore, we were forced to extend the VUB techniques in such a way that nonunit variable upper bounds as well as some simple upper bounds on x_k variables were acceptable.

With respect to the VUB & SUB structure the linear program under consideration can be formulated as follows. The numerical data consist of an m×n matrix A of rank m, a column m-vector b, a row n-vector f and a column n-vector c. In addition, the index set $N=1,2,...,n$ is partitioned into $J\cup K$, where J represents the so-called sons, i.e., variables which appear on the left-hand-side of variable upper bounds, and K represents the so-called fathers, i.e., variables which appear on the right-hand-side of variable upper bounds. Any variable that is not involved in any variable upper bound is regarded as a childless father. The set J is further partitioned into the sets $J(k)$, $k\in K$, where $J(k)$ is the set (possible empty) of sons of the father $k\in K$. It is assumed that the son has only one father and that no father has a father. The father connected to a son x_j will be denoted by $k(j)$. The problem is then

$$max \;\; fx$$

subject to

$$Ax=b$$

$$x_j \leq c_j x_k \quad \text{for } all \; k\in K \text{ and } j\in J(k)$$

$$x_k \leq c_k \quad \text{for } all \; k\in K$$

$$x \geq 0$$

Let s_j be a slack variable for the variable upper bound $x_j \leq c_j x_k$, so that

$$x_j + s_j = c_j x_k, \quad x_j \geq 0, \quad s_j \geq 0.$$

Consider a basic solution to the problem. The basis consists of the $m+v$ columns corresponding to some sons x_j, some fathers x_k and some slacks s_j (where v denotes the number of VUBs). From each VUB either one slack s_j or one son x_j belongs to the basis. Calculation of the basic slacks is out of our interest and they can be simply dropped from the basis, i.e., the corresponding rows and columns can be dropped. Further, the basic sons which arrive in the other VUBs can be eliminated by submission $x_j=c_j x_k$. So, the whole basic solution can be computed from an m×m basis consisting of some linear combinations of columns from matrix A.

A basic solution to the problem is characterized as follows. The set of sons is partitioned into the three sets $J=JL\cup JU\cup JB$, where JL denotes the set of nonbasic sons fixed at their lower limits (i.e., $x_j=0$), JU denotes the set of nonbasic sons fixed at their upper limits (i.e., $x_j=c_j x_k$) and JB denotes the set of basic sons. Similarly, the set of fathers is partitioned into three sets $K=KL\cup KU\cup KB$, where KL denotes the set of nonbasic fathers fixed at their lower limits (i.e., $x_k=0$), KU denotes the set of nonbasic fathers fixed at their upper limits (i.e., $x_k=c_k$), and KB denotes the set of basic fathers. The basis B consists of the columns corresponding to basic sons $B_j=A_j$ and of the columns corresponding to basic fathers given by the formula

$$B_k = A_k + \sum_{j\in J(k)\cap JU} c_j A_j$$

8.2. Pricing

Consider a basic solution given by a basis B and sets JL, JU, JB, KL, KU, KB. For the determination of a nonbasic variable to be enter the basis in the simplex algorithm it is necessary to compute the so-called reduced costs. Let z_i denote an ordinary reduced cost connected to the column A_i, i.e.,

$$z_i = f_i - f_B B^{-1} A_i \qquad i\in JK$$

where f_B denotes the basic part of the cost vector f. Due to implicit representation of VUBs the reduced costs associated with several nonbasic variables take then form

$$d_j = z_j \qquad\qquad \text{for } j\in J$$
$$d_k = z_k + \sum_{j\in J(k)\cap JU} c_j z_j \qquad \text{for } k\in K$$

Thus, in comparison with pricing in the standard simplex algorithm, the pricing with implicit representation of VUBs needs a calculation of linear combinations of ordinary reduced costs as the only one additional operation.

Due to the handling of the SUB structure together with the VUB constraints a nonbasic variable x_j or x_k is considered as potential incoming variable if one of the following conditions fulfills:

(A) $d_j<0$ and $j\in JL$,

(B) $d_j>0$ and $j\in JU$,

(C) $d_k<0$ and $k\in KL$,

(D) $d_k>0$ and $k\in KU$.

Implicit representation of VUBs makes some degenerated simplex iterations so simple that they can be performed on line during pricing. Namely, if x_j is an incoming variable and $k(j)\in KL$, then the corresponding simplex iteration depends only on a change in the sets JL and JU, i.e., x_j is moved from the set JL to the set JU or vice versa. Such an operation can be performed while pricing before the computation of reduced costs for fathers.

8.3. Pivoting

Let x_s $(s\in J$ *or* $s\in K)$ be a variable chosen for enter the basis. Consider changes in the basic solution while the value of x_s is either increased for $s\in JL\cup KL$ or decreased for $s\in JU\cup KU$ by a nonnegative parameter Θ. Values of the variables x_j *for* $j\in JL$, x_k *for* $k\in KL$ *and* $k\in KU$ remain on the same level but values of the variables x_k *for* $k\in KB$ and x_j *for* $j\in JB$ as well as values of the variables x_j *for* $j\in JU$ change proportionally to Θ. Namely, the following relations hold

$$x_i(\Theta) = x_i - \Theta\alpha_i \quad \textit{for} \quad i \in KB \cup JB$$
$$x_j(\Theta) = x_j - \Theta\alpha_{k(j)}c_j \quad \textit{for} \quad j \in JU \quad \textit{and} \quad k(j) \in KB$$

where

$$\alpha = \begin{cases} B^{-1}A_s & \textit{for} \quad s \in JL \\ -B^{-1}A_s & \textit{for} \quad s \in JU \\ B^{-1}(A_s + \sum\limits_{j \in J(s)JU} c_j A_j) & \textit{for} \quad s \in KL \\ -B^{-1}(A_s + \sum\limits_{j \in J(s)JU} c_j A_j) & \textit{for} \quad s \in KU \end{cases}$$

Taking into consideration constraints of the linear program we get the following restrictions on Θ :

(1) $x_k - \Theta\alpha_k \geq 0$ *for* $k \in KB$ *and* $x_j - \Theta\alpha_j \geq 0$ *for* $j \in JB$

(2) $x_k - \Theta\alpha_k \leq c_k$ *for* $k \in KB$ *and* $x_j - \Theta\alpha_j \leq c_j c_{k(j)}$ *for* $j \in JB$, $k(j) \in KU$

(3) $x_j - \Theta\alpha_j \leq x_{k(j)}\Theta\alpha_{k(j)}$ *for* $j \in JB$

(4) $\Theta \leq c_s$ *for* $s \in K$ *and* $\Theta \leq c_s c_{k(s)}$ *for* $s \in J$, $k(s) \in KU$

(5) $\Theta \leq c_s(x_{k(s)} - \Theta\alpha_{k(s)})$ *for* $s \in J$, $k(s) \in KB$

(6) $x_j - \Theta\alpha_j \leq c_j(c_s - \Theta)$ *for* $s \in KU$, $j \in J(s) \cap JB$

Hence, we get six formulae for upper bounds on the parameter Θ and six corresponding formulae for determination of the outgoing variable. Crossing these formulae with four types of incoming variables we get 19 types (five criss-crossings are not allowed) of the simplex transformations performed in the algorithm with implicit representation of the VUB & SUB structure. The simplest transformation depends only on moving some variable from one set to another without any change of the basis. Most of the transformations depend on performing one of the following operations:

(a) some basic column multiplied by a scalar is added to another basic column;

(b) some basic column is replaced by a nonbasic column or a linear combination of nonbasic columns.

More complex transformations use both the above operations and the most complex one needs two operations of type (a) and one operation of type (b).

References

[1] Forrest J.J.H., Hirst J.P.H., Tomlin J.A.: Practical solution of large mixed integer programming problems with UMPIRE. Management Science 20 (1974), pp.736-773.

[2] Glover F., Klingman D.: The simplex SON method for LP/embedded network problems. Mathematical Programming Study 15 (1981) pp.148-176.

[3] Glover F., Klingman D.: Basis exchange characterization for the simplex SON algorithm for LP/embedded networks. Mathematical Programming Study 24 (1985) pp.141-157.

[4] Grauer M.: A dynamic interactive decision analysis and support system (DIDASS) - user's guide. WP-83-60. International Institute for Applied Systems Analysis, Laxenburg, Austria (1983).

[5] Grauer M., Lewandowski A., Wierzbicki A..: DIDASS - theory, implementation and experiences. In: Grauer M., Wierzbicki A.P. (eds): Interactive Decision Analysis. Springer, Berlin 1984.

[6] Ignizio J.P.: Goal Programming and Extensions. Heath, Lexington, MA (1976).

[7] Jasinska E., Wojtych E.: Location of depots in a sugar-beet distribution system. EJOR 18 (1984), pp.396-402.

[8] Land A.H., Doig A.G.: An automatic method of solving discrete programming problems. Econometrica 28 (1960) pp. 497-520.

[9] Lewandowski A., Grauer M.: The reference point optimization approach - methods of efficient implementation. WP-82-019, International Institute for Applied Systems Analysis, Laxenburg, Austria (1982).

[10] Mitra G.: Investigation of some branch and bound strategies for the solution of mixed integer linear programs. Mathematical Programming 4 (1973) pp.155-170.

[11] Techniques. McGraw-Hill, New York (1968).

[12] Schrage L.: Implicit representation of variable upper bounds in linear programming. Mathematical Programming Study 4 (1975), pp.118-132.

[13] Schrage L.: Implicit representation of generalized variable upper bounds in linear programming. Mathematical Programming 14 (1978), pp.11-20.

[14] Todd M.J.: An implementation of the simplex method for linear programming problems with variable upper bounds. Mathematical Programming 23 (1982), pp.34-49.

[16] Wierzbicki A.P.: A mathematical basis for satisficing decision making. Math. Modelling 3 (1982), pp. 391-405.

[17] Wierzbicki A.P.: On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spectrum 8 (1986), pp.73-87.

[18] Zeleny M.: The theory of displaced ideal. In: Zeleny M. (ed), Multiple Criteria Decision Making - Kyoto. Springer Verlag, Berlin (1976).

[19] Zorychta K., Ogryczak W.: Linear and Integer Programming - The Branch and Bound Approach (in Polish). WNT, Warsaw, Poland (1981).

A Methodological Guide to the Decision Support System DISCRET for Discrete Alternatives Problems

Janusz Majchrzak

Systems Research Institute, Polish Academy of Sciences

1. INTRODUCTION

1.1. Scope of the report

This report aims to:

- provide the information necessary to use the DISCRET package and to understand its structure as well as the capabilities of the implemented approach,
- discuss such methodological issues associated with the implemented approach, which might be interesting for the user and which justify the chosen approach,
- attract and encourage the reader to take the advantage of the package utilization,

It is assumed that the reader and the package user possess just the very basic information about multicriteria optimization and discrete choice problems.

1.2. Purpose of the DISCRET package

DISCRET is a package created to solve basic multicriteria choice problems in which a finite set of feasible alternatives (and decisions) is explicitly given and, for each alternative, the values of all criteria describing its attributes interesting to the decision maker (DM) were evaluated and listed. The DM is assumed to be rational in the sense that he is looking for an efficient (Pareto-optimal) solution as his final solution of the problem.

Such a discrete multicriteria optimization problem is rather a problem of choice than optimization, since all the information necessary to make a decision is readily available. Such a problem is rather trivial for any human being as long as the number of alternatives is small (say, less than ten or twenty). However, if the number of alternatives and/or criteria grows, the limits of human information processing capabilities are reached and some decision support facilities have to be utilized to guarantee a proper and efficient decision making.

The purpose of the DISCRET package is to support the DM in his search for final decision in an interactive and user-friendly manner. It is assumed that the DM has only a limited knowledge of the problem he wants to solve at the beginning of the session with DISCRET. Therefore, during the session no difficult questions are asked (for example, about criteria trade-offs, DM' utility function or pairwise comparisons of alternatives). The package-provided information enables the DM to gather the experience related to his problem's specific features as well as his own preferences.

The implemented approach seems to be easy to understand and approve even for a user who is not very familiar with multicriteria optimization techniques.

The DISCRET package has been designed to solve medium-size discrete multicriteria problems with the number of alternatives ranging from few hundreds to few thousands. The number of criteria is in the current version restricted to 20 (mainly due to the limitations of display facilities).

During the session the user controls the decision-making process by choosing suitable options from the displayed "menu". Therefore, he does not have to learn and remember any command pseudo-language. This feature, together with special procedures for handling user's mistakes and with self-explanatory package messages, makes the package user-friendly and allows for an unexperienced user.

1.3. Fields of the package applications

In many real-life problems, decision variables take their values from a discrete set rather than from a continuous interval. Usually, there is a finite number of available facility location sites, the facility size or production capability may be chosen from a discrete set of values, during a design process the components are chosen from a set of typical elements available on the market, etc. Such problems form the "natural" field of applications for the DISCRET package.

Another field of possible applications of the DISCRET package consists of cases in which the original problem is actually continuous (rather than discrete) but the analysis restricted just to a finite number of alternatives appearing in this problem may be interesting and useful for the DM, since it may result in an enlightening and a more precise definition of his preferences, region of interest or aspiration levels.

Situations falling under the latter category may occur for at least two following reasons. Firstly, if a sample of alternatives together with the corresponding criteria values is readily available, the utilization of the DISCRET package may enable the DM to gain an insight into the original multicriteria problem. The analysis of an assembly of runs of a simulation model is an example of this case. Secondly, for the purpose of an initial analysis of a problem in which the decision variables actually take their values from continuous intervals, the DM may take into consideration just a few values for each decision variable or to generate a random sample of alternatives.

An encouraging factor that may attract the DM is the fact that the DISCRET package makes no restrictions on the forms of the criteria. Therefore, attributes as complicated as required may be considered.

2. BACKGROUND

2.1. The discrete multicriteria optimization problem

Package DISCRET has been created to support – in an interactive manner – multicriteria optimization and decision making for problems with a finite number of discrete alternatives. Such problems are frequently referred to as implicit constraints or explicit alternatives problems.

Let us consider the following discrete multicriteria optimization problem (DMOP). It is assumed that a set X^0 of feasible discrete alternatives is explicitly given and for each of its elements all criteria under consideration have been evaluated. The criteria values for all feasible discrete alternatives form the set Q of feasible outcomes or evaluation.

$$\min_{x \in X^0} f(x)$$

$$X^0 = \{x_1, x_2, \ldots, x_n\} \subset X = R^s$$

$$f(x) = (f^1(x), f^2(x), \ldots, f^m(x))$$

$$f : X^0 \rightarrow Q$$

$$Q = \{f_1, f_2, \ldots, f_n\} \subset F = R^m$$
$$f_j = f(x_j) \; , \quad j = 1,2,\ldots,n$$

Furthermore, it is assumed that a domination cone Λ is defined in the objective space F. As in most applications the positive orthant is considered, $\Lambda = R^m_+$ and $\tilde{\Lambda} = R^m_+ \backslash \{0\}$. The domination cone introduces the partial pre-order relation "$\prec$" into the objective space:

$$\forall \; f_1 \, , f_2 \in F \quad , \quad f_1 \prec f_2 \quad <=> \quad f_1 \in f_2 - \tilde{\Lambda}$$

The element f_1 dominates f_2 in the sense of the partial pre-order induced by the domination cone Λ.

Element $\bar{f} \in Q$ is nondominated in the set of feasible elements Q, if it is not dominated by any other feasible element. Let $N = N(Q) \subset Q$ denote the set of all nondominated outcomes in the objective space and let $N_X = N(X^0) \subset X^0$ denote the set of the corresponding nondominated alternatives (decisions) in the decision space. To solve the DMOP it means to find the set N of nondominated outcomes and the corresponding set N_X of nondominated decisions.

Notice that DMOP is described by the two sets Q and X^0 defined above (together with m, n and s). Therefore the package input files supplied by the user must contain these two sets.

Observe also that no assumptions were made about the nature of the criteria functions f_i. In fact, the only requirement for them is that they should assign numerical values to the alternatives, indicating their attractiveness with respect to the attribute under consideration. In particular, the criteria functions may be of the quantitative type. The single restriction is that values assigned to alternatives by criteria should be expressed by numbers and that the user is able to indicate whether he wishes to increase or decrease these numbers.

2.2 Overview of existing approaches

The discrete multicriteria optimization problem (DMOP) is a combinatorial problem involving sorting and one could expect a large number of papers in the bibliography devoted to this subject. However, the problem did not focus much attention of the researchers - except in its utility theoretical variant that actually transforms the problem to a single-criteria one - and the bibliography we are able to point at consists only of [7]-[9], plus some reports of the earlier research summarized there.

The insignificant interest in methods for solving DMOP could be explained by the fact that the solution of the DMOP, the whole set of nondominated alternatives is not the solution of the multicriteria decision making problem (MCDMP), a selected preferred alternative. However, since the efficiency of methods dealing with MCDMP usually depend on the number of alternatives, it is wise to reject the dominated alternatives.

A rather large number of approaches have been suggested for the solution of the MCDMP involving discrete alternatives. They differ both in the problem formulation and the assumptions about the decision maker (DM). Let us mention here just some most interesting ones. The method suggested in R.L. Keeney, H. Raiffa [17] is based on utility functions constructed first for each criterion and then combined into a global utility function. In S. Zionts [16] a linear, while in M. Koksalan et al. [13] a quasiconvex underlying utility function of the DM is assumed and the best alternative according to an approximation of this utility function is found by asking for answers to a number of comparisons between pairs of alternatives. Other methods, e.g. B.Roy [18] or J. Siskos [19], are based on outranking relations. In P. Rivett [11], multidimensional scaling techniques are used to obtain a graph pointing from least to most preferred alternatives.

Other group of approaches (some of them were proposed originally for some different problems) is based on an observation that if the number of the alternatives is small, then the DM is able to make a decision intuitively, without any formalism of expressing his preferences. If the number of alternatives is larger, then one has to reduce it for the DM by selecting a small but representative sample. Several methods for obtaining such a representation were proposed. They utilize cluster analysis (A.A. Torn [4], J.N. Morse [14]), filtering (R.E. Steuer, F.W. Harris [10]) or random sampling (S. Baum et al. [15]).

Approaches from the first of the two above-mentioned groups place the burden on the DM. He is asked to supply the information about his preferences by the evaluation of the alternatives - by pairwise comparisons or rankings for example. These evaluations are substantial for the methods. Each of these methods is based on certain implicit or explicit assumptions about the DM, such that, for example, he has an utility function expressing his preferences. The size of the problems that can be solved is limited by the DM's ability to provide the required amount of information by ranking or comparing pairwise the alternatives.

In the approaches from the second group, the burden is placed rather on the computer. The crucial point here is whether the obtained representation of the nondominated set will be illustrative for the DM. No special assumptions about the DM are made. He is only expected to prefer the nondominated alternatives rather than the dominated ones.

Our approach presented in this report may be classified as one of the second group. It is based on a new efficient method for DMOP, which also can efficiently produce a representation of the nondominated set.

2.3. The method of dominated approximations

The implemented method is of the explicit enumeration type. It is called the method of dominated approximations and is based on the following concept.

Def.1 Let Q be the set of all feasible alternative outcomes, N the set of corresponding nondominated alternative outcomes and Λ the domination cone. Set A is called a dominated approximation of N iff

$$N \subset A - \Lambda$$

In other words, A is a dominated approximation of N iff for each $f_i \in N$ there exists $f_j \in A$ such, that $f_i \prec f_j$ in the sense of the partial pre-order induced by Λ.

We will say that the approximation A_2 dominates the approximation A_1 of the nondominated set N iff

$$A_1 \subset A_2 + \Lambda$$

Hence, as the worst approximation of N we can consider the entire set Q, while the best approximation is the set N itself. The method of dominated approximations generates a sequence of approximations A_k , $k = 0,1,2,...,l$ such that

$$Q = A_0 \supset A_1 \supset A_1 \supset A_2 \supset \cdots \supset A_k \supset \cdots \supset A_l = N$$

The method of dominated approximations

Given Q and Λ we are supposed to determine $N = N(Q)$. Assume that all criteria are to be minimized.

Step_0 Let $A_0 = Q$, $N_0 = \phi$, $k = 0$.

Step_1 If $A_k \setminus N_k = \phi$ then STOP with $N_k = N$, else choose any index $i \in I = \{1,2,...,m\}$ and find $\bar{f} \in Q$ such that the i-th component of it is minimal in $A_k \setminus N_k$:

$$\bar{f}^i = \min_{A_k \setminus N_k} f^i$$

(See Remark 2).

Set $N_{k+1} = N_k \bigcup \{\bar{f}\}$.

Step_2 Create the new approximation A_{k+1} by rejecting from $A_k \setminus N_{k+1}$ all elements dominated by $\bar{f}$ (See Remark 1)

$$A_{k+1} = (\{A_k \setminus N_{k+1}\} \setminus \{(\bar{f} + \tilde{\Lambda}) \bigcap (A_k \setminus N_{k+1}\}) \bigcup N_{k+1}$$

Set $k = k + 1$ and go to Step_1.

Remark 1. While rejecting the elements dominated by $\bar{f}$ it is sufficient to compare elements of the set $A_k \setminus N_{k+1}$ with $\bar{f}$ according to all but i-th criterion, since $\bar{f}^i$ is minimal among all f^i in $A_k \setminus N$.

Remark 2. The minimum may happen to be non-unique. Let B be the set of those elements $f_j \in A_k \setminus N_k$ for which f_j^i appear to be minimal in $A_k \setminus N_k$. Actually not all elements of B are nondominated. One has to solve the following problem. Given B and Λ select $N(B)$. The above presented method may be used for this task with $A_0 = B$ and $I = I \setminus \{i\}$. This recurrence is applied until an unique minimum is found in the Step_1 of the algorithm. Then, after the execution of Step_2 one has to return to the lower level of recurrence. On each level *Remark 1* holds.

Note that if the recursion described in *Remark2* would not be applied, then the set of weakly nondominated alternatives would be determined by the above algorithm.

In order to measure the efficiency of the method, let us consider the number of scalar comparisons $S(m,n,p)$ required by the method to solve the DMOP with m criteria, n feasible alternatives and p nondominated alternatives. From the analysis of the method one can easily obtain

$$S(m,n,p) \leq \frac{m\,p}{2}\,(2n - p - 1)$$

As one can see, the method solves easily problems with small p. In practical problems p is usually a small fraction of n; the worst case is for $p = n$, i.e. when all alternatives are nondominated. Note that the performance of the method does not depend on the permutation of the alternatives.

2.4. Selection of the representation of the nondominated set

The biggest advantage of the method of dominated approximations is its ability to select a representation of the nondominated set N instead of the entire set N. Unlike other known approaches which find the entire nondominated set first and then select a representation (differently defined for each of those methods), the presented method selects a representation at once. This fact profits inefficiency.

Let t_i , $i = 1,...,m$ be some given tolerance coefficient for the m criteria under consideration, $t_i \geq 0$, and $T_i = (t_1, t_2, \ldots, t_{i-1}, 0, t_{i+1}, \ldots, t_m)$ be a vector in the objective space. For the sake of simplicity let us assume that all criteria are to be minimized. The following modification of the method of dominated approximations suffices to obtain a representation instead of the whole nondominated set. In the Step_2 of the method not only the elements dominated by the nondominated element $\bar{f}$ (found in the Step_1 by minimization over the i-th criterion values) have to be rejected, but also elements dominated by $f = \bar{f} - T_i$. Hence, in Step_2 is modified to:

$$A_{k+1} = (\{ A_k \setminus N_{k+1} \} \setminus \{(\bar{f} - T_i + \tilde{\Lambda}) \bigcap (A_k \setminus N_{k+1})\} \bigcup N_{k+1}$$

Observe that because the representation contains less elements than the nondominated set, it will be obtained with a smaller computational effort. Figure 1 illustrates the role of the tolerance coefficients in the process of selecting a representation.

The author is not aware of the existence of any other methods that could be effectively applied for a problem with few hundreds or few thousands of alternatives.

2.5. Outline of the approach and introduction to DISCRET

To start the session with DISCRET the user has to supply the file containing set Q of the criteria values for all feasible alternatives and (optionally) the file containing the set X^0 of feasible decisions. These files, called the data and the additional data file respectively, describe the problem under consideration.

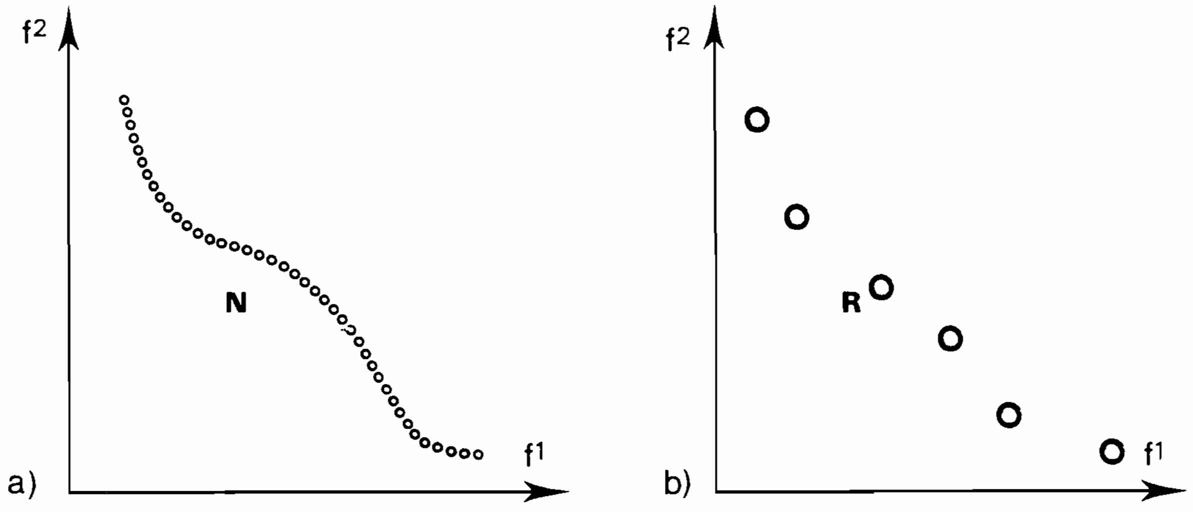
The next step is the problem specification phase in which the user is asked for some more detailed information about the problem such as the total number of criteria, the number of alternatives and the additional data on dimensionality (the command *specify*). He is also asked to indicate for each criterion whether it should be minimized, maximized or ignored and to specify the criterion value tolerance. If, for two alternatives, the difference of criterion values is lower than the criterion tolerance, then the criterion values are assumed to be equal (during the problem solving phase - the command *solve*). This mechanism of tolerances is also used to obtain a representation of the set of nondominated solutions.

After the problem generation and implementation phase the user may obtain the information about the criteria values ranges and he may put the lower and/or upper bounds on the values of some/all criteria (the
bounds command).

The bounds setting may be utilized by the user for several purposes. This is the list of some most relevant:

- to eliminate irrelevant alternatives from further considerations,
- to specify his current region of interest in the objective space,
- to redefine his problem as a problem with a fewer criteria as the original one (as in the method of equality/inequality constraints - see [6]), for example, a bicriteria problem.

In the next step the user may run the DMOP solver (by executing the command *solve*) to eliminate the dominated alternatives by an explicit enumeration technique. The tolerances for criteria values play an important role here. If they are all equal zero or have small positive values that correspond to indifference limits of the DM's for criteria values, the whole set of the nondominated solutions will be obtained. If the values of tolerances are equal to some significant fractions of the corresponding criteria ranges, then a



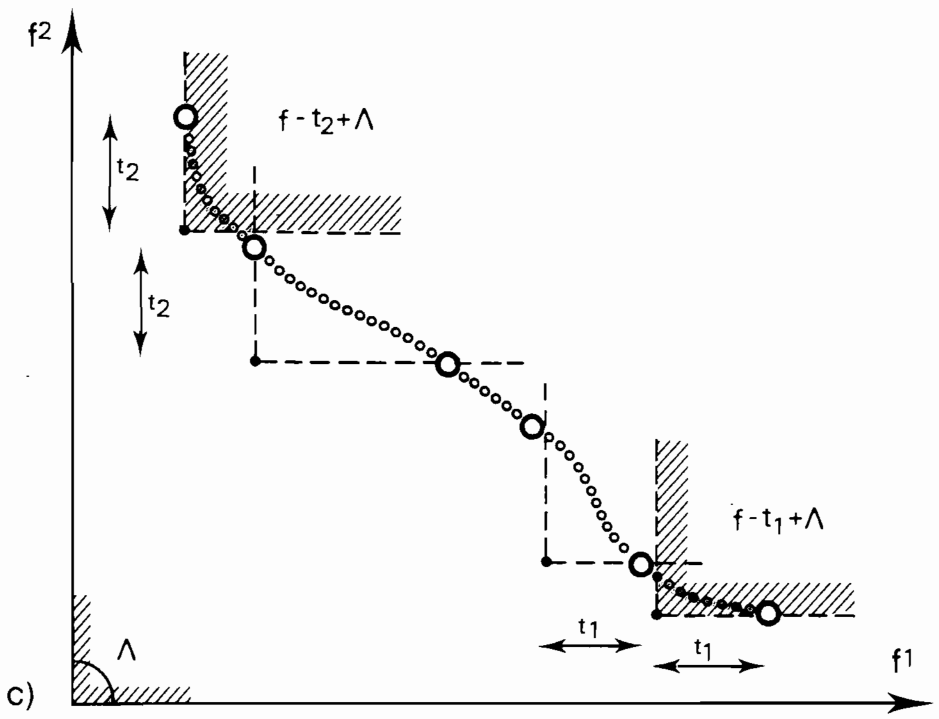


Figure 1. Selection of the representation $R = R(N) = R(Q)$ of the nondominated set $N = N(Q)$. Only nondominated elements are marked for the sake of simplicity of illustration.

a) the nondominated set N.

b) the representation R of the set N.

c) illustration of the tolerance mechanism.

representation of the set of nondominated solutions will be obtained. The representation is a subset of the set of nondominated solutions preserving its shape and containing the smaller number of elements, the larger were chosen tolerance coefficients.

The nondominated set or its representation may be sorted according to some or all criteria values (command *sort*) to provide a better analysis. As a result of this analysis, the user may proceed in one of the following paths:

- choose a new region of his interest by a proper bounds setting (by using the command ***bounds***),
- obtain a more or less dense representation by decreasing or increasing the tolerances (by using the commands ***specify, solve***),
- use graphic display to learn more about the problem and utilize the reference point approach (by using the command ***analyse***).

It is worth to mention here that – unlike in other known techniques of obtaining a representation of the nondominated set – our approach not only does not require any additional computational effort but even decreases the time of computation with the ratio of #R to #N, where #N and #R are the number of elements in the nondominated set N and its representation R, respectively.

Once any subset of the set N of nondominated solutions has been obtained, one can select the corresponding decisions from the the additional data file (the command ***pick***).

The DISCRET package provides also some more detailed information about the problem under consideration. A nondominated and a dominated linear approximations of the set of nondominated solutions are calculated (the command ***analyse***). These approximations are obtained in the following way. A linear function is defined by the combination of the criteria with coefficients determined by the criteria ranges. This function is then minimized and maximized over the set of nondominated elements to obtain the nondominated and dominated approximation, respectively.

The information contained in the lower and upper bounds for criteria, in criteria ranges, in nondominated and dominated approximations and the corresponding solutions gives a good overview of the shape of nondominated set. To learn more about the variety of available alternatives, the user may use another facility provided by the DISCRET package (in the command ***analyse***), namely the graphical display of two-dimensional subproblems on the terminal screen. The user chooses two criteria for the vertical and horizontal axes, while the other criteria are:

- left unbounded - the whole problem is projected on the two-dimensional subspace of the space of objectives, just as if all but the two selected criteria were ignored,
- restrictively bounded - a two-dimensional "slice" is cut out of the original m-dimensional problem.

Enlargements of the chosen display fragments may be obtained simply by specifying new bounds for the criteria on the axes. Another display option indicates how many elements does each of the 800 display points represent. This feature may be useful to detect and investigate the cluster structure of the problem.

The powerful tool of the reference point approach [1] is also available for the user (in the command ***analyse***). By determining a reference point, he exhibits his aspiration levels for criteria values, confronts them with the obtained solution and modifies them and the reference point. The graphical displays mentioned above could also be useful on this stage of the decision making process.

During a session with DISCRET the user does not have to necessarily follow the entire procedure presented above. Once the problem generation and specification phase has been completed, he may utilize the package facilities in any order, repeat some steps (commands) or their sequences.

The ability of ignoring some of the criteria temporarily (by specifying that they are to be neither minimized nor maximized) opens to the DM a possibility of using a lexicographic or group-lexicographic approach. He may also, besides the actual criteria, introduce in an identical way some additional criterion expressing his utility, goal or preference function or any global criterion and use them on any arbitrary chosen stage of the decision making process. Such additional criteria have to be evaluated for each alternative during the problem generation phase (just as in the case of the original criteria).

The package offers also the possibility of an immediate return to any of the previous stages of the session, provided that the corresponding files were not removed by the user.

3. STRUCTURE AND FEATURES OF THE PACKAGE

3.1. General description

The current pilot version of the DISCRET package consists of eight FORTRAN77 programs. In order to run any of them the user has to type an appropriate program name (command) on his terminal. A list of DISCRET programs is presented below.

- *test1* - first test problem generator (the Dyer's Engine Selection Problem).
- *test2* - second test problem generator (the location-allocation problem).
- *specify* - supports a detailed specification of the user problem.
- *bounds* - informs about the criteria values ranges (utopia and nadir points), nondominated and dominated approximations of the set of alternatives and supports setting of new bounds on criteria values.
- *solve* - solves the discrete multicriteria optimization problem with explicit alternatives (implicit constraints), i.e. finds the set of nondominated or weakly nondominated elements or its representation, keeping or rejecting duplicate elements.
- *analyse* - supports the reference point approach and simple graphic displays of the nondominaned set.
- *sort* - sorts the alternatives in increasing/decreasing order with respect to the values of a specified criterion.
- *pick* - finds decisions corresponding to the chosen outcomes in criteria space.

During the command execution, the user controls the process by choosing suitable items from the displayed menu (a list of options available at the moment). The menu system has been chosen instead of a pseudo-language of control commands because it does not require from the user to learn and remember a set of commands.

Each menu contains an amount of information sufficient to make the decision which of the displayed options is the most suitable one. If the user is asked to enter some information, everything he types is checked. If he makes a mistake, a message is displayed on the screen. Usually the message not only indicates the error but also shows the correct form of the required input.

In the next chapters the package commands will be briefly presented. We will not go into details of each menu since they are self-explanatory. The user will gather all the necessary experience during an introductory session with DISCRET. The test problems may be created by the commands *test1* and *test2*. The description of the test problems can be found in the user's training manual.

3.2. Problem specification phase

The command *specify* either creates the specification file for the user's problem from scratch or modifies an already existing one. In each of these two cases, by replying to the displayed questions, the user will provide all the necessary information describing specific aspects of his problem in detail without bothering about the format and order of the entered information.

3.3. The bounds setting phase

The command *bounds* reads the input data file and input specification file, evaluates the criteria values ranges and displays them together with the nondominated and dominated approximation of the set of alternatives. If the user is not satisfied with the ranges of criteria values or with the values of approximations he can change them.

Knowing the ranges of criteria values, the DM may decide that some of the values of criteria does not interest him at all or at least temporarily. The command *bounds* makes it possible to change the DM's region of interest. By setting the appropriate lower and/or upper bounds for criteria values, the DM restricts further considerations to a smaller region of the objective space - his current region of interest.

Only these alternatives that satisfy the bounds will be contained in the output data file produced by the command *bounds*. Also sections SPSA and BOUNDS of the specification file are modified.

Notice that the command *bounds* can select only a subset of alternatives from the input data file. If the DM wants to consider a completely different region of interest, he has to supply the input data file containing that set of alternatives.

To illustrate this point assume, just for the sake of simplicity, that all criteria are to be minimized. Observe that if the decreasing of an upper bound for one criterion results in increase of the lower value for some other criterion, then it indicates that a part of the nondominated set did not satisfy the bounds and was rejected. If this was not the purpose of the user, he should return to less restrictive bounds. This remark may be useful on the initial stage of the problem analysis, when the user should become aquitanted with the entire variety of the available alternatives.

3.4. The DMOP solving phase

The command *solve* results in solving the DMOP i.e. it selects the nondominated outcomes out of the set of feasible solutions. If the tolerances for all criteria values (contained in the section SPSB in the input specification file) are equal to zero or have some small positive values corresponding to the computer arithmetic accuracy (for example, 1.0e-10), then all nondominated outcomes are found. If the tolerances have larger positive values equal to some significant fractions of the criteria ranges, then just a subset of the nondominated set, called its representation, is selected.

The command *solve* asks the user also about the type of the solution he is looking for. It has the ability to find either the set of nondominated outcomes or weakly nondominated outcomes. If there are duplicate outcomes (that is, if the same outcome vector corresponds to two different decisions), then they can be treated as distinguished ones (and all preserved) or as identical ones (and all but one rejected). Options more sophisticated than the default option (nondominated outcomes, duplicates rejected) do make sense in the cases when some of the criteria are more important then the other.

3.5. The phase of selecting final solution

Once the nondominated set (or its representation or a part of it corresponding to the current region of interest of the user selected by setting of bounds) has been obtained, the user may wish to list its elements and analyse them.

The command *sort* sorts the elements of the input data file according to increasing or decreasing values of criteria chosen by the user. Another option is to sort the alternatives in increasing or decreasing order according to their identifiers. When sorted before being printed, any set of alternatives appears to be more readable and hence more useful for analysis.

The command *pick* selects from the additional input data file any additional information corresponding to the elements contained in the data file. Typically, this additional information describes the decisions leading to the obtained nondominated solutions.

The mechanism provided by the commands *sort* and *pick* may be especially useful in the case when the package user is an analyst. Properly sorted data (a nondominated set representation adequate to the current stage of the decision making process) will be more readable for the DM.

The command *analyse* was designed to help the user to define his region of interest in a more precise way or to find his final solution.

At the beginning, the user will be informed about the criteria best and worse values - the utopia and nadir points. In order to provide some more detailed but still aggregated information about the the shape of the nondominated set (or its representation or just a part of it) the nondominated and dominated linear approximations are evaluated.

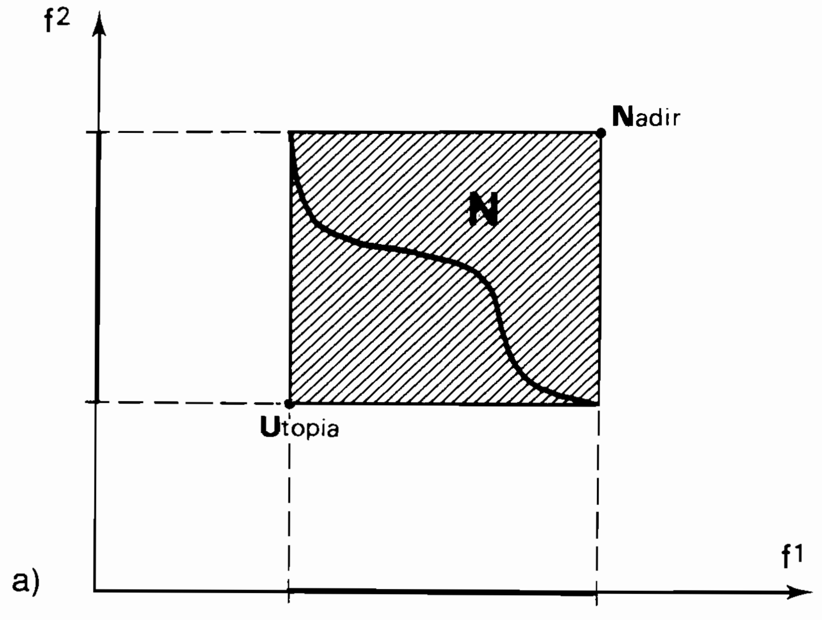
A linear combination of criteria with coefficients proportional to the criteria ranges is minimized and maximized over the nondominated set to obtain its nondominated and dominated approximation respectively. Each of these approximations may be characterized by a single parameter standing for the percentage of the range it cuts off out of each criterion values range, see Figure 2 for illustration. Solutions obtained from the linear approximations are also displayed. This aggregated information seems to provide good aggregate data on the shape of the nondominated set, no matter how many criteria are under considerations.

In order to learn more about the criteria trade-offs, the user may display on the screen of his terminal a simple graphic figure for a two-dimensional subproblem. By setting bounds on all but two criteria he is able to cut a "slice" out of the m-dimensional problem. The entire subset selected in this way will be represented by 800 fields on the screen.

Finally, the user may enter the reference point approach, interactively introduce reference point exhibiting his aspiration levels for criteria values and analyse the obtained solutions. The reference points need not to be attainable and the obtained solution is the nondominated point nearest to the reference point in the sense of the scalarizing function. A scalarizing function based on the Euclidean-norm is used. Let q be the reference point introduced by the user. Then, assuming that all criteria are to be minimized, the following scalarizing function is minimized:

$$S(f - q) = - \|f - q\|^2 + \rho\|(f - q)_+\|^2$$

where $(f - q)_+$ denotes the vector with components $max(0 , f - q)$, $\| \bullet \|$ denotes the Euclidean norm and $\rho > 1$ is a penalty scalarizing coefficient. See Wierzbicki [1], for example, for more information about the reference point approach.



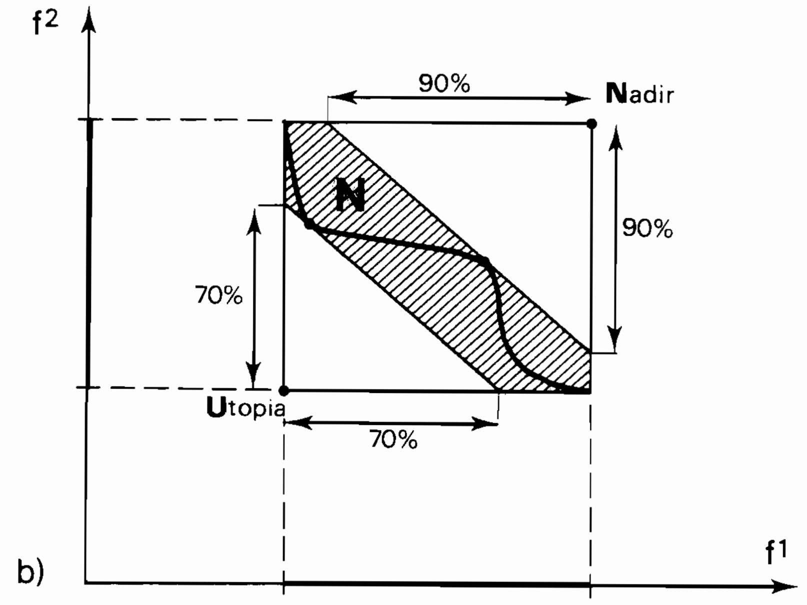


Figure 2. Two types of the aggregated information about the nondominated set N.

a) Information about the nondominated set N offered by the ***utopia*** point and the ***nadir*** point.

b) Information carried by the nondominated (70%) and dominated (90% of criteria range) approximations of the set N.

4. TEST EXAMPLES

4.1. The Dyer's "Engine Selection Problem"

For the purpose of testing the package and to be used during introductory sessions with DISCRET, a generator of the Dyer's "Engine Selection Problem" (see [3] or [4]) has been implemented. This is a very simple example of a DMOP. However, it is rather well known in the literature devoted to this field of research and therefore it seems that it will suit well as a small illustrative test problem.

Let us consider a DM who designs a new automobile and he has to choose an engine for that car. Suppose that the variety of available engines is described by three parameters (decision variables):

x_1 - compression ratio

x_2 - carburation ratio (in square inches)

x_3 - piston displacement (in cubic inches)

Suppose that the DM's preferences are described by the following three criteria:

f_1 - cost of the engine

f_2 - horsepower

f_3 - mileage per gallon

The following DMOP was proposed by Dyer [3] in 1973 - see also Törn [4] (1980).

Problem definition:

$$\begin{aligned} &\textit{minimize}: & f_1(x) &= 133(x_1 - 8) + 10x_2 + x_3 + 2000 \\ &\textit{maximize}: & f_2(x) &= 20(x_1 - 8) + x_2 + 0.5x_3 \\ &\textit{maximize}: & f_3(x) &= -1.67(x_1 - 8) - 0.2x_2 - 0.05x_3 + 35 \end{aligned}$$

subject to:

bounds:

$$\begin{aligned} 8 \leq x_1 &\leq 11 \\ x_2 &\leq 40 \\ 100 \leq x_3 &\leq 200 \end{aligned}$$

constraints:

$$\begin{aligned} 50x_1 - 30x_2 + x_3 &\leq 400 \\ 20x_1 - 3x_2 &\leq 160 \\ \frac{x_3}{x_2} &\leq 20 \end{aligned}$$

Problem generation:

Dyer and Törn proposed the following scheme to generate uniformly a set of decisions :

```
for $x_1 = l_1$ step $s_1$ until $u_1$
for $x_2 = l_2$ step $s_2$ until $u_2$
for $x_3 = l_3$ step $s_3$ until $u_3$
```

where l_i , u_i are the lower and upper bounds for x_i , i=1,2,3 while s_i are the corresponding step size. If – following Dyer and Törn – the initial data are:

	l_i	s_i	u_i
i=1	8	1	11
i=2	10	10	40
i=3	100	20	200

then 84 generated points satisfy the problem constraints. Another way to generate a test problem is a random generation of decision vectors x within bounds l and u. This test problem is generated by the DISCRET's command *test1*.

4.2. The location-allocation problem

The second test problem is a facility location-allocation problem. It is based on the problem presented by Lee, Green and Kim [5].

A firm is evaluating six potential sites of plant location (in four different states) that would serve four customer centers. The problem is where should the plants be opened and what should be the production volume of each of the new opened plants. Let $i = 1,2,\ldots,i_{\max}=6$ be the locations index and let $j = 1,2,\ldots,j_{\max}=4$ be the customer center index.

Decision variables:

$y_i = 0/1$ if a plant is not opened / opened at location i,

z_i – production volume (size) of a plant opened at location i.

Model variables and parameters:

p_j – total demand of customer center j,

c_{ij} – unit transportation cost from facility i to the customer center j,

g_i – fixed cost of opening a facility at location i (in \$1000),

l_i – life quality score for location i,

z_i^u – production upper limit for facility at location i (due to the state environment quality standards),

z_i^l – production lower limit,

z_i^s – production increment step size,

k^i – location i production limits due to state environment quality standards,

d_{ij} – demand placed on facility i by the customer center j,

$$d_{ij} = \frac{y_i \, e^{-bc_{ij}}}{\sum_i y_i \, e^{-bc_{ij}}} \quad ,$$

x_{ij} – quantity of units transported from location i to the customer center j,

$$x_{ij} = \frac{d_{ij}}{\sum_j d_{ij}} \min\{z_i \, , \, \sum_j d_{ij}\} \quad ,$$

n — number of opened facilities.

Constraints:

1. Fixed cost limitation (in \$1000):

$$\sum_i g_i y_i \leq 2000$$

2. Production limitations due to state environment quality standard:

$$z_i \leq k_i \ , \quad i = 1, \ldots, i_{\max}$$

3. Favored customer center service level :

$$z_1 y_1 + z_2 y_2 \geq 50$$

4. Number of opened facilities :

$$n_{\min} = 1 \leq \ n \ \leq 3 = n_{\max}$$

Criteria:

1. Unsatisfied demand level :

$$\min \quad f_1 = \sum_i \sum_j (d_{ij} - x_{ij})$$

2. Favored customer center (no. 1) service level :

$$\max \quad f_2 = \frac{\sum_i y_i x_{i1}}{p_1}$$

3. Total cost :

$$\min \quad f_3 = f_5 + f_6 + f_7$$

4. Average life quality score :

$$\max \quad f_4 = \frac{\sum_i y_i l_i}{\sum_i y_i}$$

5. Fixed cost :

$$\min \quad f_5 = \sum_i y_i g_i$$

6. Transportation cost :

$$\min \quad f_6 = \sum_i \sum_j s_{ij} x_{ij}$$

7. Production cost :

$$\min \quad f_7 = \sum_i z_i e^{-\alpha z_i}$$

8. Unsaled production :

$$\min \quad f_8 = \sum_i \max\{0 \ , \ (z_i - \sum_i x_{ij})\}$$

Alternatives generation scheme:

The set of feasible alternatives is generated by the following three nested loops.

1. Consider opening $n = n_{\min}, \ldots, n_{\max}$ facilities.

2. Generate all n locations subsets of the set of locations (n - elements combinations of $i_{\max}$ elements set).

3. For each facility opened at location i consider its all available sizes z_i ranging from z_i^l to z_i^u with the increment step size z_i^s.

4.3 How to get started

At the very beginning of the session a problem to be solved has to be supplied. For the first session execute the DISCRET command *test1*. When the test problem is already generated, look at three files that were produced: the specification file, the data file and the additional data file. Only the data file (and optionally the additional data file) has to be supplied in order to start a session.

Whenever you do not remember the names of the files you have created during the session, display the history.fil from your current directory. This file contains the history of your session.

In order to learn how to describe the details of your problem for DISCRET, print the specification file produced by the command *test1*. Then execute the specify command and try to create a specification file identical to that obtained from *test1*.

If you already know how to specify your problem, try some other DISCRET commands. For the first time, execute them in the following order : *bounds, solve, analyse, sort, pick,* just to learn what they can actually do for you.

Later on try to select your most preferable solution(s). Notice that DISCRET commands can be executed in any order (if only it does make any sense for you). Referee to the *history.fil* to recall the history of your session.

5. CONCLUSIONS

The DISCRET package for multicriteria optimization and decision making problems with finite number of discrete alternatives has been briefly presented. It is the author's hope that this report will attract the reader and encourage him to use the package.

DISCRET is an interactive package. The user may execute its commands in any order once the problem generation and specification phase has been completed. The variety of paths the user may follow guarantees flexibility in meeting his demands.

The author will be grateful for any critical remarks and comments concerning both the approach and the package itself. All such suggestions would be very helpful and may result in further package improvements.

REFERENCES

[1] A.P. Wierzbicki, A methodological guide to multiobjective decision making, *WP-79-122, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1979.*

[2] J. Majchrzak, Package DISCRET for multicriteria optimization and decision making problems with discrete alternatives, *IIASA Conference on Plural Rationality and Interactive Decision Processes, Sopron, Hungary, 16-26 August, 1984.*

[3] J.S. Dyer, An empirical investigation of a man-machine interactive approach to the solution of a multiple criteria problem, *in Multiple Criteria Decision Making, T.L.Cochrane and M.Zeleny (eds.)*, University of South California Press, 1973.

[4] A.A. Törn, A sampling-search-clustering approach for exploring the feasible/efficient solutions of MCDM problems, *Comput. and Ops Res., Vol. 7, No 1-2, pp. 67-79, 1980.*

[5] S.M. Lee, G.I. Green, C.S. Kim, A multiple criteria model for the location-allocation problem, *Comput. and Ops Res., Vol. 8, pp.1-8, 1981.*

[6] J.G. Lin, Three methods for determining Pareto-optimal solutions of multiple-objective problems, *in Ho and Mitter (eds.), Directions in Large-Scale Systems. Many-Person Optimization and Decentralized Control*, Plenum Press, New York and London, 1976.

[7] H.T. Kung, F. Luccio, F.P. Preparata, On finding the maxima of a set of vectors, *Journal of the Association for Computing Machinery, Vol. 22, No. 4, pp. 469-476, 1975.*

[8] E. Polak, A.N. Payne, On multicriteria optimization, *in Ho and Mitter (eds.), Directions in Large-Scale Systems. Many-Person Optimization and Decentralized Control,* Plenum Press, New York and London, 1976.

[9] H. Stahn, U. Petersohn, Discrete polyoptimization, *Systems Science, Vol. 4, No. 2, pp.101-109, 1978.*

[10] R.E. Steuer, F.W. Harris, Intra-set point generation and filtering in decision and criterion space, *Comput. and Ops Res., Vol. 7, No. 1-2, pp. 41-53, 1980.*

[11] P. Rivett, Multidimension scaling for multiobjective policies, *Omega, Vol. 5, pp. 367-379, 1977.*

[12] A.N. Payne, E. Polak, An interactive rectangle elimination method for biobjective decision making, *IEEE Transactions on Automatic Control, Vol. AC-25, No. 3, pp. 421-432, 1980.*

[13] M. Köksalan, M.H.Karwan, S. Zionts, An improved method for solving multiple criteria problems involving discrete alternatives, *IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-14, No. 1, pp. 24-34, 1984.*

[14] J.N. Morse, Reducing the size of the nondominated set: pruning by clustering, *Comput. and Ops Res., Vol. 7, No. 1-2, pp. 55-66, 1980.*

[15] S. Baum, W. Terry, U. Parekh, Random sampling approach to MCDM, *in J.N. Morse (ed.), Organisations: Multiple Agents with Multiple Criteria, Lecture Notes in Economics and Math. Systems, 190.*

[16] S. Zionts, A multiple criteria method for choosing among discrete alternatives, *Eur. J. Op. Res., Vol. 7, pp.D 143-147, 1981.*

[17] R. L. Keeney, H. Reiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs, New York, Wiley, 1976.*

[18] B. Roy, Problems and methods with multiple objective functions, *Math. Programming, Vol. 1, pp.239-266, 1971.*

[19] J. Siskos, A way to deal with fuzzy preferences in multi-criteria decision problems, *Eur. J. Op. Res., Vol. 10, pp. 314-324, 1982.*

Nonlinear Model Generator

J.Paczynski, T.Kreglewski

Institute of Automatic Control, Warsaw University of Technology

ABSTRACT

This paper presents introductory documentation and a theoretical manual for a nonlinear model generator aimed to help in formulation and evaluation of nonlinear models for applications in interactive decision analysis and support systems implemented on professional microcomputers. It was developed in 1986 in the Institute of Automatic Control, Warsaw University of Technology, under a contracted study agreement with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis. It is the first proposal of a standard for models' formulation, easy to use for non-computer specialists. Definition and edition of models is done in an easy but flexible format of a spreadsheet. All derivatives necessary for a robust optimization algorithm are automatically calculated by a symbolic differentiation package. This model generator was used in the IAC-DIDAS-N system presented in this report.

A. INTRODUCTORY DOCUMENTATION

A1. EXECUTIVE SUMMARY

In many situations a decision maker needs help of an analyst or a team of analysts to learn about possible decision options and their predicted results. Such situations include both purely engineering complex design and mixed social economical and environmental problems. A team of analysts frequently summarizes its knowledge in the form of a substantive model of the decision problem that can be formalized mathematically and computerized. A mathematical model can never be perfect but, nevertheless, it can often be of great help to a decision maker in the process of learning about novel aspects of a decision situation.

The learning process needs interaction of a decision maker with a team of analysts and substantive models prepared by them. In organizing such interaction, many techniques of optimization,multicriteria decision analysis and other tools of mathematical programming can be used. However, all such techniques must be used as supporting tools of interactive analysis rather than as means for proposing unique optimal decisions and thus replacing the decision maker.

Such considerations led to the construction of the decision analysis and support systems of DIDAS family - that is, Dynamic Interactive Decision Analysis and Support systems, see e.g. Lewandowski at al. (1983). DIDAS systems are addressed to analysts or teams of analysts who want to analyse their substantive models and, if the system is user-friendly, even to decision makers working alone.

The most general classification of models is into linear and nonlinear ones. Problems described by mathematical models of linear structure were investigated by many authors and several approaches to defining such problems interactively were proposed. The situation is much more complicated in the case of nonlinear models. A short review (see Lewandowski 1985) of problem interfaces in existing implementations of nonlinear DIDAS systems will underline the main points. Before the fulfillment of the present contract there existed three versions of nonlinear DIDAS: DIDAS-N developed by Grauer and Kaden (1983), a specialized system developed by Kaden and Kreglewski (1985) and general purpose DIDAS implemented by Kreglewski and others (Kreglewski at al. 1985). In the Grauer and Kaden's version, the equations describing objective and constrains functions must be programmed in FORTRAN. The authors supply a "skeleton" FORTRAN subroutine with empty" holes" where the user must locate his FORTRAN code. This is rather a complicated task - separate parts of the problem definition must be located in various places of the code, and the code itself must be written taking into account the variable names and structures used in this skeleton subroutine. Next, the user must calculate analytically the derivatives of the objective and the constraints functions. This is needed in each version of DIDAS, since practically only differentiable optimization methods are sufficiently efficient and robust to be applied in interactive decision support systems. This task is time consuming and can be a source of errors. A warning for persons who believe in their error-free analytic calculations can be found in Pavelle at al. (1981), in a slightly different context. They state that a computer verification (by means of symbolic computation) of eight widely employed tables of indefinite integrals discovered that about 10 percent of the formulae were erroneous; one of the tables was found to have an error rate of 25 percent. Errors in gradients are typically difficult to detect - they can only be detected when the behavior of an optimization process is observed by a user qualified in optimization techniques. After completing analytical calculation of derivatives, the user must properly augment his formulae with penalty function terms and their derivatives. This is conceptually rather difficult for a user who is not familiar with mathematical programming algorithms, and can also lead to numerous errors. The conflict of variable names is also probable.

The specialized system of Kaden and Kreglewski is of no interest in the present context, since its model was programmed once and the user interacts only on the level of input data and reference point selection. The general purpose version developed by Kreglewski and others (1985) also needs a FORTRAN subroutine containing the problem description, but there are less programming constrains put on the user. He must preserve only the general structure of the subroutine header (formal parameter declaration) and the COMMON block. No variable conflict can occur, and the standards according to which the body of the subroutine must be composed are straightforward. The task of analytical computation of derivatives remains but the errors can most probably be detected aposteriori by the method of Wolfe (1982).

Concluding, the following basic disadvantages of problem generation are found in the past implementations of nonlinear DIDAS:

- the user must compute analytically all derivatives of the objective and the constraint functions,
- programmed in FORTRAN according to the specification supplied by the manual of the system; this specification can be difficult to understand for non-experienced user,
- the user must be familiar with the details of the computing environment of the computer on which he is working, such as editor, compiler, linker, operating system command language etc.

- any changes of the model - the process being in fact one of the important stages of interactive work with the system - cannot be performed within the system, but must go through the chain: program editor - FORTRAN compiler - linker - operating system. This slows down the interaction process, makes it difficult and inefficient.

The new problem generator is aimed at a user, who is neither an experienced computer user nor an expert in optimization techniques. The work done can be divided into three parts:

- a proposal of a standard for not linear model formulation (thus extracting a concrete subclass from the universe of nonlinear problems: "a nonlinear problem is a problem, which is not linear"),
- imbedding of this model in an easy and user-friendly format of a spreadsheet,
- automatic computing of the necessary derivatives by symbolic differentiation procedures.

The standard for model formulation helps to classify thinking about model in the user's mind. Variables are divided into input and output.The input variables can be further subdivided into decision variables and parametric variables. The outcome variables can be further subdivided into guided outcomes (which are subject to equality constraints), floating outcomes (having only informative importance for the user), objectives i.e. optimal outcomes, and intermediate outcomes, introduced for ease of model formulation. Classification of the outcome variables can be changed by the user. The model includes lower and upper bounds for all the input and the outcome variables. The model equations are of explicit type: outcome variables are defined consecutively, depending on input variables and previously defined outcome variables (may be intermediate outcomes).

Formulae defining derivatives of outcome variables are computed automatically. However it must be stressed that it is the user's responsibility to ensure that formulae which have to be differentiated are differentiable. Differentiability cannot be checked automatically, because values of (future) input variables are unknown (e.g. the derivative of ABS (absolute value) is taken as SIGNUM, which is formally incorrect but gives good results unless argument of function crosses zero value. It is the user's responsibility to assure that the argument of ABS remains in the same open semiaxis.)

In the present version, the spreadsheet contains formulae for partial derivatives (with respect to input variables and intermediate outcomes) which can be inspected like any other formulae. Total derivatives are computed numerically, by suitable combinations of values of partial derivatives. This solution was chosen from speed considerations. However, sometimes it may be interesting to view the formulae of total derivatives. This option will be implemented in a later version of the generator. The shape of computed derivatives may differ from those computed by hand. They are mathematically correct but may not be presented in the simplest form (depending on the particular formula). In the program, there is a very intensive simplification applied "locally" to the graph of internal representation of a computed derivative. In the first version, however, there are no "global" simplifications, such as gathering of common simple expressions, terms etc. Such transformations present, as a rule, the most complicated part of any computer algebra system, are time consuming, and the most "handsome" form cannot be chosen automatically without interaction with the user. Global simplifications will be included in a later version of the generator, after gaining experience for their proper tuning (final form versus computing speed) on the experimental model.

However, this approach has a well hidden source of potential errors. Usually it is tacitly understood that formulae are entered into a spreadsheet in a "natural" form and very little attention is paid to their syntax. The danger is that what is natural to the implementer may not be natural for the user and vice versa. Habits (of expressing mathematical formulae) are formed by one's education in mathematics and one's experience with programming languages. A closer look on these sources reveals immediately that there is no consensus on the syntax (and even semantics - the results of evaluation) of mathematical formulae. Most confusion arises from the relation between the unary minus and the power operator. If the unary minus has higher precedence than power (i.e. evaluates as first) then the following identities hold true:

-2^2=4 but 4-2^2=0.

(In the first formula, the minus sign denotes the unary operator,in the second - binary). When the precedence relation is reversed, we have

-2^2=-4.

The second source of confusion is the associativity of the power operator, i.e. whether the sequence *a^b^c* evaluates as *a^(b^c)* or *(a^b)^c*. In the first case (right associativity) 2^2^3=256, while in the second (left associativity) 2^2^3=64. What is more, this associativity affects not only the process of formula evaluation but even the process of computation of derivatives. For the right-associative power operator (*e* in this example denotes the base of natural logarithms):

d(e^x^2)/dx=2xe^x^2,

see e.g. Fichtenholz (1966), vol.I, par.99; while for the left-associative operator

d(e^x^2)/dx=2e^(2x).

The moral is, one cannot be too confident when starting work with "evidently simple" piece of software.

A2. SHORT PROGRAM DESCRIPTION

The nonlinear model generator was designed as a part of a new nonlinear version of the DIDAS system, however it is available as a separate program. It is recorded on a single diskette that should be installed on an IBM-PC-XT or a compatible computer with Hercules or color graphic card, preferably with a hard disk. A diskette contains compiled code of a Pascal program and several data files with examples of nonlinear models generated by the generator. The program can be activated by the command NGENER at the DOS prompt. The nonlinear model generator supports the following general functions:

1) The declaration of the numbers of outcomes (dependent variables) and independent variables. Those numbers are used to form a spreadsheet of appropriate dimensions. Rows of the spreadsheet represent outcomes (dependent variables), whereas columns represent independent variables. Separate row and column are used for user-defined names of both kinds of variables. One dedicated row is used for the entry of user-defined values of all independent variables, and one dedicated column is reserved for program- calculated values of outcomes (they are used as inputs and outputs, respectively, in the simulation of the model).

2) The definition and edition of a nonlinear model in the form of mathematical formulae. A spreadsheet column intended for outcome values is used also as a storage place for this formulae. It is not wide enough , however, for the edition of formulae up to 255 characters long, therefore an edited formula is displayed in the bottom line of the screen and scrolled left and right, if necessary, in this line. In the spreadsheet cell three asterisks are displayed if the formula related to this cell is already defined.

3) The generation of all partial derivatives. The derivative of an outcome, say y, with respect to an independent variable, say x, is stored in the i-th row of the j-th column of the spreadsheet. There are also additional columns of the spreadsheet for the formulae of the cross-reference derivatives among outcomes. Although formulae for derivatives are generated automatically, they could be viewed and edited by the user in same way as outcome formulae.

4) The simulation of the model. The calculation of all values of outcomes and derivatives is performed. Numerical results are displayed in the spreadsheet cells intended for formulae, each three asterisks marker is replaced by an appropriate number.

The nonlinear model generator has been developed in the Institute of Automatic Control, Warsaw University of Technology, Warsaw, Poland, in a contracted study agreement "Theory, Software and Testing Examples for Decision Support Systems" with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis, Laxenburg, Austria, which has the copyright for this system.

B1 THEORETICAL MANUAL

B1.1 SYNTAX OF FORMULAE

When working with such a user-friendly tool like a spreadsheet, one can be tempted with the following thoughts: "I am so qualified in mathematics and I have so much computational experience that everything is self-evident to me". In fact there may be problems, and with such seemingly harmless things as formulae in the spreadsheet. Their syntax is as "natural" as possible, i.e. near to one's notational habits. These habits are created by mathematical education and by experience in programming languages. A closer look reveals that those sources are completely inconsistent.

Formulae are composed usually of numerical constants, cell names, parenthesis, some set of standard functions and operators denoted by the characters +, -, *, /, ^. Two of them, + and -, have nonunique meaning - they denote both the binary operators of addition and subtraction and the unary operators of plus and minus (change of sign). The operator properties are usually described by their precedence and associativity. It seems however that school mathematics establishes accepted conventions only for binary operators. Further patterns can be sought in popular programming languages and tools. Such review depicts only the total lack of commital conventions.

The most exotic rules are used in APL. All operators have the same precedence level and are right-associative. This contradicts even the "school" rules. Leaving this case out of considerations, most problems arise from the treatment of unary minus. Aho and Ullman (1977) give a dramatic warning: "Beware the treatment of unary minus!". Three different syntaxes can be considered as eligible candidates for the spreadsheet. They will be presented below using the notation of Modified Backus-Naur Form. The meaning of meta-symbols is as follows:

=	- denotes the definition,
|	- separates alternative options within the clause,
...	- terminal symbols are quoted,
(...)	- exactly one of the enclosed alternatives must be selected,
[...]	- zero or one occurrence of the enclosed subclause,

{...} - zero or any number of occurrence of the enclosed subclause.

The syntax definitions below are left unfinished. Lacking definition of factor will be presented later in two variants, any of them can be composed with previous syntaxes giving the total number of six variants.

Syntax S1

expression= ["+"] simple_expression {("+" |"-") simple_expression}
simple_expression = term {(" * " | "/") term }
term = signed_factor {"^" signed_factor }
signed_factor = ["-"] factor

Syntax S2

expression= ["+"] simple_expression {("+" |"-") simple_expression}
simple_expression = signed_term {(" * " | "/")signed_term }
signed_term = ["-"] term
term = factor {"^" factor }

Syntax S3

expression=["+"|"-"]simple_expression {("+"|"-")simple_expression}
simple_expression = term {(" * " |"/") term }
term = factor { "^" factor }

The S1 syntax is used e.g. in ALGOL 68, SNOBOL, MICROCALC (a demonstration spreadsheet program supplied by Borland together with TURBO-PASCAL). From the definition, the following operator precedences can be read:

-(unary) > ^ > * ,/ > +,-(binary).

The S2 syntax is used e.g. in FORTRAN, BASIC, PL/I, Lotus 1-2-3 by Lotus (1983), MUSIMP (the implementation language for MUMATH symbolic computation system by Microsoft(1983)). It has the following precedences:

^ > -(unary) > * ,/ > ,-(binary).

The S3 syntax is that of PASCAL extended with the power operator. Its precedences are:

^ > * ,/ > +,-(unary and binary).

Each syntax has its peculiarities, e.g.

in S1: -2^2=4, 4-2^2=0;

in S2,S3: -2^2=-4.

The syntax rules must be supplemented by associativity rules. In our context they are essential only for the power operator. The problem is whether it is right-associative, i.e. *a^b^c* evaluates as *a^(b^c)*, or left-associative: *(a^b)^c*. Both cases lead to completely different values, e.g. 2^(2^3)=256 while (2^2)^3=64. Thus the semantics of a formula depends on the chosen rule. Associativity affects not only the process of formula evaluation, but even the process of computation of derivatives. For right-associative power operator

d(e^x^2)/dx=2xe^x^2,

(e denotes here the base of natural logarithms), while for left- associative

d(e^x^2)/dx=2e^(2x).

It seems that courses in differential calculus do not explicitly declare associativity but tacitly use the first variant, see e.g.examples in the classical Russian textbook Fichtenholz (1966), vol. I, par. 99. MUMATH assumes right associativity (under normal setting, properties of all operators can be freely modified by the user). Similarly does MACSYMA, perhaps the world's largest computer algebra system, see an differentiation example in Pavelle (1985).

Associativity rules used in programming languages are nonunique. Aho and Ullman(1977) state on p. 47: "ALGOL evaluates all binary operators left-associatively. FORTRAN lets the compiler designer choose the associativity, and PL/I evaluates all binary operators left-associatively, except for exponentiation, which is right-associative". In implementations of BASIC used by the author the power operator was left-associative, and that of FORTRAN - right-associative. In hand calculators it is left associative, perhaps due to hardware and software limitations.

In this work, the following rules were accepted for associativity of binary operators:

- the power operator is right-associative,
- all other operators are left-associative.

The lacking part of syntax definition can have two forms:

Syntax S4

```
factor = constant | variable | "(" expression ")" |
        standard_function factor.
```

Syntax S5

```
factor = constant | variable | "(" expression ")" |
        standard_function "(" expression ")".
```

The difference between S4 and S5 is in the allowed forms of arguments of standard functions. Syntax S5 is conservative; each argument must be put into parenthesis, e.g.
sin(x), arctan(25.6), exp(sin(log(y))).

Syntax S4 is more flexible, it allows for simplified forms for "simple" arguments but includes also the forms from S5. Thus in S4 the following formulas are equivalent:
sin x, sin(x)
cos 55, cos(55)
exp sin log y, exp(sin(log(y))).

By clever implementation of the lexical analyzer, even the delimiting spaces may be unnecessary, thus
sinx, cos55, expsinlogy can be made acceptable too.

Arguments which are not factors must of course be put into parenthesis e.g.
*sin(a+b), log(2 * x)*.

A search for inspiration gives as usual no results. Handbooks of mathematics do not use parenthesis, majority of computer languages do use but some of them do not. Some BASIC dialects even differ between themselves in this point.

Of course any syntax convention will do, as long as formula evaluation and differentiation are consistent and the user is aware of its properties. The present implementation uses the syntax S1+S5.

B1.2 SYMBOLIC DIFFERENTIATION OF FORMULAE.

PARSING

Programmers recognized very early that computers can work not only with numbers themselves but also with more abstract symbols. Computer programs for symbolic manipulation have been appearing since late fifties. However the popularity of symbolic computing is orders of magnitude less then that of numeric computing. Many people are even not aware of their existence. There are numerous reasons of this situation. Restricted availability - at first at some university centers and only some types of computers and/or operating systems, high cost - large computers, high demand of operating memory and usually long (and unpredictable) computation time. The situation has changed at first with the introduction of computer networks and then with the era of microcomputers. However, a professional system for symbolic computation needs a powerful workstation with the operating memory of some mega bytes. In the IBM-PC class there is a MUMATH system by Microsoft. The use of it in this work was excluded for two reasons:

- licence problem,
- interface with the rest of program would be possible only at the operating system level via exchange of files.

It is worth noticing that besides systems which produce the formula of a derivative there are systems which give the value of a derivative. Examples of such systems can be found e.g. in series of papers by Rall; the book by Rall (1981) is the good and most detailed representative.

A program for any symbolic computations resembles strongly the compiler of a programming language. Therefore some knowledge of elements of compiler-writing problems would be of great help in understanding the following. An excellent source of information is Wirth (1976), Chap. 5.

It is customary to partition the compilation process into a series of subprocesses called phases. At first the input language must be specified. In our case it is very simple, it consists only of formulae and is defined by means of syntax MBNF clauses or syntax diagrams. The first phase of compilation, called the lexical analysis, separates characters of the source language (the formula) into groups that logically belong together. They are called symbols or tokens, they are e.g. identifiers of variables, identifiers of standard functions, symbols of operators and punctuation symbols. The output of the lexical analyzer is a stream of symbols, which is passed to the next phase, to the syntax analyzer or parser.

The parser checks whether the symbols appearing at its input form a legal sequence of the input language (defined by its syntax rules). Besides, it does some other functions. In the model generator there are three parsers differing by these functions. During edition of a formula parser detects errors (if any) and interacts with the user on their correction. During "compilation" of a formula it produces code for a stack machine. During differentiation it produces a binary tree - an internal representation of a formula.

Many parsing methods have been developed and applied in practice. In this work a recursive-descent parsing was implemented (top-down parsing without backtracking). This method is easy to implement by hand and enables to express the generation of output directly in terms of the syntactic structure of the source language. Presentation of syntax in the form of a diagram gives immediately the block scheme of the parser. Each occurrence of a terminal symbol corresponds to the instruction, which recognizes this symbol and reads the next symbol from the input. Each occurrence of a nonterminal symbol corresponds to the call of a procedure, whose structure is given by its own diagram.

B1.3 INTERNAL REPRESENTATION OF FORMULAE

The majority of languages, in which systems for symbolic computation are implemented, are of list-processing type, i.e. a list is their primal informational structure. An external representation of a list can be e.g. (A,B,C,D,E), while the internal representation consists of chained nodes. Each node contains some information (in the above example - a letter character) or the pointer (an address) to "own" information, and the pointer to the next element of the list. There is a pointer constant, usually called NIL, which means "pointer to nowhere" and is used to denote the end of a list. The simplest "units" of information stored in a node (or pointed to from a node) are called atoms. Usually they are numbers, identifiers and pointers. The simplest and most commonly used list representation of formulae corresponds to their prefix form, e.g. $x+y$ becomes $(+,x,y)$. Each element of a list can be a list themselves, e.g. the expression $x+2*y+3$ can correspond to the list $(+,x,(*,2,y),3)$.

Thus a tree of large complexity can be created. In PASCAL nodes will be represented by records and connections between them by pointers. There are no stiff rules for choosing a particular representation of a node; it depends on habits, experience and preferences of the implementer. Many decisions must be taken in connection with the usual trade-off between the range of needed memory and the speed of execution. E.g. some redundant information can be stored in nodes thus being immediately available (but more memory is used for data). In the other variant it can be extracted from the tree structures when needed - thus more memory is used for program code for additional procedures, and the execution is slower.

In the presented implementation, there are two pointers in each node. It means that the structure formed with the use of them is a binary tree. The usual form of visualization of such structure is with the root being most upper and leaves dangling down. In the present context it is, however, more convenient to imagine, that the binary tree is turned counter-clockwise so that the right branches are horizontal and the left - vertical. To limit the number of procedures needed for operations on structures, the internal representation of structures does not use the binary minus and the division. Instead the addition and multiplication is used with the right arguments multiplied by -1 or taken to power -1, respectively.

The above-mentioned list representation is used in MUMATH symbolic programming system. Its code is known and some conclusions about the efficiency of this representation can be deduced, although the code has been written in a specialized MUSIMP programming language and therefore any comparison with a PASCAL implementation must be rather rough. The main advantage of this representation is the economy of memory. The price is the relative complexity of arithmetic operations, since a list contains only very condensed syntactic information. For example, the process of adding two formulae represented by lists $(+,x,y)$ and $(*,2,y)$ must at first reconstruct their syntactic relations. Arithmetic and algebraic packages contain about 100 functions (not to mention the "primitive" MUSIMP functions) and from the programming point of view are the most complicated part of the whole system. Functions are highly recursive and the number of their calls during evaluation the expression is rather high. An experiment with tracing (of main algebraic functions only) was performed for two simple expressions $e1=x+y$ and $e2=2*x-y$. In the process of their addition there were 18 function calls, in the process of subtraction - 62 calls. The solution is computed (and simplified) recursively e.g. by adding successive simple expressions. It means that simplification is repeated many times, but when the final result is obtained it is already in a simplified form.

In the presented implementation quite reverse principles were chosen, basing on the following considerations. At a time there exist only two structures, that of a differentiated formula and that of a derivative. After the derivative is compressed into the formula form, the memory used for representation of structures is released. Thus the data storage presents no special limitation. On the other hand, the Turbo PASCAL limits the amount of program code to 64 Kbytes (and overlays decrease speed). PASCAL is rather talkative for such type of problems, program grows large - so it was thought desirable to limit the number of procedures. The chosen structure expresses the syntactic properties of a formula in an explicit form, at the cost of larger number of nodes. In MUMATH a n-argument sum, product or power expression is represented by a $(n+1)$-element list. In the presented implementation the number of nodes is equal to $3n$, $2n+1$ and $n+2$ respectively (the more complicated is the formula - the better is the comparison). As the result, there are only about 20 "algebraic" procedures. Once the formulae are converted into internal form, their addition is achieved in 1 procedure call. Compression of structures into formulae is divorced from the arithmetic and differentiating operations, i.e. analytical operations are performed on structures and only the final result is transformed into a formula form (and simplified). To improve speed further, the iterative methods are used whenever possible.

Below, there are some examples of internal representations of simple formulae. The rules of visualization are as follows. To each node, there corresponds one row of text; "lower" nodes are recursively intended. Nodes from the same "level" have the same indentation. Characters '+', ' * ', '^' denote consecutive levels of the graph. Simple-expressions are linked together via nodes at the '+' level, terms - at the ' * ' level, signed-factors - at the '^' level.

```
expression 1:  a             expression 2:  x+y
structure:                  structure:
       +                          +
         *                          *
           ^  a                       ^ x
                                 +
                                   *
                                     ^ y
```

Structures corresponding to formulae without parentheses are three nodes "deep". The highest level, denoted by '+' sign, corresponds to simple-expressions. Addition of two formulae corresponds to linking of two structures at this level. In the expression 2 there are two simple-expressions, each consisting of a simple term, each of them being a single signed-factor, each being a factor (without sign).

```
expression 3: x * y          expression 4:  x^3
structure:                  structure:
    +                           +
      *                           *
        ^  x                        ^  x
      *                               ^  3
        ^  y
```

The middle level, denoted by ' * ' sign, corresponds to terms. Linking of structures at that level corresponds to multiplication. The low level, denoted by '^' sign, corresponds to signed-factors. Constants and names of variables are contained only in nodes of the ^ level, nodes from levels + and * are used only for structuring. The expression 3 consists of one simple-expression, which has two terms. The expression 4 has one term consisting of two signed-factors, each of them being a factor.

```
expression 5: -2 * x+y^z * (a-3.5)   expression 6: sin(x)
    structure:                          structure:
      +                                   +
        *                                   *
         ^ 2                                 ^ sin
        *                                        +
         ^ x                                       *
        *                                           ^ x
         ^ -1
     +
       *
        ^ y
        ^ z
       *
        ^
          +
            *
             ^ a
          +
            *
             ^ 3.5
            *
             ^ -1
```

In the example 5 there are two simple-expressions. The first one consists of 3 terms. Each of these terms is a single factor. The second simple-expression has two terms. The first one consists of two factors, the second has one factor, which is an expression in parentheses. Each pair of nested parentheses increases the depth of a structure by three nodes. Standard functions are just another form of factors. Their names appear in nodes of the ^ level. Arguments in parentheses are expressed by a lower "layer" of structure, three nodes deep.

```
expression 7: sin x     expression 8: sin ln cos x
  structure:              structure:
  +                       +
    *                       *
     ^ sin x                 ^ sin
                             ^ ln
                              ^ cos x
```

In the S4 syntax, standard functions have factors as arguments. When this factor is a variable or a constant, the whole "compound" factor corresponds to one node (expression 6). An interesting situation occurs, when the argument is an standard function (see expression 8). The up to here recursive structure of the graph is perturbed.

B 1.4 ARITHMETIC AND DIFFERENTIATION OPERATIONS

All mathematical operations are performed on structures - on copies of their arguments. The addition is the simplest, the corresponding structures are linked together at the + level (see the structure of expression 2). In the multiplication procedure a skeleton structure is produced as in expression 3, but without the variables x, y in ^ nodes. The copies of arguments are linked under those nodes, instead. The power procedure acts similarly using the structure presented for expression 4. Subtraction and division appear only in the "external" formula representation and never - in the "internal" structural form, so they need not be implemented. Operations of standard functions are performed according to patterns presented for expressions 6,7 and 8.

Differentiating procedures take structures as their input and produce structures representing derivatives, basing on rules of differentiation and using the library of mathematical operations. The derivative of an expression is formed as the sum of derivatives of simple- expressions. A simple-expression has the form of *t * t * .. * t* , where t denote a term. Its derivative if formed as the sum of all products of the form *t * .. * dt /dx * .. * t* . Differentiation of these both syntactic constructs can (and is) be implemented iteratively. A term has the general form *f ^f ^..^f* , where *f* denote signed-factors. The derivative of a term is computed according to the formula

*d(f ^R)/dx=f ^R * ln f * dR/dx+R * f ^(R-1) * df /dx,*

where *f* denote the first factor and *R* - the rest of a term. It is a recursive definition and must be implemented recursively. The derivative is produced by creating all necessary factors and linking them into two simple-expressions and finally into one expression. The derivative of a signed-factor (or a factor) is computing according to its form. A derivative of a constant or a variable is the structure corresponding to the constant 0 or 1. When a factor has the form of an expression enclosed in parentheses, its derivative is computed by the recursive call of the expression differentiating procedure. When a factor is a standard function, its derivative is computed by creating a structure corresponding to the derivative of this function and multiplying it by the derivative of its argument.

B 1.5 COMPRESSION OF STRUCTURES

SIMPLIFICATION

Differentiating procedures give the result in the form of a structure. It must be compressed to the form of an expression. This structure is highly redundant, e.g. the derivative of *2 * x* is computed as *0 * x+2 * 1*, so the compression must be accompanied by the simplification.

Implementation of any simplification procedure presents a real problem. First, there exists no universal "simplest" form of an expression, e.g. for *e1=(1+a) -3 * a -a* the good result would be obtained after expanding powers, while for *e2=(1+a)* just the reverse. Large symbolic systems try several possibilities and automatically choose the most concise form. Small systems, like MUMATH, apply automatically only a reduced set of simplifying actions. The user can interactively invoke other, more complicated but also more time consuming actions, but it requires both some experience and patience. Both approaches rather cannot be used in the presented program. Second, it is rather a complicated programming task because of large number of interactions between operators, variables and special values (such as 0 and 1) used subconsciously by anyone trained in mathematics - all of them must be modelled in the program in order to produce the output in the expected form.

There are two possible approaches to simplification of formulae. The first - by transforming a structure into another, simpler structure and the second - by transforming a structure into a simplified formula. Both methods are used in the program. Procedures of the first method recognize some characteristic patterns of the argument structure and create another structure by copying suitable parts of the arguments into proper places of the result structure. The second method gives (partial) simplification with simultaneous compression of a structure into a formula. It is implemented as a recursive procedure. The structure is searched in post-order. For a given node there is a formula or a value corresponding to the (part of original) structure pointed to by the "low" pointer of this node and a formula or a value corresponding to the structure pointed to by the "right" pointer. These two arguments are combined together into a formula or a value, according to the "contents" of the node, i.e. they are added, multiplied, exponentiated, the sign is changed or a standard function is applied. The simplification may be partial only, because it has "local" character (the number of intermediate expansions of a formula into a structure and subsequent compressions is kept to minimum).

The kind of necessary simplifying actions and their order depend on the particular expression. There are some typical transformations, which are often used:

- distribution of the base of an expression over the simple expressions of an exponent, which is a sum;
- distribution of the exponent of an expression over the base, which is a product;
- different kinds of distributions connected with the numerator and the denumerator of an expression;
- expansion of powers of sums;
- factoring of the base from the simple expressions of an exponent, which is a sum;
- factoring of the exponent from the base, which is a product;
- factorials connected with the numerator and the denumerator.

For expressions of modest complexity, usually only one or two of them gives improvement, while others only increase processing time without producing essentially simpler result. Therefore a compromise should be sought between computing speed and "simplification power", based on the expected average "complexity level" of differentiated formulae.

B 1.6 EVALUATION OF FORMULAE

Two variants of the formulae evaluation were implemented. In the first one a specialized parser was used for the analysis of a formulae and accumulating its value. The speed of this method proved too slow for applications in DIDAS, although it was satisfactory for the problem generator needs. In the second method there is a procedure implementing a virtual computer. Formulae are compiled into the code for that computer. This code is stored in the spreadsheet and is used for computing of formulae value.

REFERENCES

Aho A.V. and Ullman .D.(1977). Principles of Compiler Design. Addison Wesley.

Fichtenholz G.M.(1966). Handbook of Differential and Integral Calculus. Nauka.(in Russian).

Kaden S. and Grauer M.(1984). A Nonlinear Dynamic Interactive Decision Analysis and Support System (DIDAS-N). User's Guide. WP-84-23. IIASA. Kreglewski T. and Kaden S.(1985). Decision Support System MINE. Problem Solver for Nonlinear Multicriteria Analysis. IIASA.

Lewandowski A., Kreglewski T. and Rogowski T.(1985). DIDAS-NL -the Nonlinear Version of DIDAS System. In: Theory,Software and Test Examples for Decision Support Systems. A.Lewandowski and A.Wierzbicki,Eds. IIASA.

Lewandowski A.(1985). Problem Interface for Nonlinear DIDAS. Draft. Pavelle R.(1985). MACSYMA: Capabilities and Applications to Problems in Engineering and the Science. In: EUROCAL'85, European Conference on Computer Algebra. B.Buchberger,Ed. Lecture Notes in Computer Science, Vol.203.Springer.

Pavelle R.,Rothstein M. and Fifch J.(1981). Computer Algebra. Scientific American, December,1981.

Rall L.B.(1981). Automatic Differentiation:Techniques and Applications. Lecture Notes in Computer Science, Vol.120.Springer.

Wirth N.(1976). Algorithms + Data Structures = Programs. Prentice Hall.1976.

Wolfe P.(1982). Checking the Calculation of Gradients. ACM Trans. Math. Software. Vol.8, December, 1982.

IAC-DIDAS-N A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models on Professional Microcomputers

T. Kreglewski, J.Paczynski, A.P. Wierzbicki

Institute of Automatic Control, Warsaw University of Technology

ABSTRACT

This paper presents introductory documentation and theoretical manual for a version of decision analysis and support systems of DIDAS family that is designed for work with nonlinear models on professional microcomputers. This version has been developed in 1986 in the Institute of Automatic Control, Warsaw University of Technology, under a contracted study agreement with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis and differs from previous nonlinear DIDAS versions in several aspects. It can be run on professional microcomputers compatible with IBM-PC-XT (with Hercules or color graphics card and, preferably, with a numeric co-processor and a hard disk) and supports graphical representation of results in interactive analysis. Moreover, this version called IAC-DIDAS-N is equipped with a new nonlinear model generator and editor that supports, in an easy standard of a spreadsheet, the definition, edition and symbolic differentiation of nonlinear substantive models for the multiobjective decision analysis. A specially introduced standard of defining nonlinear programming models for multiobjective optimization helps to connect the model generator with other parts of the system. The optimization runs helping the interactive, multiobjective decision analysis are performed with the help of a new solver, that is, a new version of nonlinear programming algorithm especially adapted for multiobjective problems. This algorithm is based on shifted penalty functions and projected conjugate directions techniques similarly as in former nonlinear versions of DIDAS, but it is further developed and written in PASCAL together with other algorithms and programs of IAC-DIDAS-N system.

A. INTRODUCTORY DOCUMENTATION

A1. EXECUTIVE SUMMARY

In many situations of complex decisions involving economic, environmental and technological decisions as well as in the cases of complex engineering design, the decision maker needs help of an analyst or a team of them to learn about possible decision options and their predicted results. The team of analysts frequently summarizes its knowledge in the form of a *substantive model* of the decision problem that can be formalized mathematically and computerized. While such a model can never be perfect and cannot encompass all aspects of the problem, it is often a great help to the decision maker in the process of learning about novel aspects of the decision situation and gaining expertise in handling problems of a given class. Even if the final decisions are typically made judgementally - that is, are based on holistic, deliberative assessments of all available information without performing a calculative analysis of this information, see S.Dreyfus (1984) - the interaction of a decision maker with the team of analysts and substantive models prepared by them can be of great value when preparing such decisions.

In organizing such interaction, many techniques of optimization, multicriteria decision analysis and other tools of mathematical programming can be used. To be of value for a holistically thinking decision maker, however, all such techniques must be used as supporting tools of interactive analysis rather than as means for proposing unique optimal decisions and thus replacing the decision maker. The decision analysis and support systems of DIDAS family - that is, Dynamic Interactive Decision Analysis and Support systems, see e.g. Lewandowski et al. (1983) - are especially designed to support interactive work with a substantive model while using multicriteria optimization tools, but they stress the learning aspects of such work, such as the right of a decision maker to change his priorities and preferences when learning new facts. DIDAS systems can be used either by analysts who want to analyse their substantive models, or by teams of analysts and decision makers, or even by decision makers working alone with a previously defined substantive model; in any case, we shall speak further about *the user* of the system.

There are several classes of substantive models that all require special technical means of support. The IAC-DIDAS-N version is designed to support models of multiobjective nonlinear programming type. While some nonlinear DIDAS versions have been developed before, they did not follow any standards of defining such models, since such standards did not exist. In order to support the work with a user that is not necessarily a specialist in computer programming and nonlinear optimization programming, it has become necessary to introduce such standards.

Models of multiobjective nonlinear programming type specify, firstly, the following classes of variables: input variables that can be subdivided into decision variables (that is, means of multiobjective optimization) and *parametric variables* (that is, model parameters that are kept constant during multiobjective analysis but might be changed during parametric or sensitivity analysis) - and *outcome variables* that can be subdivided into *guided outcomes* (that should be kept constant, subject to equality constraints), *floating outcomes* (either used only for the easiness of definition of the nonlinear model or having only informative importance for the user) and *optimized outcomes* or *objectives* (the ends of multiobjective optimization that can be either maximized or minimized or stabilized, that is, kept close to a desired level). Actually, the distinction between various types of outcome variables is not necessarily sharp and the user might change their classification and select his objectives among various outcome variables when defining the multiobjective analysis problem.

For all input and outcome variables, a reasonably defined nonlinear model should include *lower* and *upper bounds*, that is, reasonable ranges of admissible changes of these variables. Moreover, an essential part of a nonlinear model definition are *model equations*, that is, nonlinear functions that define the dependence of all outcome variables on input variables. To make the model definition easier for the user, it is assumed that outcome variables are defined consecutively and that they can depend not only on input variables, but also on previously defined outcome variables. However, all outcome variables must be defined explicitly (an implicit definition of an outcome variable is also possible, but is performed indirectly - see theoretical manual).

There are many examples of decision problems that can be analysed while using a substantive model of multiobjective nonlinear programming type; for example, DIDAS-type systems with multiobjective nonlinear programming models have been used in analysing various environmental or technological problems (see Kaden, 1985, Grauer et al., 1983). As a demonstrative or tutorial example, IAC-DIDAS-N uses a multiobjective nonlinear programming model for a control system design, where the decision variables are the parameters of a controller and the objectives are various performance indices of the control system, either in time-domain or in frequency domain. The user can also define substantive models of multiobjective nonlinear programming type for his own problems and analyse them with the help of IAC-DIDAS-N.

A typical procedure of working with a DIDAS-type system consists of several phases.

In the first phase, a user - typically, an analyst - defines the substantive model and edits it on the computer. In earlier versions of nonlinear DIDAS-type systems (which were mostly implemented on bigger mainframe computers) this phase has not been explicitly supported in the system and the user had to separately prepare (define and edit) his nonlinear model, typically in the form of a FORTRAN procedure that contained also user-supplied formulae for the derivatives of all outcome functions with respect to decision variables. It is a known fact that most mistakes in applying nonlinear programming methods are made when determining analytically derivatives; thus, this form of preparation of a substantive model required rather much experience in applications of nonlinear programming.

The new features of IAC-DIDAS-N are, firstly, the definition and edition of substantive models in an easy but flexible standard format of a spreadsheet, where the input variables correspond to spreadsheet columns and the outcome variables - to spreadsheet rows; special cells are reserved for types of variables, scaling units, lower and upper bounds for all variables, constraining values for guided outcomes as well as aspiration or reference levels for stabilized, maximized and minimized outcomes, for results of various optimization computations, etc. However, another unique new feature of IAC-DIDAS-N is an automatic support of calculations of all needed derivatives by a symbolic differentiation program. The user does not need to laboriously calculate many derivatives and to check whether he did not make any mistakes; he must only define model equations or outcome functions (possibly in recursive, but explicit form) and make sure that these functions are differentiable and admissible for the symbolic differentiation program - which admits functions from a rather wide class. The spreadsheet format currently implemented does limit somehow the size of substantive models that can be defined in it (in the particular implementation of IAC-DIDAS-N, to 26 decision variables, 26 parameters and 26 outcome variables) but reasonable models of nonlinear programming type that can be usefully analysed on professional microcomputers should not be too large anyway; on the other hand, the spreadsheet format allows also for display of automatically determined formulae for derivatives or their computed values in appropriate cells. The user of IAC-DIDAS-N can also have several substantive models recorded in a special model directory, use old models from this directory to speed up the definition of a new model, etc., while the system supports automatically the recording of all new or modified models in the directory.

In the second phase of work with DIDAS-type systems, the user - here typically an analyst working together with the decision maker - specifies a multiobjective analysis problem related to his substantive model and participates in an initial analysis of this problem. There might be many multiobjective analysis problems related to the same substantive model: the specification of a multiobjective problem consists in designating guided outcomes (constraints) and optimized outcomes (objectives) between outcome variables, defining whether an objective should be minimized, or maximized, or stabilized - kept close to a given level. Moreover, the user can also shift bounds on any variable when specifying a multiobjective analysis problem. For a given definition of the multiobjective analysis problem, the decision and outcomes in the model are subdivided into two categories: those that are ***efficient*** with respect to the multiobjective problem (that is, such that no objective can be improved without deteriorating some other objective) and those that are inefficient. It is assumed that the user is interested only in efficient decisions and outcomes (this assumption is reasonable provided he has listed all objectives of his concern; if he has not, or if some objectives of his concern are not represented in the model, he can still modify the sense of efficiency by adding new objectives, or by requiring some objectives to be kept close to given levels, or by returning to the model definition phase and modifying the model).

One of main functions of a DIDAS-type systems is to compute efficient decisions and outcomes - following interactively various instructions of the user - and to present them for analysis. This is done by solving a special parametric nonlinear programming problem resulting from the specification of the multiobjective analysis problem; for this purpose, IAC-DIDAS-N contains a specialized nonlinear programming algorithm called ***solver***. Following the experiences with previous versions of nonlinear DIDAS systems, a robust nonlinear programming algorithm based on shifted penalty functions and projected conjugate directions techniques was further developed for IAC-DIDAS-N; this specialized algorithm is written in PASCAL, similarly as all other parts of the system.

However, a multiobjective problem definition admits usually many efficient decisions and outcomes; the user should first learn about ***bounds*** on ***efficient outcomes***. This is the main function of IAC-DIDAS-N in the initial analysis phase. The user can request the system to optimize any objective separately; however, there are also two special commands in the system related to this function. First, called "utopia", results in subsequent computations of the best possible outcomes for all objectives treated separately (such outcomes are practically never attainable jointly, hence the name "utopia" for the point in outcome space composed of such outcomes). Second, called "nadir", results in an estimation of the worst possible between efficient outcomes (defining precisely the worst possible between efficient outcomes is a very difficult computational task; in some simple cases, the "utopia" computations give enough information to determine the worst possible between efficient outcomes, but for more general cases this information is not reliable and a more reliable way of estimating the worst possible between efficient outcomes is implemented in IAC-DIDAS-N). The "utopia" and "nadir" computations give important information to the user about reasonable ranges of decision outcomes; in order to give him also information about a reasonable compromise efficient solution, a ***neutral efficient solution*** can be also computed in the initial analysis phase following a special command. The neutral solution is an efficient solution situated "in the middle" of the range of efficient outcomes, while the precise meaning of being "in the middle" is defined by the distances between the utopia and the nadir point. After analysing the utopia point, the nadir point and a neutral solution (which all can be represented graphically for the user), the initial analysis is completed and the user has already learned much about ranges of attainable efficient objectives and the possible tradeoffs between these objectives. Each change of the definition of the substantive model or of the multiobjective analysis problem, however, necessitates actually a repetition of the initial analysis phase; on the other hand, the user can omit this repetition if he judges that the changes in the model or multiobjective analysis definition have been small.

The third phase of work with DIDAS-type systems consists in interactive scanning of efficient outcomes and decisions, guided by the user through specifying ***aspiration levels*** for each objective, called also ***reference points***. The user has already reasonable knowledge about the range of possible outcomes and thus he can specify aspiration levels that he would like to attain. IAC-DIDAS-N utilizes the aspiration levels as a parameter in a special achievement function coded in the system, uses its solver to compute the solution of a nonlinear programming problem equivalent to maximizing this achievement function, and responds to the user with an attainable, efficient solution and outcomes that strictly correspond to the user-specified aspirations.

If the aspirations are "too high" (better than attainable), then the response of the system is a solution with attainable, efficient outcomes that are uniformly as close to the aspirations as possible. If the aspirations are "too low" (if they correspond to attainable but inefficient outcomes that can be improved), then the response of the system is a solution with outcomes that are uniformly better than the aspirations. The precise meaning of the uniform approximation or improvement depends on ***scaling units*** for each objective that can be either specified by the user or defined automatically in the system as the differences between the utopia point and the current aspiration point. This second, automatic definition of scaling units has many advantages to the user who is not only relieved of specifying scaling units but also has a better control of the selection of efficient outcomes by changing aspiration levels in such a case.

After scanning several representative efficient solutions and outcomes controlled by changing aspirations, the user learns typically enough either to select subjectively an actual decision (which needs not to correspond to the decisions proposed in the system, since even the best substantive model might differ from real decision situation) or to select an efficient decision and outcome proposed in the system as a basis for actual decisions. Rarely, the user might be still uncertain what decision to choose; for this case, several additional options can be included in a system of DIDAS type. Such options include two more sophisticated scanning options: a ***multidimensional scanning***, resulting from perturbing current aspiration levels along each coordinate of objective space, and a directional scanning, resulting from perturbing current aspiration levels along a direction specified by the user (see Korhonen, 1985). Another option is forced convergence, that is, such changes of aspiration levels along subsequent directions specified by the user that the corresponding efficient decisions and outcomes converge to a final point that might represent the best solution for the preferences of the user. However, these additional options are not implemented in IAC-DIDAS-N, since the experience of working with DIDAS-type systems shows that these options are rarely used.

A2. SHORT PROGRAM DESCRIPTION

The IAC-DIDAS-N system (Institute of Automatic Control, Dynamic Interactive Decision Analysis and Support, Nonlinear version) is decision support system designed to help in the analysis of decision situations where a mathematical model of substantive aspects of the situation can be formulated in the form of a multiobjective nonlinear programming problem, possibly of dynamic structure.

The IAC-DIDAS-N system is recorded on a single diskette that should be installed on an IBM-PC-XT or a compatible computer with Hercules or a color graphic card and, preferably, with a numeric coprocessor and a hard disk (if a numeric coprocessor is present then special version of the IAC-DIDAS-N system can be used taking advantage of the coprocessor computational capacity). A diskette contains compiled code of a PASCAL program and several data files supporting windows and graphics of the system. After installing it in the user directory, it can be activated (by the command **DIDASN** at the DOS prompt). System supports the following general functions:

1) The definition and edition of a substantive model of the decision situation in a user-friendly format of a spreadsheet.
2) The specification of a multiobjective decision analysis problem related to the substantive model. This is performed by specific features of spreadsheet edition.
3) The initial multiobjective analysis of the problem, resulting in estimating bounds on efficient outcomes of decisions and in learning about some extreme and some neutral decisions. These functions are supported by some specific commands and the results are presented to the user in the spreadsheet form.
4) The interactive analysis of the problem with the stress on learning by the user of possible efficient decisions and outcomes, organized through system's response to user-specified aspiration levels or reference points for objective outcomes. The IAC-DIDAS-N system responds with efficient solutions and objective outcomes obtained through the maximization of an achievement function that is parameterized by the user-specified reference points and, optionally, user-specified scaling coefficients. The maximization is performed through a nonlinear programming algorithm called *solver*, written entirely in PASCAL. The interactive analysis is supported by entering user data into specific cells in the spreadsheet and executing commands from the current menu.

The main menu of IAC-DIDAS-N performs various functions used in several phases of the interactive analysis process. Most of the functions of phase 1) and 2) are specific commands in the spreadsheet edition (the decision variables are defined as columns of the spreadsheet, the outcome variables are defined as rows, outcome formulae are entered in the corresponding cells, there are special rows and columns for scale units, lower and upper bounds, for defining user names of objective outcomes and their types, reference points, utopia and nadir points, for solutions corresponding to the reference points). The functions of other phases are executed by macrocommands using various function keys; the user can get various help displays that suggest in an easy fashion the commands useful in a current phase of work with the system.

All commands from the main menu are invoked with function keys. Main menu commands are as follows:

F3 *Select a model*: this command enters another menu called directory menu. Commands from this menu allow changing a drive and a path where models and their data are stored, displaying contents of selected directory, renaming and deleting files.

F4 *Create new model*: this command enters the spreadsheet for a definition of a new model. Old, previously defined models could by used in several ways as templates for a new one for farther editing.

F5 *Edit a model*: this command enters the spreadsheet for editing an existing model.

F6 *Multiobjective analysis*: this command enters another part of the spreadsheet where data related to multiobjective analysis are stored. Menu of commands available here contains commands for phases 2) and 3)

F9 *Graphical representation*: Data from the spreadsheet together with some data from database are presented in several graphical forms selected by the user from a displayed menu.

F10 *Quit*: this command conditionally ends the work with the system, it is checked whether the current model and its data were saved on disk, if no the command must be confirmed by the user.

IAC-DIDAS-N system has been developed in the Institute of Automatic Control, Warsaw University of Technology, Warsaw, Poland, in a contracted study agreement "Theory, Software and Testing Examples for Decision Support Systems" with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis, Laxenburg, Austria, which has the copyright for this system.

B1. THEORETICAL MANUAL.

The standard form of a multiobjective nonlinear programming problem is defined as follows:

$$maximize\ [q=f(x)]; \tag{1}$$
$$X=\{x\in R^n:\ g'(x)=0,\ g''(x)\leq 0\}$$

where $x\in R^n, q\in R^p$, $f{:}R^n\rightarrow R^p$ is a given function (assumed to be differentiable), $g'{:}R^n\rightarrow R^{m'}$ and $g''{:}R^n\rightarrow R^{m''}$ are also given functions (of the same class as f) and the maximization of the vector q of p objectives is understood in the Pareto sense: $\hat{x}$, $\hat{q}$ are solutions of (1) iff $\hat{q}=\hat{f}(x)$, $\hat{x}\in X$ and there are no such x, q with $q=f(x)$, $x\in X$ that $q\geq\hat{q}$, $q\neq\hat{q}$. Such solutions $\hat{x}$, $\hat{q}$ of (1) are called, respectively, an *efficient decision* $\hat{x}$ and the corresponding *efficient outcome* $\hat{q}$. If, in this definition, it were only required that there would be no such x,q with $q=f(x)$, $x\in X$ that $q>\hat{q}$, then the solutions $\hat{x},\hat{q}$ would be called *weakly efficient*. Equivalently, if the set of all attainable outcome is denoted by

$$Q=\{q\in R^p{:}q=f(x),\ x\in X\} \tag{2}$$

and so called *positive cones* $\hat{D}=R^p_+, \tilde{D}=R^p_+\setminus\{\phi\}, \tilde{\tilde{D}}=intR^p_+$ are introduced (thus, $q\geq\hat{q}$ can be written as $q-\hat{q}\in D$, $q\geq\hat{q}$, $q\neq\hat{q}$ as $q-\hat{q}\in\tilde{D}$ and $q>\hat{q}$ as $q-\hat{q}\in D$), then the sets of efficient outcomes $\hat{Q}$ and of weakly efficient outcomes $\hat{Q}^w$ can be written as:

$$\hat{Q}=\{q\in Q:\ (\hat{q}+\tilde{D})\cap Q=\phi\} \tag{3}$$
$$\hat{Q}^w=\{q\in Q:\ (\hat{q}+\tilde{\tilde{D}})\cap Q=\phi\} \tag{4}$$

The set of weakly efficient outcomes is larger and contains the set of efficient outcomes; in many practical applications, however, the set of weakly efficient outcomes is decisively too large. Some efficient outcomes for multiobjective nonlinear programming problems might have unbounded *trade-off coefficients* that indicate how much an objective outcome should be deteriorated in order to improve another objective outcome by a unit; therefore, it is important to distinguish also a subset $\hat{Q}^p\subset\hat{Q}$ called the set of *properly efficient* outcomes, such that the corresponding trade-off coefficients are bounded.

The *abstract problem of multiobjective nonlinear programming* consists in determining the entire sets $\hat{Q}^p$ or $\hat{Q}$ or $\hat{Q}^w$. The *practical problem of multiobjective decision support* using nonlinear programming models is different and consists in computing and displaying for the decision maker (or, generally, for the user of the decision support system) some selected properly efficient decisions and outcomes. However, a properly efficient outcome with trade-off coefficients that are extremely high or extremely low does not practically differ from a weakly efficient outcome. Thus, some a priori bound on trade-off coefficients should be defined and properly efficient outcomes that do not satisfy this bound should be excluded. This can be done by defining a slightly broader positive cone:

$$D_\epsilon = \{ q \in R^p : dist(q,D) \leq \epsilon \, \|q\| \} \tag{5}$$

where any norm in R^p is used, also to define the distance between q and D. The corresponding, modified definition of D_ϵ -efficiency:

$$\hat{Q}^{p\epsilon} = \{ \hat{q} \in Q : (\hat{q} + \tilde{D}_\epsilon) \cap Q = \phi \}; \;\; \tilde{D}_\epsilon = D_\epsilon \setminus \{\phi\} \tag{6}$$

applies to properly efficient outcomes that have tradeoff coefficients a priori bounded by approximately ϵ and $1/\epsilon$; such outcomes are also called properly *efficient with* (a priori) *bound.*

The selection of properly efficient outcomes with bound and the corresponding decisions should be easily controlled by the user and should result in any outcome in the set $\hat{Q}^p_\epsilon$ might wish to attain. Before turning to some further theoretical problems resulting from these practical requirements, observe first that the standard formulation of multiobjective nonlinear programming is not the most convenient for the user. Although many other formulations can be rewritten to the standard form by shifting scales or introducing proxy variables, such reformulations should not bother the user and should be automatically performed in the decision support system. Therefore, we present here another basic formulation of the multiobjective nonlinear programming problem, more convenient for typical applications.

A *substantive model* of multiobjective nonlinear programming type consists of the specification of vectors of n decision variables $x \in R^n$ and of m outcome variables $y \in R^m$ together with nonlinear *model equations* defining the relations between the decision variables and the outcome variables and with *model bounds* defining the lower and upper bounds for all decision and outcome variables:

$$y = g(x); \;\; x^{lo} \leq x \leq x^{up}; \;\; y^{lo} \leq y \leq y^{up} \tag{7}$$

where g: $R^n \rightarrow R^m$ is a (differentiable) function that combines the functions f, g' and g'' from the standard formulation. Thus, $m = m' + m'' + p$; but the choice, which of the components of the outcome variable y correspond to constraints and which correspond to objectives, is flexible and can be modified by the user. Between outcome variables, some might be chosen as corresponding to equality constraints; denote these variables by $y^c \in R^{m'} \subset R^m$ and the constraining value for them by b^c to write the additional constraints in the form:

$$y^c = g^c(x) = b^c; \;\; y^{c,lo} \leq b^c \leq y^{c,up} \tag{8}$$

where g^c is a function composed of corresponding components of g. The outcome variables corresponding to equality constraints will be called *guided outcomes* here. Some other outcome variables can be also chosen as optimized objectives or *objective outcomes*. Denote the vector of p objective outcomes by $q \in R^p \subset R^m$ (some of the objective variables might be originally not represented as outcomes of the model, but we can always add them by modifying this model) to write the corresponding objective equations in the form:

$$q = f(x) \tag{9}$$

where f is also composed of corresponding components of g. Thus, the set of attainable objective outcomes is again $Q = f(X)$, but the set of admissible decisions X is defined by:

$$X = \{ x \in R^n : \;\; x^{lo} \leq x \leq x^{up}; \;\; y^{lo} \leq q(x) \leq y^{up}; \;\; g^c(x) = b^c \} \tag{10}$$

Moreover, the objective outcomes are not necessarily maximized; some of them might be minimized, some maximized, some stabilized or kept close to given *aspiration levels* (that is, minimized if their value is above aspiration level and maximized if their value is below aspiration level). All these possibilities can be summarized by introducing a different definition of positive cone D:

$$D=\{q\in R:\ q_i\geq 0,\ 1\leq i\leq p';\ q_i\leq 0,\ p'+1\leq i\leq p'';\ q_i=0,\ p''+1\leq i\leq p\} \tag{11}$$

where the first p' objectives are to be maximized, the next from $p'+1$ until p'' - minimized, and the last from $p''+1$ until p - stabilized. The definition of the cone D_ϵ does not change its analytical form (5), although the cone itself changes appropriately. Actually, the user needs only to define what to do with subsequent objectives; the concept of the positive cones D and D_ϵ is used here only in order to define comprehensively what are efficient and properly efficient outcomes for the multiobjective problem.

Given some aspiration levels $\bar{q}_i$ for stabilized objectives and the requirement that these objectives should be minimized above and maximized below aspiration levels, the set of efficient outcomes can be defined only relative to the aspiration levels. However, since the user can define aspiration levels arbitrarily, of interest here is the union of such relative sets of efficient outcomes. Let $\tilde{D}=D\backslash\{0\}$ and $\tilde{D}_\epsilon=D_\epsilon\backslash\{0\}$; then the outcomes that might be efficient or properly efficient with bound for arbitrary aspiration levels for stabilized objectives can be defined, as before, by the relations (3) or (6). The weakly efficient outcomes are of no practical interest in this case, since the cone D typically has empty interior which implies that weakly efficient outcomes coincide with all attainable outcomes.

The stabilized outcomes in the above definition of efficiency are, in a sense, similar to the guided outcomes; however, there is an important distinction between these two concepts. Equality constraints must be satisfied; if not, then there are no admissible solutions for the model. Stabilized objective outcomes should be kept close to aspiration levels, but they can differ from those levels if, through this difference, other objectives can be improved. The user of a decision support system should keep this distinction in mind and can, for example, modify the definition of the multiobjective analysis problem by taking some outcomes out of the guided outcome category and putting them into the stabilized objective category.

By adding shifting scales, adding a number of proxy variables and changing the interpretation of the function g, the substantive model formulation (7), (8), (9), (10) together with its positive cone (11) and the related concept of efficiency could be equivalently rewritten to the standard form of multiobjective nonlinear programming (1); this, however, does not concern the user. More important is the way of user-controlled selection of an efficient decision and outcome from the set (3) or (6). For stabilized objective outcomes, the user can change the related aspiration levels in order to influence this selection; *it is assumed here that he will do so for all objective outcomes*, that is, *use the corresponding aspiration levels in order to influence the selection of efficient decisions.* The aspiration levels are denoted here $\bar{q}_i$ or $\bar{q}$ as a vector and called also, equivalently, *reference points.*

A special way of parametric scalarization of the multiobjective analysis problem is utilized for the purpose of influencing the selection of efficient outcomes by changing reference points. This parametric scalarization is obtained through maximizing an *order-approximating achievement function* (see Wierzbicki 1983, 1986). There are several forms of such functions; properly efficient outcomes with approximate bound ϵ, $1/\epsilon$ are obtained when maximizing a function of the following form:

$$s(q,\bar{q})=\min_{1\leq i\leq p} z_i(q_i,\bar{q}_i)+\frac{\epsilon}{p}\sum_{i=1}^{p} z_i(q_i,\bar{q}_i) \tag{12}$$

where the parameter ϵ should be positive, even if very small; if this parameter would be equal zero, then the above function would not be ***order-approximating*** any more, but order-representing, and its maximal points could correspond to weakly efficient outcomes.

The functions $z_i(q_i,\bar{q}_i)$ are defined by:

$$z_i(q_i,\bar{q}_i)=\begin{cases}(q_i-\bar{q}_i)/s_i, & \text{if } 1\leq i\leq p', \\ (\bar{q}_i-q_i)/s_i, & \text{if } p'+1\leq i\leq p'', \\ min(z_i', z_i''), & \text{if } p''+1\leq i\leq p'',\end{cases} \tag{13}$$

where

$$z_i'=(q_i-\bar{q}_i)/s_i', \quad z_i''=(\bar{q}_i-q_i)/s_i'' \tag{14}$$

The coefficients s_i, s_i', $s_i''$are scaling units for all objectives, either defined by the user or determined automatically in the system, see further comments.

The achievement function $s(q,\bar{q})$ can be maximized with $q=f(x)$ over $x\in X$; however, the function (12) is nondifferentiable (for example, if $q=\bar{q}$). On the other hand, if the function $g(x)$ (and thus also $f(x)$) is differentiable, then the maximization of function (12) in the system can be converted automatically to an equivalent differentiable nonlinear programming problem by introducing proxy variables and substituting the *min* operation in (12) by a number of additional inequalities. If the coefficient $\epsilon > 0$, then the achievement function has the following properties (see Wierzbicki, 1986):

a) For an arbitrary aspiration level or reference point $\bar{q}$, not necessarily restricted to be attainable ($\bar{q}\in Q$) or not attainable ($\bar{q}\notin Q$), each maximal point $\hat{q}$ of the achievement function $s(q,\bar{q})$ with $q=f(x)$ over $x\in X$ is a D_ϵ-efficient solution, that is, a properly efficient solution with tradeoff coefficients bounded approximately by ϵ and $1/\epsilon$.

b) For any properly efficient outcome $\hat{q}$ with tradeoff coefficients bounded by ϵ and $1/\epsilon$, there exist such reference points $\bar{q}$ that the maximum of the achievement function $s(q,\bar{q})$ is attained at the properly efficient outcome $\hat{q}$. In particular, if the user (either by chance or as a result of a learning process) specifies a reference point $\bar{q}$ that in itself is such properly efficient outcome, $\bar{q}=\hat{q}$, then the maximum of the achievement function $s(q,\bar{q})$, equal zero, is attained precisely at this point.

c) If the reference point $\bar{q}$ is 'too high' (for maximized outcomes; 'too low' for minimized outcomes), then the maximum of the achievement function, smaller than zero, is attained at an efficient outcome $\hat{q}$ that best approximates uniformly, in the sense of scaling units s_i, the reference point. If the reference point $\bar{q}$ is 'too low' (for maximized outcomes; 'too high' for minimized outcomes and it can happen only if there are no stabilized outcomes), then the maximum of the achievement function, larger than zero, is attained at an efficient outcome $\hat{q}$ that is uniformly, in the sense of scaling units s_i, 'higher' than the reference point.

d) By changing his reference point $\bar{q}$, the user can continuously influence the selection of the corresponding efficient outcomes $\hat{q}$ that maximize the achievement function.

The parameter ϵ in the achievement function determines bounds on tradeoff coefficients: if an efficient solution has tradeoff coefficients that are too large or too small (say, lower than 10 or higher than 10) than it does not differ for the decision maker from weakly efficient outcomes - some of its components could be improved without practically deteriorating other components. Another interpretation of this parameter is that it indicates how much an average overachievement (or underachievement) of aspiration levels should correct the minimal overachievement (or maximal underachievement) in the function (12).

The maximization of an achievement function in IAC-DIDAS-N is performed by a specially developed nonlinear optimization algorithm, called solver. Since this maximization is performed repetitively, at least once for each interaction with the user that changes the parameter q, there are special requirements for the solver that distinguish this algorithm from typical nonlinear optimization algorithms: it should be robust, adaptable and efficient, that is, it should compute reasonable fast an optimal solution for optimization problems of a broad class (for various differentiable functions g(x) and f(x)) without requiring from the user that he adjusts special parameters of the algorithm in order to obtain a solution. The experience in applying nonlinear optimization algorithms in decision support systems (see Kreglewski and Lewandowski, 1983, Kaden and Kreglewski, 1986) has led to the choice of an algorithm based on penalty shifting technique and projected conjugate gradient method. Since a penalty shifting technique anyway approximates nonlinear constraints by penalty terms, an appropriate form of an achievement function that differentiably approximates function (12) has been also developed and is actually used in IAC-DIDAS-N. This *smooth order-approximating achievement function* has the form:

$$s(q,\bar{q})=1-\left\{\frac{1}{p}\left[\sum_{i=1}^{p''}(w_i)^{\alpha}+\sum_{i=p''}^{p}+1(max(w_i',\ w_i''))^{\alpha}\right]\right\}^{1/\alpha} \tag{15}$$

where w_i, w_i', w_i'' are functions of q_i, $\bar{q}_i$:

$$w(q_i,\bar{q}_i)=\begin{cases}(q_{i,max}-q_i)/s_i, & \text{if } 1\le i\le p' \\ (q_i-q_{i,min})/s_i, & \text{if } p'+1\le i\le p''\end{cases} \tag{16}$$

$$\left.\begin{aligned} w_i'(q_i,\bar{q}_i)&=(q_{i,max}-q_i)/s_i \\ w_i'(q_i,\bar{q}_i)&=(q_i-q_{i,min})/s_i\end{aligned}\right\},\ \text{if } p''+1\le i\le p, \tag{16b}$$

and the dependence on q_i results from a special definition of the scaling units that are determined by:

$$s_i=\begin{cases}(q_{i,max}-\bar{q}_i)/r_i, & \text{if } 1\le i\le p', \\ (\bar{q}_i-q_{i,min})/r_i, & \text{if } p'+1\le i\le p'',\end{cases} \tag{17a}$$

$$\left.\begin{aligned} s_i'&=(q_{i,max}-\bar{q}_i)/r_i, \ \text{if } 1\le i\le p', \\ s_i'&=(\bar{q}_i-q_{i,min})/r_i, \ \text{if } p'+1\le i\le p'',\end{aligned}\right\},\ \text{if } p''+1\le i\le p, \tag{17b}$$

where r_i are additional weighting coefficients that might be defined by the user (however, the system does not need them and works also well if they are set by their default values $r_i=1$). In the initial analysis phase, the values $q_{i,max}$ and $q_{i,min}$ are set to the upper and lower bounds specified by the user for the corresponding outcome variables; later, they are modified, see further comments. The parameter $\alpha \geq 2$ is responsible for the approximation of the function (12) by the function (15): if $\alpha \rightarrow \infty$ and $\epsilon \rightarrow 0$ then these functions converge to each other (if $r_i=1$ and while taking into account the specific definition of scaling coefficients in (15)). However, the use of too large parameters α results in badly conditioned problems when maximizing function (15), hence $\alpha=4 \cdots 8$ are suggested to be used.

The function (15) must be maximized with $q=f(x)$ over $x \in X$, while x is determined by simple bounds $x^{lo} \leq x \leq x^{up}$ as well as by inequality constraints $y^{lo} \leq g(x) \leq y^{up}$ and equality constraints $g^c(x)=b^c$. In the shifted penalty technique, the following function is minimized instead:

$$P(\chi,\chi'\chi',\chi,u',u'',v)=-s(f(x),\bar{q})+\frac{1}{2}\sum_{i=1}^{p}\chi_i'\max(0,g_i(x)-y^{up_i}+u_i'))^2+ \tag{18}$$

$$+\frac{1}{2}\sum_{i=1}^{p}\chi_i''max(0,y-g(x)+u''))^{2+}+\frac{1}{2}\sum_{i=1}^{m'}\chi_i g^{c_i}(x)-b^{c_i}+v_i))^2$$

where χ',χ'',χ are penalty coefficients and u',u'',v are penalty shifts. This function is minimized over x such that $x^{lo} \leq x \leq x^{up}$ while applying conjugate gradient directions, projected on these simple bounds if one of the bounds becomes active. When a minimum of this penalty function with given penalty coefficients and given penalty shifts (the latter are initially equal zero) is found, the violations of all outcome constraints are computed, the penalty shifts and coefficients are modified according to the shifted-increased penalty technique (see, e.g., Wierzbicki, 1984), and the penalty function is minimized again until the violations of outcome constraints are admissibly small. The results are then equivalent to the outcomes obtained by maximizing the achievement function (15) under all constraints. This technique, though it might seem cumbersome, is according to our experience one of the most robust nonlinear optimization methods; the user of the system is not bothered with its details, since the adjustment of penalty shifts and coefficients is done automatically in this technique.

Another advantage for the user is that he is not bothered with the definition of derivatives of penalty function (18), needed in the conjugate gradient method, nor even with the definition of the derivatives of outcome functions $g_i(x)$. This is unique feature of IAC-DIDAS-N: all needed derivatives are automatically (symbolically) determined and computed either in the nonlinear model generator that supports the model definition phase or in the solver algorithm using shifted penalty technique.

The only parameter that might influence the interaction of the system with the user is the parameter a in the smooth order-approximating function (15). Thus, the user can select this parameter; if this parameter is very large, his control of efficient outcomes obtained by maximizing (15) is somewhat easier, but the solver might take long time or produce not quite robust results in this case. Therefore, the user is advised not to exceed the reasonable range $2<\alpha<8$; the default value is $\alpha = 4$.

The maximization of an achievement function is a convenient way of organizing the interaction between the model and the user. Before the interactive analysis phase, however, the user must firstly define the substantive model, then define the multiobjective analysis problem by specifying outcome variables that should be maximized, minimized, stabilized, guided or *floating* (that is, displayed for users' information only, but not included as optimized or guided objectives; various decision variables of interest to the user can be also included into one of these categories). Before the initial analysis phase, the user should also define some reasonable lower and upper bounds for each variable, which results in an automatic definition of reasonable scaling units s_i for optimized outcome variables. In further phases of analysis, these scaling units s can be further adjusted; this, however, requires an approximation of bounds on efficient solutions. Such an approximation is performed in the initial analysis phase.

The 'upper' bound for efficient solutions could be theoretically obtained through maximizing each objective separately (or minimizing, in case of minimized objectives; in the case of stabilized objectives, the user should know their entire attainable range, hence they should be both maximized and minimized). Jointly, the results of such optimization form a point that approximates from 'above' the set of efficient outcomes $\hat{Q}$, but this point almost never (except in degenerate cases) is in itself an attainable outcome; therefore, it is called the ***utopia point***.

However, this way of computing the 'upper' bound for efficient outcomes is not always practical; thus, IAC-DIDAS-N uses a different way of estimating the utopia point. This way consists in subsequent maximizations of the achievement function $s(q,\bar{q})$ with suitably selected reference points $\bar{q}$. If an objective should be maximized and its maximal value must be estimated, then the corresponding component of the reference point should be very high, while the components of this point for all other maximized objectives should be very low (for minimized objectives, very high; stabilized objectives must be considered as floating in this case, that is, should not enter the achievement function). If an objective should be minimized and its minimal value must be estimated, the corresponding component of the reference point should be very low, while other components of this point are treated as in the previous case. If an objective should be stabilized and both its maximal and minimal values must be estimated, then the achievement function should be maximized twice, first time as if for a maximized objective and the second time as if for a minimized one. Thus, the entire number of optimization runs in utopia point computations is $p''+2(p-p'')$. It can be shown that this procedure gives a very good approximation of the utopia point $\hat{q}^{uto}$, whereas the precise meaning of very high reference component should be interpreted as the upper bound for the objective minus, say, 0.1% of the distance between the lower and the upper bound, while the meaning of very low is the lower bound plus 0.1% of the distance between the upper and the lower bound.

During all these computations, the 'lower' bound for efficient outcomes can be also estimated, just by recording the lowest efficient outcomes that occur in subsequent optimizations for maximized objectives and the highest ones for minimized objectives (there is no need to record them for stabilized objectives, where the entire attainable range is anyway estimated). However, such a procedure results in the accurate, tight 'lower' bound for efficient outcomes - called ***nadir point*** $\hat{q}^{nad}$ - only if $p''=2$; for larger numbers of maximized and minimized objectives, this procedure can give misleading results, while an accurate computation of the nadir point becomes a very cumbersome computational task.

Therefore, IAC-DIDAS-N offers an option of improving the estimation of the nadir point in such cases. This option consists in additional p'' maximization runs for achievement function $s(q,\bar{q})$ with reference points $\bar{q}$ that are very low, if the objective in question should be maximized, very high for other maximized objectives and very low for other minimized objectives, while stabilized objectives should be considered as floating; if the objective in question should be minimized, the corresponding reference component should be very high, while other reference components should be treated as in the previous case. By recording the lowest efficient outcomes that occur in subsequent optimizations for maximized objectives (and are lower than the previous estimation of nadir component) and the highest ones for minimized objectives (higher that the previous estimation of nadir component), a better estimation $\hat{q}^{nad}$ of the nadir point is obtained.

Once the approximate bounds $\hat{q}^{uto}$ and $\hat{q}^{nad}$ are computed and known to the user, they can be utilized in various ways. One way consists in computing a ***neutral efficient solution***, with outcomes situated approximately 'in the middle' of the efficient set. For this purpose, the reference point $\bar{q}$ is situated at the utopia point $\hat{q}^{uto}$ (only for maximized or minimized outcomes; for stabilized outcomes, the user-supplied reference component $\bar{q}_i$ must be included here) and the scaling units are determined by:

$$s_i=|\hat{q}_i^{uto}-\hat{q}_i^{nad}|,\ 1\le i\le p'' \tag{19a}$$

for maximized or minimized outcomes, and:

$$\left.\begin{aligned} s_i'&=\bar{q}_i-\hat{q}_i^{nad}+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad})\\ s_i''&=\hat{q}_i^{uto}-\bar{q}_i+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad})\end{aligned}\right\}\ p''+1\le i\le p \tag{19b}$$

for stabilized outcomes, while the components of the utopia and the nadir points are interpreted respectively as the maximal and the minimal value of such an objective; the corrections by $0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}$ ensures that the scaling coefficients remain positive, if the user selects the reference components for stabilized outcomes in the range $\hat{q}_i^{nad}\le\bar{q}_i\le\hat{q}_i^{uto}$ (if he does not, the system automatically projects the reference component on this range; the user-supplied weighting coefficients are automatically set to their default values $r_i=1$ when computing a neutral efficient outcome). By maximizing the achievement function $s(q,\bar{q})$ with such data, the neutral efficient solution is obtained and can be utilized by the user as a starting point for further interactive analysis of efficient solutions.

In further interactive analysis, an important consideration is that the user should be able to easily influence the selection of the efficient outcomes $\hat{q}$ by changing the reference point $\bar{q}$ in the maximized achievement function $s(q,\bar{q})$. It can be shown (see Wierzbicki, 1986) that best suited for the purpose is the choice of scaling units determined by the difference between the slightly displaced utopia point and the current reference point:

$$s_i=\begin{cases}(\hat{q}_i^{uto}-\bar{q}_i+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}))/r_i, & \text{if } 1\le i\le p'\\ (\bar{q}_i-\hat{q}_i^{uto}+0.01(\hat{q}_i^{nad}-\hat{q}_i^{uto}))/r_i, & \text{if } p'+1\le i\le p''\end{cases} \tag{20a}$$

for maximized or minimized outcomes. For stabilized outcomes, the scaling units are determined somewhat differently than in (19b):

$$\left.\begin{aligned} s_i'&=(\hat{q}_i^{uto}-\bar{q}_i+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}))/r_i\\ s_i''&=(\bar{q}_i-\hat{q}_i^{nad}+0.01(\hat{q}_i^{uto}-\hat{q}_i^{nad}))/r_i\end{aligned}\right\},\ \text{if } p''+1\le i\le p \tag{20b}$$

It is assumed now that the user selects the reference components in the range $\hat{q}_i^{nad}\le\bar{q}_i\le\hat{q}_i^{uto}$ for maximized and stabilized outcomes or $\hat{q}_i^{uto}\le\bar{q}_i\le\hat{q}_i^{nad}$ for minimized outcomes (if he does not, the system automatically projects the reference component on these ranges). The weighting coefficients r_i might be used to further influence the selection of efficient outcomes, but the automatic definition of scaling units is sufficient for this purpose even if $r_i=1$ by default; thus, the user needs not be bothered by their definition. The interpretation of the above way of setting scaling units is that the user attaches implicitly more importance to reaching a reference component $\bar{q}_i$ if he places it close to the known utopia component; in such a case, the corresponding scaling unit becomes smaller and the corresponding objective component is weighted stronger in the achievement function $s(q,\bar{q})$. Thus, this way of *scaling relative to utopia-reference difference* is taking into account the implicit information given by the user in the relative position of the reference point.

When the relative scaling is applied, the user can easily obtain - by suitably moving reference points - efficient outcomes that are either situated close to the neutral solution, in the middle of efficient outcome set $\hat{Q}$, or in some remote parts of the set $\hat{Q}$, say, close to various extreme solutions. Typically, several experiments of computing such efficient outcomes give enough information for the user to select an actual decision - either some efficient decision suggested by the system, or even a different one, since even the best substantive model cannot encompass all aspects of a decision situation. However, there might be some cases in which the user would like to receive further support - either in analysing the sensitivity of a selected efficient outcome, or in converging to some best preferred solution and outcome.

For analysing the sensitivity of an efficient solution to changes in the proportions of outcomes, a *multidimensional scan* of efficient outcomes is applied in IAC-DIDAS-N. This operation consists in selecting an efficient outcome, accepting it as a base $\bar{q}^{bas}$ for reference points, and performing p sup " additional optimization runs with the reference points determined by:

$$\bar{q}_i=\bar{q}_i^{bas}+\beta(\hat{q}_j^{uto}-\hat{q}_j^{nad}),\ \ \bar{q}_i=\bar{q}_i^{bas},\ \ i\neq j,\ \ 1\leq j\leq p'', \tag{21}$$

where β is a coefficient determined by the user, $-1\leq\beta\leq 1$; if the relative scaling is used and the reference components determined by (21) are outside the range $\hat{q}_j^{nad}$, $\hat{q}_j^{nad}$, they are projected automatically on this range. The reference components for stabilized outcomes are not perturbed in this operation (if the user wishes to perturb them, he might include them, say, in the maximized category). The efficient outcomes resulting from the maximization of the achievement function $s(q,\bar{q})$ with such perturbed reference points are typically also perturbed mostly along their subsequent components, although other their components might also change.

For analysing the sensitivity of an efficient solution when moving along a direction in the outcome space - and also as a help in converging to a most preferred solution - a *directional scan* of efficient outcomes is implemented in IAC-DIDAS-N. This operation consists again in selecting an efficient outcome, accepting it as a base $\bar{q}^{bas}$ for reference points, selecting another reference point $\bar{q}$, and performing a user-specified number K of additional optimizations with reference points determined by:

$$\bar{q}(k)=\bar{q}^{bas}+\frac{k}{K}(\bar{q}-\bar{q}^{bas}),\ \ 1\leq k\leq K \tag{22}$$

The efficient solutions $\hat{q}(k)$ obtained through maximizing the achievement function $s(q,\bar{q}(k))$ with such reference points constitute a cut through the efficient set $\hat{Q}$ when moving approximately in the direction $\bar{q}-\bar{q}^{bas}$. If the user selects one of these efficient solutions, accepts as a new $\bar{q}^{bas}$ and performs next directional scans along some new directions of improvement, he can converge eventually to his most preferred solution (see Korhonen, 1985). Even if he does not wish the help in such convergence, directional scans can give him valuable information.

Another possible way of helping in convergence to the most preferred solution is choosing reference points as in (22) but using a harmonically decreasing sequence of coefficients (such as 1/j, where j is the iteration number) instead of user-selected coefficients k/K. This results in convergence even if the user makes stochastic errors in determining next directions of improvement of reference points, or even if he is not sure about his preferences and learns about them during this analysis (see Michalevich, 1986). Such a convergence, called here forced convergence, is rather slow. Neither the forced convergence nor multidimensional nor directional scan are yet implemented in the first pilot version of IAC-DIDAS-N, but they will be implemented in later versions.

REFERENCES

Dreyfus, S. (1984) Beyond rationality. In M.Grauer, M.Thompson, A.P.Wierzbicki (eds), Plural Rationality and Interactive Decision Processes, Proceedings Sopron 1984. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 248).

Kaden, S. (1985) Decision support system for long-term water management in open-pit lignite mining areas. In G.Fandel, M.Grauer, A.Kurzhanski and A.P.Wierzbicki (eds), Large Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Kaden, S. and T.Kreglewski (1986) Decision support system MINE - problem solver for nonlinear multi-criteria analysis. CP-86-5, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Kreglewski, T. and A.Lewandowski (1983) MM-MINOS - an integrated decision support system. CP-83-63. International Institute for Applied Systems Analysis, Laxenburg, Austria. Korhonen, P. (1985) Solving discrete multiple criteria decision problems by using visual interaction. In G.Fandel, M.Grauer, A.Kurzhanski and A.P.Wierzbicki (eds), Large Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Lewandowski, A., M.Grauer, A.P.Wierzbicki (1983) DIDAS: theory, implementation. In M.Grauer, A.P.Wierzbicki (eds), Interactive Decision Analysis, Proceedings Laxenburg 1983. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 229).

Makowski, M., and J.Sosnowski (1984) A decision support system for planning and controlling agricultural production with a decentralized management structure. In M.Grauer, M.Thompson, A.P.Wierzbicki (eds), Plural Rationality and Interactive Decision Processes, Proceedings Sopron 1984. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 248).

Messner, S. (1985) Natural gas trade in Europe and interactive decision analysis. In G.Fandel, M.Grauer, A.Kurzhanski and A.P.Wierzbicki (eds), Large Scale Modelling and Interactive Decision Analysis, Proceedings Eisenach 1985. Springer Verlag, Berlin Heidelberg New York Tokyo (Lecture Notes in Economic and Mathematical Systems 273).

Michalevich, M. (1986) Stochastic approaches to interactive multicriteria optimization problems. WP-86-10. International Institute for Applied Systems Analysis, Laxenburg, Austria. Wierzbicki, A.P. (1983) A mathematical basis for satisficing decision making. Mathematical Modelling 3, 391-405.

Wierzbicki, A.P (1984) Models and Sensitivity of Control Systems. Elsevier, Amsterdam.

Wierzbicki, A.P. (1986) On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spektrum 8, 73-87.

An Experimental System Supporting Multiobjective Bargaining Problem: a Methodological Guide

Piotr Bronisz, Lech Krus, Bozena Lopuch

Systems Research Institute, Polish Academy of Sciences.

ABSTRACT

The experimental system illustrates a new algorithm of interactive search for a cooperative, efficient solution in a multicriteria bargaining problem. The paper presents the methodological basis for the algorithm together with an outline of the program. The system and parts of the theoretical research have been done under a contracted study agreement with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis. The system is implemented on professional microcomputers compatible with IBM-PC-XT (with Hercules or Color graphics card). It is written in Turbo Pascal utilizing Turbo Graphix Toolbox.

1. INTRODUCTION

In most approaches (see Nash [1950], Raiffa [1953], Kalai and Smorodinsky [1975], A.E.Roth [1979]), the bargaining problem has been considered in the case of unicriterial payoffs of players, i.e. when the preferences of particular players are expressed by utility functions. In many practical applications however, players trying to balance a number of objectives might have difficulties while constructing such utility functions. Moreover, the classic literature considers mostly axiomatic models of bargaining which yield one-shot solutions and do not result in procedures describing a process of reaching a binding agreement.

In this paper we consider n players each with m objectives. We are dealing with a multiobjective bargaining problem; in this problem, the players are faced with an agreement set of feasible outcomes, and any such outcome can be accepted as the result if it is specified by an unanimous agreement of all players. In the event that no unanimous agreement is reached, the players act independently; the joint outcome of such independent actions is called the disagreement solution. If there are feasible outcomes which all participants prefer to the disagreement solution, then there is an incentive to reach an agreement. In most situations, players differ in their opinions which outcome is most preferable, hence there is a need for bargaining and negotiation.

Dealing with multiple payoffs, we do not assume that there exist explicitly given utility functions of the players. In this paper, under a suggestion of Wierzbicki [1983], an interactive process is discussed starting from the disagreement solution and leading to a nondominated, individually rational solution belonging to the agreement set. During the interaction, players can express their preferences and can influence the course of the iterative process. The proposed process consists of two phases. In the first phase, the players act independently on their disagreements sets and select the disagreement solution. In the second phase, the cooperative action on the agreement set is considered.

International cooperation can give many examples of bargaining problems. Let us consider several countries interested in the development of production of some kind of goods. Each country can decide to realize its own independent development program or they can create a joint program. A program is characterized by volumes of produced goods, and required resources. The development programs are typically described by linear programming problems in which the cost of resources is minimized under the given volumes of developed production, or the production is maximized under the constraints resulting from the given resources. Such unicriterial models are not suitable for real decision making problems. Decision makers, especially in nonmarket economy countries, have various preferences related to the particular kind of resources and the volume of developed production. Let us observe that prices in nonmarket economies are not necessarily a good way for aggregating resources into a joint cost. Therefore, we might also consider a Multicriteria model in which particular resources are considered as independent criteria to be minimized and the produced good to be maximized. In such an example, linear programming problems related to independently considered development programs describe the disagreement sets in the space of goals, where the linear programming problem related to joint project describes the agreement set.

As it has been mentioned before, the theory of multiobjective bargaining problem has not been fully developed yet. Moreover, concepts of iterative solutions of a bargaining problem have not been considered. Therefore, the main effort in this research has been directed towards theoretical and methodological problems. Experimental software has been developed to compute examples illustrating theoretical and methodological results; it cannot be treated as a transferable software yet. However, this software has been useful in testing and checking various theoretical concepts. In this state of research, we have assumed a relatively simple description of disagreement and agreement sets. Two players are considered, each of them having three goals - one produced good and two kinds of resources (capital investments and labor force). The development programs are described by functions that assign required volumes of resources to the volume of produced goods. In the final version of the software, more complex linear programming description of the development programs will be also investigated.

2. PROBLEM FORMULATION AND DEFINITIONS

Let $N = \{1,2,\ldots,n\}$ be the finite set of players, each player having m objectives. A multiobjective bargaining problem can be described in the form

$$(S , S_1 , S_2 , \cdots , S_n) ,$$

where $S_i \in R^m$ is a disagreement set of the $i-th$ player, $i \in N$, $S \in R^{n \times m}$ is an agreement set of all the players.

The bargaining problem has the following intuitive interpretation: every point x , $x=(x_1,x_2,\ldots,x_n)$, $x_i=(x_{i1},x_{i2},\ldots,x_{im}) \in R^m$, in the agreement set S represents payoffs for all the players that can be reached when they do cooperate with each other (x_{ij} denotes the payoff of the $j-th$ objective for the $i-th$ player). If the players do not cooperate, each player $i \in N$ can reach the payoffs from his disagreement set S_i. The players are interested in finding an outcome in S which will be agreeable to all the players.

We employ a convention that for $x,y \in R^k$, and $x \geq y$ implies $x_i \geq y_i$ for all $i \in N$, $x > y$ implies $x \geq y$, $x \neq y$ *and* $x \gg y$ implies $x_i > y_i$ for all $i \in N$. We say that $x \in R^k$ is a weak Pareto optimal point in X if $x \in X$ and there is no $y \in X$ such that $y \gg x$; $x \in X$ is a Pareto optimal point in X if there is no $y \in X$ such that $y > x$.

In this paper, we assume that each player tries to maximize his every objective. As it was mentioned, the proposed interactive process consists of two phases. In the first phase, each player $i \in N$ acts independently of the others on his disagreement set S_i to select the most preferable point d_i. In the second, the players bargain over the agreement set S assuming that $d = (d_1, d_2, \ldots, d_n)$ is the status quo or disagreement point.

Let $R^k_x = \{y \in R^k : y \geq x\}$. We say that a set $X \subset R^k$ belongs to the class B^k if and only if X satisfies the following conditions.

(i) For any $x \in X$, the set $X \cap R^k_x$ is compact.

(ii) The set X is comprehensive, i.e. for any $x \in X$, if $y \in R^k$ is such that $x \geq y$ then $y \in X$.

(iii) For any $x \in X$, let $Q(X,x) = \{i : y \geq x,\ y_i > x_i \ \textit{for some}\ \ y \in X\}$. Then for any $x \in X$, there exists $y \in X$ such that $y \geq x$, $y_i > x_i$ for each $i \in Q(X,x)$.

In the paper, we confine our consideration to the multiobjective bargaining problems satisfying: $S \in B^{n \times m}$, $S_i \in B^m$ for $i \in N$.

Intuitively, Condition (i) states that the set X is closed and upper bounded. Condition (ii) says that objectives are disposable, i.e. that if the players can reach the outcome x then they can reach any outcome worse than x. $Q(X,x)$ is the set of all coordinates in R^k, payoffs of whose members can be increased from x in X. Condition (iii) states that the set of Pareto optimal points in X contains no "holes". Figure 1 shows an example of a set that does not fulfill Condition (iii). It is easy to notice that, for example, any convex set satisfies Condition (iii).

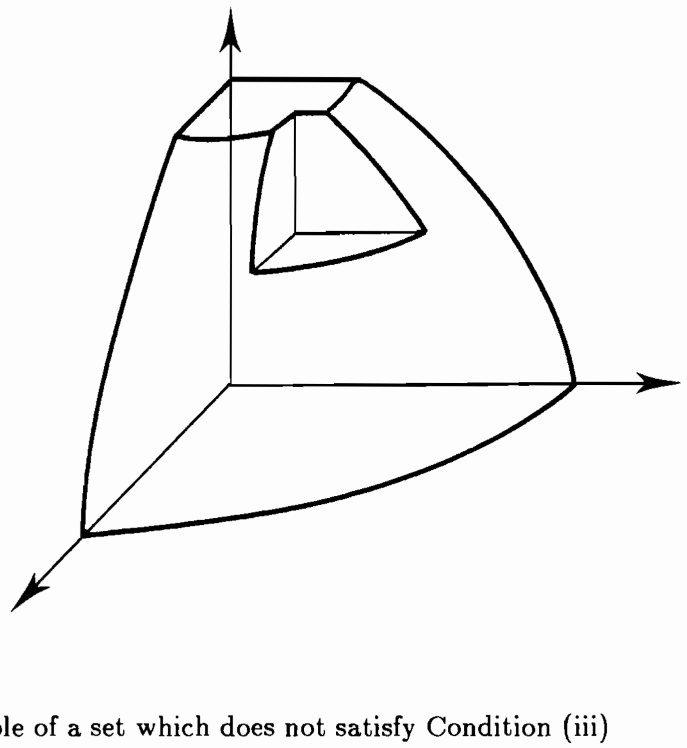


Fig 1. An example of a set which does not satisfy Condition (iii)

3. FIRST PHASE. MULTIOBJECTIVE DECISION PROBLEM

Let us consider the $i-th$ player, $i \in N$. To simplify notation, let $X = S_i$ and $M = \{1,2,\ldots,m\}$.

We define an affine transformation of R^m by

$$T(\cdot,x^r,x^0) : R^m \rightarrow R^m \ , \ T = (T_1,T_2,\ldots,T_m) \ ,$$

$$T_i(x,x^r,x^0) = (x_i - x_i^0)/(x_i^r - x_i^0) \ \text{ for } i \in M \ ,$$

where $x^0 \in X$, $x^r \in R^m$, $x^r \gg x^0$. The transformation T depends on two points settled by the player, the point x^0 define lower bounds on efficient outcomes (it may be, for example, the "nadir" point of X), the point x^r, called a reference point, reflects preferences of the player. The transformation T normalizes the problem in a sense that $T(x^0,x^r,x^0) = (0,0,\ldots,0)$ and $T(x^r,x^r,x^0) = (1,1,\ldots,1)$.

To select a Pareto optimal outcome in X according to x^0 and x^r, we utilize the Rawlsian lexmin principle (see Rawls [1971], [1983]). Let $\succ^l$ be the lexicographical ordering on R^m, i.e. for $x,y \in R^m$, $x \succ^l y$ if and only if there is $i \in M$ such that $x_i > y_i$ and $x_j = y_j$ for $j < i$. Let $L : R^m \rightarrow R^m$ be such that for $x \in R^m$, there is a permutation on M , π, with $L(x) = \pi^* x$ and $L_1(x) \leq L_2(x) \leq \cdots \leq L_m(x)$. Then the lexicographical maxmin $\succ$ ordering on R^m (with respect to x^0 and x^r), is defined by

$$x \succ y \ \Leftrightarrow \ L(T(x,x^r,x^0)) \succ^l L(T(y,x^r,x^0))$$

for $x,y \in R^m$.

Let $F(X,x^r,x^0)$ denote the lexicographical maxmin outcome of X according to x^r and x^0, and is defined by $F(X,x^r,x^0) = x \ \Leftrightarrow \ x$ is a maximal element in X according to $\succ$.

Theorem 1. (Bronisz and Krus [1987b]) For $X \in B^m$, $x^0 \in X$, $x^r \in R^m$, $x^r \gg x^0$, $F(X,x^r,x^0)$ exists uniquely and is Pareto optimal in X.

Theorem 1 is a generalization of the result in Schmeidler [1969] for a nonconvex set X. The proposed approach is very close to the achievement function concept (Wierzbicki [1982]) from the point of view of the user. Analogously, a special way of the parametric scalarization of the multiobjective problem is utilized to influence on the selection of Pareto optimal outcomes by changing reference points. However, in place of parametric scalarization through the order-approximating achievement function (for example, weighted sum of l_1 and l_∞ Chebyshev norm), we propose the scalarization by the weighted l_∞ norm and then the lexicographical improvement of a weak Pareto point to a Pareto optimal outcome. The proposed solution, under not quite restrictive assumptions about the set X ($X \in B^m$) can be determined in a simple way (even in a case of complicated nonconvex sets where the problem of maximizing the Chebyshev norm can be ill-conditioned).The corresponding algorithm is based on several (at most m) directional maximizations using, for example, a bisection method which works very quickly and effectively.

The procedure for locating the lexicographical maxmin outcome can be formalized as follows. If Q is a subset of M, let $e(Q) \in R^m$ be such that $e_i(Q) = (x_i^r - x_i^0)$ for $i \in Q$ and $e_i(Q) = 0$ otherwise. Given $y \in X$ with $Q(X,y) \neq \phi$, define $x(X,y) \in X$ by

$$x(X,y) = max_{\geq} \{x \in X : x = y + ae(Q(X,y)) \ \mathit{for\ some}\ a \in R\}.$$

We construct a sequence $\{x^j\}_{j=0}^{\infty}$ such that $x^0 \in X$ is fixed by the player, and

$x^j = x(X,x^{j-1})$ for $j=1,2,\cdots$. Such sequence is uniquely defined and the following theorem can be proved (Bronisz and Krus [1987b]). It is a generalization of the results in Imai [1983] for nonconvex set X.

Theorem 2. For $X \in B^m$, $x^0 \in X$, $x^r \in R^m$, $x^r \gg x^0$, let k be the smallest j with $x^j = x^{j+1}$. Then $k \le m$ and $F(X,x^r,x^0) = x^k$ i.e. the sequence $\{x^j\}_{j=0}^{\infty}$ yields the lexicographical maxmin outcome in X in at most m steps.

4. SECOND PHASE. COOPERATION

Let $d_i \in S_i$ be the lexicographical maxmin outcome in S_i calculated in the first phase by the $i-th$ player, $i \in N$, and let $d=(d_1,d_2,\ldots,d_n)$ be the resulting disagreement solution or status quo point. Now we reduce the problem to the pair (S,d), where S is the agreement set. If the point d belongs to S and is not Pareto optimal point in S then there is an incentive to cooperation among the players; in the other case cooperation is not profitable. We assume that $d \in S$ and there exists $x \in S$ such that $x > d$.

We are interested in a constructive procedure that is acceptable by all players, starts at the status quo point and leads to a Pareto optimal point in S. The procedure can be described as a sequence, $\{d^t\}_{t=0}^{k}$, of agreement points d^t such that $d^0=d$, $d^t \in S$, $d^t \ge d^{t-1}$, for $t=1,2,\ldots$, and d^k is a Pareto optimal point in S. The assumption $d^t \ge d^{t-1}$ follows from the fact that no player will accept improvement of payoffs for other players at the cost of his concession. At every round t, each player $i \in N$ specifies his improvement direction $\lambda_i^t \in R^m$, $\lambda_i^t > 0$ and his confidence coefficient $\alpha_i^t \in R$, $0 < \alpha_i^t \le 1$. The improvement direction λ_i^t indicates the $i-th$ players preferences over his objectives at round t. The confidence coefficient α_i^t reflects his ability at round t to describe preferences and to predict precisely all consequences and possible outcomes in S. For more detailed justification, see Fandel, Wierzbicki [1985], and Bronisz, Krus, Wierzbicki [1987].

We propose an interactive negotiation process defined by a sequence

$$\{d^t\}_{t=0}^{\infty} \text{ such that } d^0=d,$$
$$d^t = d^{t-1} + \epsilon^t[u(S,d^{t-1},\lambda^t) - d^{t-1}] \quad \text{for } t=1,2,\ldots,$$

where $\lambda^t \in R^{n \times m}$, $\lambda^t=(\lambda_1^t,\lambda_2^t,\ldots,\lambda_n^t)$, is the improvement direction specified jointly by all players, $u(S,d^{t-1},\lambda^t) \in R^{n \times m}$ is the utopia point relative to the direction λ^t at round t defined by

$$u(S,d^{t-1},\lambda^t) = (u_1(S,d^{t-1},\lambda_1^t),u_2(S,d^{t-1},\lambda_2^t),\ldots,u_n(S,d^{t-1},\lambda_n^t))$$
$$u_i(S,d^{t-1},\lambda^t) = max_{\ge}\{x_i \in R^m : x \in S,\ x \ge d^{t-1},\ x_i = d_i^{t-1} + a\lambda_i^t \ \textit{for some} \ a \in R\}.$$

Moreover, $\epsilon^t = min(\alpha_1^t,\alpha_2^t,\ldots,\alpha_n^t,\alpha_{\max}^t) \in R$, where $\alpha_{\max}^t$ is the maximal number α such that $d^{t-1}+\alpha[u(S,d^{t-1},\lambda^t)-d^{t-1}]$ belongs to S.

Intuitively, the utopia point $u(S,d^{t-1},\lambda^t)$ relative to the direction λ^t reflects the "power" of the particular players when the improvement direction λ^t is specified at round t. The individual outcome $u_i(S,d^{t-1},\lambda^t)$ is the maximal payoff in S for the $i-th$ player from d^{t-1} according to improvement direction λ_i^t, while ϵ^t is the minimal confidence coefficient of the players at round t (we assume that no player can agree on a coefficient greater than his) such that a new calculated agreement point belongs to S.

Theorem 8. For an agreement set $S \in B^{n \times m}$ and a status quo point $d \in S$, let improvement directions of the players, $\lambda^t \in Rn \times m$, $\lambda^t = (\lambda_1^t, \lambda_2^t, \ldots, \lambda_n^t)$, be such that $\lambda_{ij}^t > 0$ if the coordinate ij belongs to $Q(S,d^{t-1})$ and $\lambda_{ij}^t = 0$ in the other case, for $t=1,2,\cdots$. Then the interactive negotiation process $\{d^t\}_{t=0}^{\infty}$ yields a Pareto optimal outcome in S. If k is the smallest t with $d^t = d^{t+1}$ then d^k is a Pareto optimal outcome in S. In other case, if there is a number $\alpha > 0$ such that $\alpha_i^t \geq \alpha$ for $i \in N$, $t=1,2,\cdots$, then the limit $\lim\limits_{t \to \infty} d^t$ exists and it is a Pareto optimal outcome in S. Moreover, for each $t=1,2,\cdots$ and for each $i \in N$ there is a number β such that $d_i^t - d_i^{t-1} = \beta \lambda_i^t$, i.e. at each round t the improvement of players' payoffs is compatible with their improvement directions.

Proof. From the proposed construction of the sequence $\{d^t\}_{t=0}^{\infty}$ and from the assumptions (i)-(iii) for S, it follows that if $d^k = d^{k+1}$ then $Q(S,d^k) = \phi$, i.e no coordinate of d^k may be improved in S. Thus d^k is a Pareto optimal outcome in S. In other case let us consider a sequence $\{d^t\}_{t=0}^{\infty}$. This sequence is monotonically increasing and limited so it is convergent. Let $d_{\lim} = \lim\limits_{t \to \infty} d^t$. From the proposed construction, $d_{\lim} \in S$. Let us assume that $d_{\lim}$ is not a Pareto optimal outcome in S. Then for any round t, we have

$$\|d^t - d^{t-1}\| = \|\epsilon^t[u(S,d^{t-1},\lambda^t) - d^{t-1}]\| \geq \alpha * \|u(S,d_{\lim},\lambda^t) - d_{\lim}\| \geq$$

$$\geq \alpha * \min\{\|u(S,d_{\lim},\lambda) - d_{\lim}\| : \lambda \in R^{n \times m},$$

$$\lambda_{ij} > 0 \;\; \textit{if} \;\; ij \notin Q(S,d_{\lim}),\; \lambda_{ij} = 0 \;\; \textit{otherwise} \;\; \} = \gamma > 0.$$

Thus the sequence $\{d^t\}_{t=0}^{\infty}$ is not convergent. Contradiction. This proves that $d_{\lim}$ is a Pareto optimal outcome in S.
The statement $d_i^t - d_i^{t-1} = \beta \lambda_i^t$ follows from the fact that $u(S,d^{t-1},\lambda^t) = d^{t-1} + \delta \lambda^t$ for some $\delta \in R$. *Q.E.D.*

The interactive negotiation process, $\{d^t\}_{t=0}^{\infty}$, is a generalization of the iterative negotiation model for the unicriterial bargaining problem proposed and discussed by Bronisz, Krus [1986a] and Bronisz, Krus, Wierzbicki [1987].

The presented approach has been examined in a case of one-round process with confidence coefficients of the players equal one (Bronisz, Krus [1987a]). Given an improvement direction of the players $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n) \in R^{n \times m}$, the one-round process yields an outcome $G(S,d,\lambda)$ d^1 such that

$$G(S,d,\lambda) = \max\nolimits_{\geq}\{x \in S : x = d + a[u(S,d,\lambda) - d] \;\text{ for } \textit{some} \;\; a \in R\} \;.$$

$G(S,d,\lambda)$ may be only a weak Pareto optimal point in S. Intuitively, the outcome $G(S,d,\lambda)$ is a unique point of intersection of the line connecting $u(S,d,\lambda)$ to d with the boundary of S.

We show that the one-round process can be characterized axiomatically. Let D denote the class of all pairs (S,d) satisfying $S \in Bn \times m$, $d \in S$, and that there exists a $x \in S$ such that $x \gg d$.

Let $f : D \times R^{n \times m} \to R^{n \times m}$ be a function which associates a point of S, denoted $f(S,d,\lambda)$, to each $(S,d) \in D$, and each improvement direction $\lambda \gg 0$. Let us impose the following axioms on the function f.

A1. Weak Pareto optimality. There is no $x \in S$ such that $x \gg f(S,d,\lambda)$.

A2. Invariance under Positive Affine Transformations of Objectives. Let

$$Tx = (T_1x_1, \dots, T_nx_n)$$

be an arbitrary affine transformation such that $T_ix_i = (a_{ij}x_{ij}+b_{ij})_{j=1,\dots,m}$, $a_{ij}>0$, $i\in N$, and let $Lx = (L_1x_1, \dots, L_nx_n)$ be a linear transformation connected with T, i.e. $L_ix_i = (a_{ij}x_{ij})_{j=1,\dots,m}$, $i\in N$. Then $f(TS,Td,L\lambda) = T\, f(S,d,\lambda)$.

A3. Symmetry. For any point $x\in R^{n\times m}$ and for any permutation on N, π, let $\pi^*x=(x_{\pi(1)},\dots,x_{\pi(n)})$. We say that (S,d) is a symmetric game if $d^1=d^2=\dots=d^n$ and $x\in S$ imply that, for every permutation π on N, $\pi^*x\in S$. If (S,d) is a symmetric game and $\lambda_1=\lambda_2=\cdots=\lambda_n$ then $f_1(S,d,\lambda) = f_2(S,d,\lambda) = \cdots = f_n(S,d,\lambda)$.

A4. Restricted Monotonicity. If (S,d) , (T,d) , $\lambda\in Rn\times m$ are such that $u(S,d,\lambda) = u(T,d,\lambda)$ and $S\subset T$ then $f(S,d,\lambda) \le f(T,d,\lambda)$.

Theorem 4. (Bronisz and Krus [1987a]) There is a unique function satisfying the axioms $A1 - A4$. It is the function defined by $f(S,d,\lambda)$ $G(S,d,\lambda)$ for $(S,d)\in D$, $\lambda\in Rn\times m$, $\lambda\gg 0$.

It is easy to note in the unicriterial case, i.e. when $m=1$, that each game (S,d) has a unique utopia point which coincides with the ideal point and the solution $G(S,d,\lambda)$ coincides with the Raiffa solution (see Raiffa [1953], Roth [1979]).

5. A SIMPLIFIED MODEL OF A JOINT DEVELOPMENT PROGRAM

The model relates to two countries (treated as players) which consider realization of a development program. The program requires some amount of resources of various kinds and gives as a result some volume of production. Each country can realize the project independently, or both the countries can decide on a joint development program. Joint program, due to scale effects, can allow for a decrease of required resources at a given production volume or an increase of the production under given resources in comparison to two independent programs.

In the model, two kinds of resources are considered : labor resources and capital assets. Each player is assumed to maximize the obtained production volume and to minimize the resources put in the joint program, but they can differ in preferences among the quantities. The problem consists in a choice of the production scale of the joint program and the sharing of the required resources and of the production volume - which should be agreeable and possibly close to the preferences of the players.

To deal with the case of maximization of objectives only, we assume that each player has given a disposable fund of capital assets $C_i\in R_+$, and a disposable labor resources $L_i\in R_+$ $i=1,2$, and tries to maximize slack variables $sc_i=C_i-c_i$, and $sl_i=L_i-l_i$, where c_i, l_i are the capital and labor resources, respectively, which should be put into the joint project by the $i-th$ player.

The development program, which can be realized in various scales is described by two functions:

$$c : R_+ \to R_+ \quad , \text{ and } \quad l : R_+ \to R_+,$$

where $c(p)$ are capital assets required in the program of the scale or production volume p , $l(p)$ are labor resources required in the program. Assumed shapes of the functions are presented in Figure 2. Similar shapes has been obtained by Bronisz, Krus [1986b] in an example of joint water resources project. In the model, the same forms of the functions are assumed for independently and jointly realized programs, but even in this case the problem is not trivial.

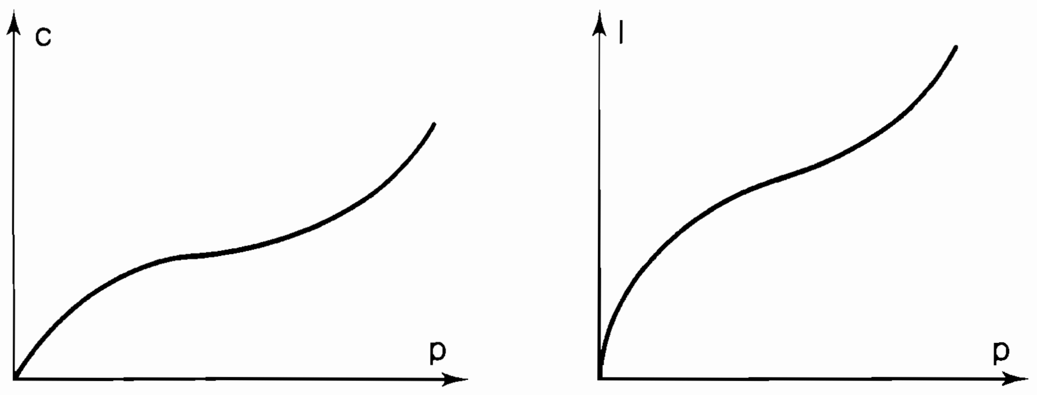


Fig.2 Examples of functions $c(p)$ and $l(p)$.

Each player $i=1,2$ maximizes three objectives: p_i , sc_i , sl_i. The disagreement sets are described by : $S_i \subset R^3_+$, $i=1,2$, where

$$S_i = \{ (p_i,sc_i,sl_i) \in R^3 : c(p_i) \leq \leq C_i - sc_i,\ l(p_i) \leq \leq L_i - sl_i \},$$

The agreement set has the form $S \subset R^{2 \times 3}_+$, where

$$S = \{(p_1,sc_1,sl_1,p_2,sc_2,sl_2) \in R^6 :$$
$$c(p_1+p_2) \leq C_1+C_2-sc_1-sc_2,\ l(p_1+p_2) \leq L_1+L_2-sl_1-sl_2\}.$$

On this example, the negotiation process proposed above has been included into an experimental system of bargaining support.

6. SHORT PROGRAM DESCRIPTION

The experimental system of bargaining support with multiple objectives has been built for the simplified model outlined in the previous section. It can be considered as an illustration of the theoretical results related to the interactive process in multiobjective bargaining problem and its application in support of negotiations.

The system supports two players, each maximizing three objectives, to find an acceptable, cooperative, Pareto optimal solution in an interactive procedure. This is done in two phases:

first: a status quo is derived,

second: a cooperative solution is found.

The status quo is defined as being a composition of the outcomes preferable to players in the noncooperative case. The cooperative solution is found in an iterative process starting from the status quo point.

The first phase deals with the noncooperative case, in which the players look for preferable outcomes assuming independent realizations of the development programs. Each player tests efficient solutions and selects the preferable one. This is done in two steps. In the first step, the player defines reference points in his objective space according to his preferences. The system calculates related efficient solution using the approach described in Section 3. In the second step the player selects the preferable solution among the obtained efficient solutions.

The second phase deals with the cooperative case. It proceeds in a number of iterations. Each iteration consists of two stages:

first: both players define their desired, preferable directions for improvements of outcomes.

second: the system calculates the cooperative outcome on the basis of the status quo point and directions of improvement specified by the players according to the solution concept presented in Section 4.

In the first stage, each player tests directions that improve his outcome and selects a preferable one. This is done in three steps. In the first step, the player defines a step coefficient. In the second step, the player defines directions according to his preferences. Then the system calculates related improved outcomes, assuming the same improvement direction of the counterplayer as in the previous iteration. In the third step, the player selects a preferable direction among the tested directions.

Given the preferable directions of both players, the cooperative outcome is calculated in the second stage. The cooperative outcome is assumed as a new status quo point for the next iteration, and the process is repeated until an efficient cooperative solution is reached.

REFERENCES

Bronisz P., L. Krus, [1986a], "A New Solution of Two-Person Bargaining Games", ZTSW-17-1/86, Report of Systems Research Institute, Polish Academy of Sciences, Warsaw.

Bronisz P., L. Krus, [1986b], "Resource Allocation and Cost Sharing in Common Enterprise. Game Approach", Proceedings of the Polish-DDR Symposium on Nonconventional Optimization, Prace IBS PAN, Warsaw.

Bronisz P., L. Krus, [1987a], "The Raiffa Solution for Multicriterial Bargaining Problems", ZTSW-17-1/87, Report of Systems Research Institute, Polish Academy of Sciences, Warsaw.

Bronisz P., L. Krus, [1987b], "Application of the Rawlsian Maxmin Principle in Multicriteria Decision Support", draft paper, Systems Research Institute, Polish Academy of Sciences, Warsaw.

Bronisz P., L. Krus, A.P. Wierzbicki, [1987], "Towards Interactive Solutions in Bargaining Problem", (forthcoming).

Fandel G., A.P. Wierzbicki, [1985], "A Procedural Selection of Equilibria for Supergames", (private unpublished communication).

Imai H., [1983], "Individual Monotonicity and Lexicographical Maxmin Solution", Econometrica, Vol.51, pp. 389-401.

Kalai E., M. Smorodinsky, [1975], "Other Solutions to Nash's Bargaining Problem", Econometrica, Vol. 43, pp. 513-518.

Nash J.F., [1950], "The Bargaining Problem", Econometrica, Vol. 18, pp. 155-162.

Raiffa H., [1953], "Arbitration Schemes for Generalized Two-Person Games", Annals of Mathematics Studies, No. 28 pp. 361-387, Princeton.

Rawls J., [1971], "A Theory of Justice", Cambridge: Harvard University Press.

Roth A.E., [1979], "Axiomatic Model of Bargaining", Lecture Notes in Economics and Mathematical Systems, Vol. 170, Springer-Verlag, Berlin.

Schmeidler D.,[1969], "The nucleus of a Characteristic Function Game", SIAM Journal on Applied Mathematics, Vol.17, pp. 1163-1170.

Wierzbicki A.P., [1982], "A Mathematical Basis for Satisficing Decision Making", Mathematical Modelling, Vol. 3, pp. 391-405.

Wierzbicki A.P., [1983],"Negotiation and Mediation in Conflicts I: The Role of Mathematical Approaches and Methods", Working Paper WP-83-106, IIASA, Laxenburg; also in H. Chestmat et al., eds: Supplemental Ways to Increase International Stability, Pergamon Press, Oxford, 1983.

A Permutative Scheduling Problem with Limited Resources and Interoperation Constraints

Tomasz Ryś, Wiesław Ziembla

Joint System Research Department,
Institute for Control and Systems Engineering,
Academy of Mining and Metallurgy, Cracow,
Industrial Chemistry Research Institute, Warsaw.

1. Introduction.

Dynamic development in the area of scheduling and operational control of production processes is observed recently. It evolves from natural tendency to realize given task in the optimal way and in the optimal (shortest) time. Actually, the theory of scheduling as an area of operation research is well known but still many problems arise in practical implementation. It results from high computational complexity of those problems (e.g.: Garey, Johnson, 1979; Lenstra, Rinnoy Kan, 1978) - in a computer, they are time and memory consuming. Therefore, the solutions are often obtained by means of heuristic methods and sub-optimal solutions are considered as sufficient instead of optimal solutions.

This work deals with scheduling problems designed in the classification (Grabowski, Smutnicki 1980; Lenstra, Rinnoy Kan 1978) as flow-shop i.e. permutative problems with limited resources and interoperation constraints. These problems arise in production systems (machine-building industry, chemical industry), computer systems and in other service systems. It was the intention of the authors to propose a universal algorithm for solving these problems basing on branch and bound method (Lageweg et al. 1978). This method is particularly recommended in the case of NP-complete problems, which class comprises also scheduling tasks of flow-shop type. The authors assume that the reader is acquainted with scheduling problems in terms of terminology, classification and issues of computational complexity.

2. Problem definition.

In flow-shop production system with limited resources and interoperation constraints a set of tasks may be defined

$$J = \{ J_1, J_2, \ldots, J_n \}$$

while each of above J_i is composed of a sequence of operations

$$< O_{i1}, O_{i2}, \ldots, O_{im} >$$

performed on machines $M_1, M_2, \ldots, M_m$. The time necessary for performing an operation O_{ik} is p_{ik}. In addition, conditions of machine-to-machine transitions are determined. We consider "no wait" (nw), "no store" (ns), "limited waiting" (lw) and "limited temporary storage" (ls) types of constraints (while maximal durations of interoperation waiting and capacities of temporary storages are given). A relation of partial order RT on set J is defined as follows: $(J_i, J_j) \in RT$ implies that task J_j is started on the first machine only when task J_i has been completed at the last machine. It is assumed that the graph of relation RT is acyclic. Moreover requirements for resources ρ_{ikl} in each moment of realization of operation O_{ik} are determined, i.e. in each moment from starting operation O_{ik} to the moment of its completion resources of type l are required in the amount ρ_{ikl}. Maximal availability of a resource l in the moment t is R_{tl}. For each task, a cost function of its realization f_i is defined depending on the time of completion of the task (problems $C_{\max}$ with minimization of performing time required for all the tasks and $T_{\max}$ with minimization of maximal tardiness are most frequently met in literature).

An optimization problem of flow-shop production consists usually in designing such times of starting and completing the tasks which minimize the following optimization criterion:

$$f = \max_{1 \le i \le n} f_i\,(C_{im})$$

where: C_{im} - time of completion of the task i at the last (m-th) machine.

The following assumptions are made:

- a machine can not perform at the same time more than one operation at any moment,
- the realization of a started operation can not be interrupted,
- the sequence of performing tasks on each machine is the same for each task,
- the sequence of performing tasks accords with relation RT,
- the resource and interoperation constraints are satisfied.

It is assumed that functions $f_i(C_{im})$ are nondecreasing in their arguments.

3. The solving algorithm.

3.1. General enumeration scheme.

The optimization problem defined above can be equivalently stated as to find an admissible permutation $\pi \in P(RT)$ (where $P(RT)$ - set of all permutations that accord with RT) for which minimum value of objective function is reached. The following scheme is typically used for generating admissible scheduling permutations. A node by branch t of the search tree defines a partial admissible scheduling

$$\sigma = (\sigma(1)\ ,\sigma(2)\ ,\ \cdots\ ,\sigma(t-1))$$

This means that task $J_{\sigma(i)}$ is performed on all machines as an i-th one. Such a node defines at the same time the set of all permutations which are such continuations of permutation σ that relation RT still holds.

Consider the following denotations:
$J^{\bar{\sigma}}$ - a set of non-scheduled tasks,
$\bar{t}$ - the number of elements in this set,
$\bar{\sigma}$ - the complement of permutation σ.

Thus:

$$J^{\bar{\sigma}} = \{\ J_i \in J \mid i \neq \sigma(k)\ ,\ \ k = 1,2,...,t-1\}$$

$$\bar{t} = \mid J^{\bar{\sigma}} \mid = n - t + 1$$

$$\pi = \sigma\bar{\sigma} = (\sigma(1),\sigma(2),...,\sigma(t-1),\bar{\sigma}(1),\bar{\sigma}(2),...,\bar{\sigma}(\bar{t}))$$

A consecutive node in the search tree is obtained by the choice of task J_i, not scheduled so far, at position t.

$$\sigma i = (\sigma(1),\sigma(2),...,\sigma(t-1),i) \text{ , } where\ J_i \in J^{\bar{\sigma}}.$$

Some of partial solutions obtained in this way can be eliminated due to different partial scheduling and assumed optimization criterion. Let $\sigma\prime$ and $\sigma\prime\prime$ represent some partial scheduling and $\bar{\sigma}\prime$ and $\bar{\sigma}\prime\prime$ be their complements to full permutations, $\sigma\prime\,\bar{\sigma}\prime$, $\sigma\prime\prime\,\bar{\sigma}\prime\prime$ $\in P(RT)$, and let $H(\sigma)$ denote optimal value of objective function for permutation σ. Partial scheduling $\sigma\prime\prime$ dominates over $\sigma\prime$ if for any complement $\bar{\sigma}\prime$ of partial permutation $\sigma\prime$, there exists such complement $\bar{\sigma}\prime\prime$ of permutation $\sigma\prime\prime$ that:

$$H(\sigma\prime\prime\,\sigma'') \leq H(\sigma\prime\,\bar{\sigma}\prime)$$

Rules for rejecting partial solutions in the way presented above result from specific features of the regarded problem. They are called elimination rules. In our algorithm we do not use such rules; instead a partial solution $\sigma\prime$ is eliminated if:

1. The value of objective function. for a partial solution is not less than the value of an upper bound

   $$H(\sigma) \geq UB$$

2. The value of a lower bound in a node of the search tree is not less than the value of the upper bound

   $$LB \geq UB$$

The method of designing upper bound will be presented now and in the next chapter we will deal with the problem of designing lower bounds.

The value of objective function for currently best solution is the upper bound of objective function for all nodes of the search tree until a new permutation $\pi\prime$ is generated, $\pi\prime \in P(RT)$ such that $H(\pi\prime) < UB$

A problem arises how to assign the first upper bound. In many papers (Grabowski, 1979; Lageweg et al., 1978; Smutnicki, 1981) heuristic rules were used (for problems with no resource constraints, in which optimization criterion was of type $C_{\max}$, rules H_1 and H_2, while for criterion $L_{\max}$, rule H_3):

H_1 (Palmer, 1965) Arrange the tasks according to nondecreasing coefficient λ_i, where

$$\lambda_i = \sum_{k=1}^{m} \left[k - \frac{m+1}{2} \right] p_{ik}$$

H_2 (Campbell et al. 1970) According to the optimal permutation for 2-machines problem, where

$$p_{i1} = \sum_{k=1}^{l} p_{ik}$$

and

$$p_{i2} = \sum_{k=l+1}^{M} p_{ik} \text{ , for } l = 1,...,m-1$$

H_3 (Grabowski, 1979) According to a nondecreasing coefficient λ_i, where

$$\lambda_i = \sum_{k=1}^{m} p_{ik}^{m-k+1} + \left| r_i - q_i \right|^m sign(r_i - q_i)$$

where:

$$q_i = d_i - r_i - \sum_{k=1}^{m} p_{ik}$$

d_i - desired completion time of task J_i

r_i - earliest starting time of task J_i

In our algorithm rules H_2 and H_3 are used. It should be noted that the methods described above may be used for improving the value of upper bound in each node of search tree as well.

3.2. Calculation of lower bounds.

For each node t in the search tree while designing a partial schedule σ, a lower bound is to be assigned for all continuations $\sigma\bar{\sigma}\in P(RT)$. Calculation of lower bounds can be based on a relaxation of constraints imposed on process and a modification of optimization criterion. Resource limits, interoperation constraints and assumptions concerning machines' capacity can be subject of relaxation. The assumption that a given machine may at any moment perform at most one task can be replaced by not limiting the number of tasks performed by the machine.

When designing a lower bound *LB*, a similar method as in (Grabowski, 1979; Lageweg et al., 1978; Smutnicki, 1981) will be used. Only two machines M_u and M_v among all machines $(1 \leq u \leq v \leq m)$ will be left unrelaxed. Adjoining relaxed machines will be aggregated to a single machines of unlimited capacity $M_{.u}$, M_{uv}, $M_{v.}$. Thus the problem is reduced to 5 machines, 3 of which that have unlimited capacity i.e. problem of $n|5|P$, $M_{.u}$, M_{uv}, $M_{v.}$ - not of bottleneck type, *lr* (limited resources), $RT|f_{\max}$.

Times of performing operations for these machines can be assigned as follows:

$$p_{i.u} = \max_{1\leq l\leq u} \{ C_{\sigma(t-1)l} + \sum_{k=1}^{u-1} p_{ik} \} \tag{1}$$

$$p_{iuv} = \sum_{k=u+1}^{v-1} p_{ik} \tag{2}$$

$$p_{iv.} = \sum_{k=v+1}^{m} p_{ik} \tag{3}$$

for each $J_i \in J^{\bar{\sigma}}$, where $C_{\sigma(t-1)l}$ is the time of completing the last task in a schedule σ assigned according to the principle of maximal usage of machines and resources (which principle holds for nondecreasing functions f_i). In the case when $u = v$ a 3-machines problem is obtained with one machine of limited capacity - it is a problem of $n|3|P$, $M_{.u}$,$M_{u.}$ not of bottleneck type, *lr*, $RT|f_{\max}$. Then for designing lower bounds resource limits will not be taken under consideration. The computation will be performed for a continuation σ of the problems $n - t|5|P$,M_u,M_{uv},M_v - not of bottleneck type, $RT|f_{\max}$, when the criterion is assigned as follows:

$$\max_{1 \leq i \leq \bar{t}} f_i(C_{\bar{\sigma}(i)v} + p_{iv.})$$

for the 5-machines problem, and

$$\max_{1 \le i \le \bar{t}} \quad f_i(C_{\overline{\sigma}(i)u} + p_{1u.})$$

for the 3-machines problem. Both problems mentioned above are NP-complete (Grabowski, Smutnicki, 1980; Smutnicki, 1981).

Next relaxation consists in eliminating the machine of unlimited capacity. In problems obtained in this way, the cost functions $f_{iu}(C_{\overline{\sigma}(i)v})$, $f_{iuv}(C_{\overline{\sigma}(i)v})$, $f_{iv}(C_{\overline{\sigma}(i)v})$ will be defined as follows:

$$f_{iu}(C_{\overline{\sigma}(i)v}) = f_i(C_{\overline{\sigma}(i)v} + \min_{J_i \in J^{\overline{\sigma}}} p_{i.u}) \tag{4}$$

$$f_{iuv}(C_{\overline{\sigma}(i)v}) = f_i(C_{\overline{\sigma}(i)v} + \min_{J_i \in J^{\overline{\sigma}}} p_{iuv}) \tag{5}$$

$$f_{iv}(C_{\overline{\sigma}(i)v}) = f_i(C_{\overline{\sigma}(i)v} + \min_{J_i \in J^{\overline{\sigma}}} p_{iv.}) \tag{6}$$

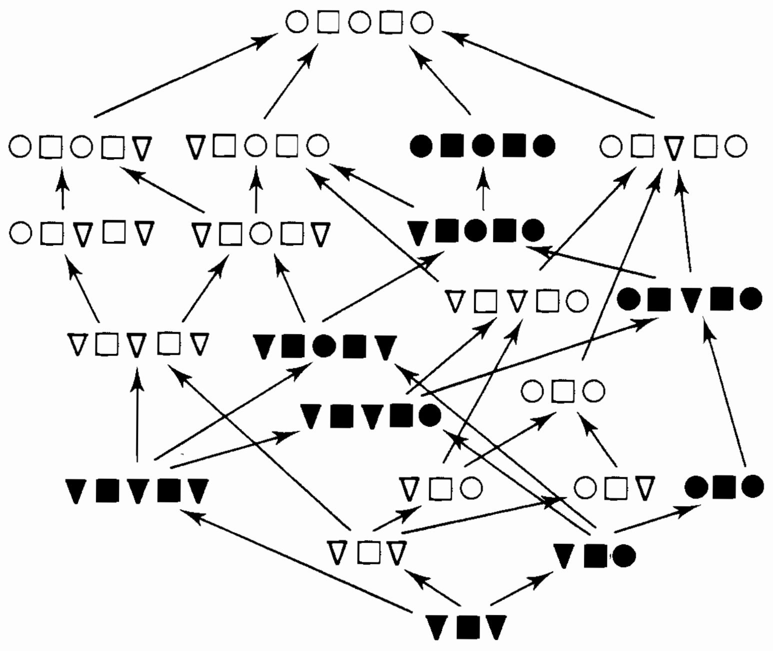


Figure 1. Graph of domination for Ω sequences.

The problems with cost functions as in (4)- (6) may be relaxed further on by eliminating the cost according to the formula:

$$f_*(t) = \min_{J_i \in J^{\overline{\sigma}}} f_i(t) \quad , \quad C_{\sigma(i-1)v} \le t < \infty \tag{7}$$

Instead of the problem with objective function as in (7), the problem with criterion $C_{\max}$ can be solved and then the cost equal to $f_*(C_{\max})$ can be assumed.

We shall classify now the methods of computing LB. Each of those methods may be characterized by a sequence Ω composed of three or five symbols from the set $\{\nabla, PI, \bigcirc\}$, or the set $\{\tilde{\nabla}, \tilde{\Pi}, \tilde{\bigcirc}\}$, where

Π - stands for a machine with limited capacity,

$\bigcirc$ - stands for a machine with unlimited capacity,

∇ - stands for a machine with unlimited capacity, eliminated according to the formula (4) - (6)

$\tilde{\nabla},\tilde{\Pi},\tilde{\bigcirc}$ – stand for problems, in which the costs were eliminated according to (7).

Lower bound $LB(u,v,\Omega,RT_{\bar{\sigma}})$ is the value of optimal solution for a problem with machines M_u, M_v, of Π type and $\bigcirc$ type with compensation of eliminated machine of ∇ - type (eventually with eliminated cost). As $LB(u,v,\Omega,RT_{\bar{\sigma}})$ is the lower bound for any of the pairs (u,v), $1 \leq u \leq v \leq m$, we may define:

$$LB\ (\Omega,RT_{\bar{\sigma}}) \;=\; \max_{1\,\leq\, u\,\leq\, v\,\leq\, m} LB(u,v,\Omega,RT_{\bar{\sigma}})$$

A relation of domination may be defined on the set of sequences Ω. The graph of that relation is showed on Fig.1. An arch $(\Omega,\Omega\prime)$ in that picture means that sequence $\Omega\prime$ dominates over sequence Ω i.e. for each pair (u,v) another pair $(u\prime,v\prime)$ may be found such that the following inequality holds:

$$LB(u,v,\Omega,RT_{\bar{\sigma}} \leq LB(u\prime,v\prime,\Omega\prime,RT_{\bar{\sigma}})$$

The relation of domination is transitive but not all sequences are comparable (in Fig.1 - the sequences not connected directly or by other intermediary sequences).

Only 6 from resulting methods of LB calculation can be executed by means of polynomial time algorithms (e.g.: Grabowski, 1979; Lageweg at al., 1978). These methods will be described here and compared with other methods occurring in the literature. Symmetric sequences which are not marked on Fig.1 (due to equivalence of scheduling problems) will not be considered.

(A) $(\tilde{\nabla}\tilde{\Pi}\tilde{\nabla})$

Eliminating the first and the last machine and the costs, problem of minimization $C_{\max}$ will be obtained for the M_u machine i.e., the problem has the form $n{-}t|1|RT_{\bar{\sigma}}|C_{\max}$

$$LB(u,u(\tilde{\nabla}\tilde{\Pi}\tilde{\nabla}),RT_{\bar{\sigma}}) \;=\; f_*\,\{\sum_{J_i\in J^{\bar{\sigma}}} p_{iu} + \min_{J_i,J_j\in J^{\bar{\sigma}},i\neq j} \quad (p_{i.u} + p_{ju.})\}$$

It is the least stringent lower bound from the mentioned above. For $f_{\max} = C_{\max}$ it was used together with another bound, as described inter alia in (Grabowski, 1979; Lageweg at al., 1978). Computational complexity of the algorithm is $O(n{-}t)$.

(B) $(\tilde{\nabla},\tilde{\Pi},\tilde{\bigcirc})$

Eliminating the costs and machine $M_{.u}$ the problem of maximal tardiness in performing the tasks with given times of completion p_{iu} is obtained.

$$LB(u,u,(\nabla\Pi\bigcirc),RT_{\bar{\sigma}} \;=\; f_*\{LB^*(u,u,(\nabla\Pi\bigcirc),RT_{\bar{\sigma}}) + \min_{J_i\in J^{\bar{\sigma}}} \quad p_{i.u}\}$$

The value of $LB^*(u,u,(\tilde{\nabla}\tilde{\Pi}\tilde{\bigcirc}),\ RT_{\bar{\sigma}})$ can be computed using an algorithm which requires $O(n{-}t)^2$ iterations (Lawler, 1964).

$(\tilde{\bigcirc}\tilde{\Pi}\tilde{\nabla})$

The problem is analogous to (B). By omitting the last machine of unlimited capacity and modifying the criterion according to (7) the problem of minimization of $C_{\max}$ is obtained on one machine with nonzero times of starting the tasks: $n{-}t|1|r_i,RT_{\bar{\sigma}}|C_{\max}$. The optimal value of the objective function for that problem is equal to

$LB^*(u,u,(\tilde{\bigcirc}\tilde{\Pi}\tilde{\nabla}),RT_{\bar\sigma})$ and may be computed by means of a polynomial time algorithm of complexity $O(n-t)^2$ (Lawler, 1964). Thus

$$LB(u,u,(\bigcirc\Pi\nabla),RT_{\bar\sigma} = f_*\{LB^*(u,u,(\bigcirc\Pi\nabla),RT_{\bar\sigma}) + \min_{J_i\in J^{\bar\sigma}} p_{iu.}\}$$

(C) $(\nabla\ \Pi\ \nabla\ )$

By omitting the first and the last machine of unlimited capacity the scheduling problem $n-t|1|RT_{\bar\sigma}|f_{\max}$ will be obtained, where:

$$f_{\max} = \max_{J_i\in J^{\bar\sigma}} f_i\{C_{\bar\sigma(i)u} + \min_{J_i,J_j\in J^{\bar\sigma},i\neq j}(p_{j.u} + p_{iu.})\}$$

$LB^*(u,u,(\nabla\Pi\nabla),RT_{\bar\sigma}) = f_{\max}$ and may be computed by means of an algorithm described in (Smutnicki, 1981) in $O(n-t)^2$ iterations.

(D) $(\nabla\ \Pi\ \bigcirc\ )$

By omitting the first machine with unlimited capacity, the scheduling problem $n-t\,|\,1\,|\,RT_{\bar\sigma}|\,f_{\max}$ will be obtained, where

$$f_{\max} = \max_{J_i\in J^{\bar\sigma}} f_i[C_{\bar\sigma(i)u} + p_{iu.} + \min_{J_i\in J^{\bar\sigma}} p_{i.u}]$$

$LB(u,u,(\nabla\Pi\bigcirc),RT_{\bar\sigma}){=}f_{\max}$ may be computed in $O(n-t)^2$ iterations (Smutnicki, 1981).

All other problems are NP-complete. Two of them for $RT_{\bar\sigma} = \phi$ are polynomial problems; these are $\Omega = (\nabla\tilde{\Pi}\tilde{\bigcirc}\tilde{\Pi}\tilde{\nabla})$ and for $\Omega = (\tilde{\nabla}\tilde{\Pi}\tilde{\nabla}\tilde{\Pi}\tilde{\nabla})$

(E) $(\tilde{\nabla}\tilde{\Pi}\tilde{\nabla}\tilde{\Pi}\tilde{\nabla})$

By eliminating machines $M_{.u}$, M_{uv}, $M_{v.}$ and modifying the costs (7), $LB^*(u,v,(\nabla\Pi\nabla\Pi\nabla))$ can be computed with Johnson's algorithm in $O(n-t)ln(n-t)$ iterations.

$$LB(u,v,(\nabla\Pi\nabla\Pi\nabla)) =$$
$$=f_*\{LB^*(u,v,(\nabla\Pi\nabla\Pi\nabla)) + \min_{J_i\in J^{\bar\sigma}}(p_{iuv} + \min_{J_i,J_j\in J^{\bar\sigma},\ i\neq j}(p_{i.u} + p_{jv.})\}$$

(F) $(\ \tilde{\nabla}\tilde{\Pi}\tilde{\bigcirc}\tilde{\Pi}\tilde{\nabla})$

While eliminating machines $M_{.u}$ and $M_{v.}$ and modifying costs in (7), the problem of minimization of $C_{\max}$ is obtained which also can be solved by means of Johnson's algorithm by changing the times of operating on machines M_u and M_v according to:

$$p'_{iu} = p_{i.u} + p_{iuv}$$

$$p'_{iv} = p_{iv.} + p_{iuv}$$

for $J_i\in J^{\bar\sigma}$. In that way $LB^*(u,u,(\tilde{\nabla}\tilde{\Pi}\tilde{\bigcirc}\tilde{\Pi}\tilde{\nabla}))$ is obtained, and:

$$LB(u,v,(\tilde{\nabla}\tilde{\Pi}\tilde{\bigcirc}\tilde{\Pi}\tilde{\nabla})) = f_*\{LB^*(u,v,(\nabla\Pi\bigcirc\Pi\nabla)) + \min_{J_i,J_j\in J^{\bar\sigma},\ i\neq j}(p_{i.u} + p_{jv.})\}$$

Such a criterion for $f_{\max} = C_{\max}$ was considered in (Lageweg et al., 1978) and the criterion:

$$LB_1^* = C_{\sigma(t)u} + \max_{J_i\in J^{\bar\sigma}}\{p_{iu} + p_{iuv} + p_{iv} + \sum_{J_j\in J^{\bar\sigma}\setminus\{J_i\}} min(p_{ju},p_{jm})\}$$

used in (Mc Mahon et al., 1975) is an approximation of $LB(n,m,(\tilde{\nabla}\tilde{\bigcirc}\tilde{\nabla}))$ not surpassing that value. In computations following lower bound [9] can also be used:

$$LB_1\,(McM) \;=\; f_*(\max_{1\le u\le m} LB_1^*)$$

and

(G) (McM)

$$LB_2(McM) \;=\; max\,\{LB_1(McM), LB(\nabla\Pi\nabla)\}$$

All other problems defined by the sequences from Fig.1 are *NP-complete*, and as lower bounds are computed in each node of the search tree, such methods will not be used in proposed algorithms.

4. Example.

As an example of using the methods presented above a problem of designing the optimal schedule and resource allocation in medical service system will be presented. Such an example was chosen due to its illustrative character. The data are easy to obtain and the description is clear. Therefore, no detailed description or identification of industrial technology was required. It is obvious that this system can be easily adapted for industrial conditions - it is sufficient to replace operation rooms by machines and prescribed treatments by production tasks. In fact our team has worked at a real chemical production problem. However, due to an agreement with the client, it was not released for publication.

Presented example concerns a hospital of sanatorium type - the computer system created for solving the optimization problem is called *SANATORIUM*.

4.1. General description of the problem.

The scheduling problem presented here may be briefly described as follows:

> For a given sanatorium turn (a group of patients) a treatment for each patient is designed, a set of cabins (operation rooms) is fixed as well as the set of operators working in these rooms. Daily schedules for patients, rooms and operators are to be found for each day of patients' stay in the sanatorium.

Following assumptions were made when designing the system:

a) for each patient, the term of his arrival and leaving the sanatorium as well as the treatment (number of operations) are defined,

b) operations of the same type are to be distributed uniformly during the patient's stay,

c) one patient can be submitted only to a limited number of operations per day,

d) for each treatment, a relation may exist imposing a sequence of realization of particular operations,

e) the treatment is to be started according to the assumption of uniform usage of resources,

f) a given number of patients may be submitted to operation in one room at a given moment,

g) each room is attended by an operator - one operator may attend more than one cabin but not at the same time,

h) time of operations as well as the time necessary for the operator to stay with the patients are fixed,

i) an operation can not be interrupted,

j) only the patients of the same sex may be treated in the same room at the same moment,

k) the schedule is to satisfy the condition of minimization of daily usage of the rooms (machines).

4.2. Implementation.

The computer system is functionally divided into three modules. First of them is a problem generator devised for storage and correction of input data, two others are solver modules that solve the presented scheduling problem.

The first module creates a data base concerning the resources i.e. cabins and operators as well as defines the problem by creating the actual data base concerning the patients and prescribed treatments.

Division of the system into two other modules results from assumed the method of solving that consists in a division of the task into two parts concerning two different horizons of scheduling, with different units of time (minutes and days).

In the first part of solver module, the schedule of treatment for a given day is worked out so as to satisfy the assumptions a,b,c,d,e, (see 4.1). This problem is solved by a heuristic algorithm of pseudo-polynomial complexity of computations.

In the second part of solver module, exact times are to be determined for starting and finishing the treatment and assignment of machines rooms, and operators that are required for consecutive days of sanatorium work. Solving this problem is based on the method described in chapter 3; the algorithm is of NP-complete computational complexity.

The system *SANATORIUM* was written in *TURBO-PASCAL* (Borland International) for IBM-XT computer and includes graphical display of results.

5. Concluding remarks.

The work presented here has an experimental character (especially regarding the software). It was aimed at a numerical test of chosen heuristic and optimization algorithms that are used for solving a class of scheduling problems, namely permutative problems with limited resources and interoperation constraints. This type of problems was chosen because they frequently occur in real life.

The intention of the authors is to create a software library devised for a personal computer. The structure of programs is unified and they could be applied to a wide range of problems. It seems that this task can be realized due to the fact that scheduling problems are well systematized. The class of problems being defined, a user can chose appropriate procedures and construct (or select) an interface for this particular problem. Introductory phase of preparing any problem consists always in input data definition or updating and the final phase is the output of results mostly in the form of bar-charts. In both phases, it is possible to use some standard software tools - e.g. *dBase III* or *INFORMIX* in the introductory phase and *Turbo Graphics* or other integrated packages in the final phase. Therefore, it is possible to prepare a standard shell for an interface with the user.

In the field of theoretical research, a number of important problems are still unsolved or even not well formulated. The authors want to focus their future interest on the following problems:

- the analysis of other optimization criteria such as e.g. cost criteria in connection with the time of starting and ending the tasks, as well as the use of limited resources and limited quantity of machines. Solving these problems could help in a new application of the approach presented here, namely, in the design of optimal production structures;
- various issues of multiobjective analysis in process scheduling, interactive problem definition and selection between efficient alternative solutions.

REFERENCES.

1. Blazewicz J. (1979). *Computational Complexity of Algorithms and Scheduling Problems.,* Technical University of Poznan, series: Essays, No **104** , Poznan.
2. Campbell H.H., Dudek R.A., Smith M.L. (1970). *A Heuristic Algorithm for the n Job m Machine Sequencing Problem.* Management Sci. Vol. **16** , pp. B 630-637.
3. Garey M.R., Johnson D.S., (1979). *Computer and Intractability: A Guide to the Theory of NP-Completeness.* ed. W.H. Freeman, San Francisco.
4. Grabowski J.(1979). *General Problems of Optimal Scheduling in Discrete Production Systems.* Scientific Bulletin of Institute for Cybernetics, Technical University of Wroclaw, Monographies No 50, Wroclaw.
5. Grabowski J., Smutnicki C., (1980). *Scheduling Problems: Classification and Computational Complexity of the Algorithms.* Report PRE 141/890, Technical University of Wroclaw.
6. Lageweg B.J., Lenstra J.K., Rinnoy Kan A.H.G. (1978). *A General Bounding Scheme for the Permutation Flow-Shop Problem.* Opus Res., Vol. **26** , pp.33-67.
7. Lawler E.L. (1964). *On Scheduling Problems with Deferal Costs.* Management Science, Vol. **11** , pp.280-288.
8. Lenstra J.K., Rinnoy Kan A.H.G. (1978). *Complexity of Scheduling under Precedence Constraints.* Opus Res., Vol. **26** , pp. 22-35.
9. McMahon G.B., Florian M. (1975). *On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness.* Opus Res., Vol **23** , pp. 475-482.
10. Palmer D.S. (1965). *Sequencing Jobs through a Multi-Stage Process in the Minimum Total Time - a Quick Method of Obtaining a Near-Optimum.* Optimal Res. Quart., Vol. **16** , pp.101-107.
11. Slowinski R. (1980) *Algorithms for Control of Resource Allocation of Different Types in a Complex of Operations.* Technical University of Poznan, series: Essays No **114** , Poznan.
12. Smutnicki C. (1981) *Problem of Operation Sequence Optimization in Discrete Production Systems.* PhD Thesis, Technical University of Wroclaw.

Multiobjective Evaluation of Industrial Structures MIDA application to the Case of Chemical Industry

Maciej Zebrowski

Joint Systems Research Department
Institute for Control and Systems Engineering
Academy of Mining and Metallurgy, Cracow
Industrial Chemistry Research Institute, Warsaw.

1. Introduction

This paper reports the results of continuation of research in the area of decision support for industrial development strategy (see Dobrowolski and Zebrowski, 1985).

The research was sponsored by the Polish Government Energy Program, but since the sponsor willingly accepted collaboration with IIASA specifically with the Study on Theory, Software and Testing Examples for Decision Support Systems, the concepts originated in the initial project were further developed and as such presented here.

The first part of the paper conveys a theoretical background, the core of which is a substitution model based on the concept of a Production Distribution Area (PDA, see Dobrowolski et al.. 1979, 1984). This model describes general properties of an industrial branch when a development process is investigated. The model discussed here deals with the phenomena of substitution. Three types of substitution are considered:

- substitution of feedstock,
- substitution of final products,
- substitution of technologies.

Physically, it is the substitution of technologies - old by new - that enables the two other types of substitution to take place. The goal of the substitution model based on PDA concept is to provide a decision maker via decision support system (DSS) that constitutes a practical tool for the control of substitution processes. The practical usefulness of the proposed model comes from the fact that some intuitively obvious and pragmatically applied industrial rules of substitution analysis are formally embedded in this model.

An application of the model is presented. It is based on the case of feedstock and fuels PDA (PDA-FF). Purposefully, the same area was chosen as in the phase I of the project (Dobrowolski, Zebrowski, 1985), although the numerical data used are different and modified. For obvious reasons, numerical results do not precisely correspond to those used in the originally sponsored program.

2. Formal framework for the analysis.

2.1. The model.

An industrial branch or production distribution area of chemical industry can be described in terms of a PDA model (Dobrowolski et al. 1979, 1984), regarded here as a model of technological network. This model can be developed into a substitution model:

$$(B-A)z = \begin{bmatrix} y^I \\ y^O \end{bmatrix}$$

where:

(B - A) - matrix of technological links seen from the production level of particular installations,

y^I - input resources,

y^O - output resources.

Only two types of resources are considered - consumed or utilized by PDA and those that can be obtained from the PDA. The resources of y^I and y^O type could be chemical raw materials, semi-products and products as well as water, technological energy and investment. The set of solutions of this model designates, a maximum range of substitution dependent on the repertoire of technologies considered in the network. In real life cases, the range of substitution is narrowed by additional restrictions imposed on y^I resources (availability) and y^O resources (salability and/or demand patterns) and the production level (production capacities). Such constraints may either represent actual information (as a result of identification) or also be postulated in order to obtain additional information on the substitution properties of the network.

We specify now some general assumptions that enable further formalization of the control of substitution process and thus will be useful for carrying out the pre-decision analysis for an object as complex as PDA-FF.

Assumption 1. Effective-feasible solution.

The solutions of the substitution equation that satisfy specific additional constraints are considered as feasible, whereas such feasible solutions that:

$$y^O \to \max$$
$$y^I \to \min$$

are regarded as effective-feasible. The assumption that we might restrict the analysis to effective-feasible solutions limits the substitution range. Therefore at this stage of considerations we restrict the substitution analysis to such cases which are most effective. As it will be explained below, effectiveness can be formally expressed in a number of other ways.

Although this assumption may seem to be intuitively obvious, it is practically difficult to satisfy it if the number of y^I and y^O flows is large. This obstacle may be overcome by following a modified assumption instead:

Assumption 2. Aggregate effectiveness.

Instead of optimizing each component of flow of the resources y^I and y^0 a smaller number of attributes can be defined to characterize the resources y^I and y^O thus enabling for their aggregation. These aggregates will be used for representation of the area resources. The price of a product or a raw material, or their heating values may be regarded as examples of such attributes.

We assume that it is possible to characterize the input resources by n_i aggregate quantitative attributes that will be minimized and the output resources by n_0 attributes that will be maximized:

$$a^I(y^I) \in R^{n_i}$$
$$a^0(y^0) \in R^{n_0}$$

where R^{n_i}, R^{n_0} - the corresponding spaces of real vectors.

Aggregate attributes can be also obtained through subtraction of other attributes provided those attributes have the same physical meaning and formal rules of subtraction are followed. Economic effectiveness calculated as a difference between value of the total sale of PDA and the total cost of production in a particular state of the flows of resources, or the total energy balance of the PDA calculated as the difference between total energy consumed and total energy obtained (in products) may serve as examples of such aggregates.

We shall thus assume that:

- the attributes of resources of both I and O type can be quantified in positive numbers only,
- attributes of I,O resources are classified accordingly as consumed (minimized) and obtained (maximized),
- the difference between two attributes of O-I type can be defined in cases of attributes of the same nature, whereas for O>I, the result of subtraction O-I brings gain to the system while for O<I the result of I-O means loss.

Thus, the Assumption 1 is modified to the following:

$$a^O \rightarrow \max$$
$$a^I \rightarrow \min$$

and correspondingly for attributes that allow from subtraction:

$$a^O - a^I \rightarrow \max$$
$$a^I - a^O \rightarrow \min$$

Three other remarks are related to the above principles of aggregation:

1) The number of attributes a^I, a^0 decreases with aggregation, hence satisfying Assumption 2 is easier than satisfying Assumption 1.
2) The substitution may be limited also by imposing constraints on aggregate attributes.
3) Substitution may take place both on the side of input aggregates and on the side of output aggregates. Should such a substitution occur, each state of the PDA can be characterized by different effectiveness of the transition from inputs to outputs. Therefore, the concept of effectiveness is fundamental for the model and it will be explained further in detail.

Assumption 3. Equivalence and effectiveness.

Either the resources or their aggregate attributes are considered as hierarchically equivalent. It results from the fact that in Assumptions 1 or 2 the optimization applies equivalently to all resource components or individual attributes, which corresponds to multiobjective optimization in Pareto sense.

A hierarchy of resources or attributes can be introduced first when entering the sphere of the formulation of a development thesis (see Dobrowolski, Zebrowski, 1985). Such a formulation, however, goes beyond the problem of substitution.

Within the formal framework of the substitution model one should solve the problem of how to represent the effectiveness of substitution. The effectiveness rules are to provide a formal interface between substitution and preferences assigned by a decision maker when formulating a development thesis.

Therefore let us analyze again Assumption 1. As it was mentioned, it comprises the operations min, max and connects de facto substitution with its effectiveness. Observe that in order to consider a substitution of that kind at least two resources of the same type and one of the opposite must be considered - i.e. two output resources and one input resource or vice versa. For instance, raw materials substitution usually means a substitution of a raw material 1 for a raw material 2; obviously the effectiveness of such an operation must be related to the product or products obtained.

The problem arises how to express effectiveness of obtaining a given product y^0 from raw materials y_1^I or y_2^I. It may be solved by introducing the concept of effectiveness ratios where:

$$\frac{y^0}{y^{I_1}}, \quad \frac{y^0}{y^{I_2}}, \quad \to \ \max$$

denotes the effectiveness of obtaining y_1^I from y_2^I, respectively, and

$$\frac{y_1^I}{y^0}, \quad \frac{y_2^I}{y^0}, \quad \to \ \min$$

denote the effectiveness of consuming y_1^I from y_2^I, respectively, and obtain y^0. Therefore, they may be regarded as equivalent inverse intensity ratios. The same applies also for the aggregate attributes:

$$\frac{a^I}{a^O} \ \to min$$

$$\frac{a^O}{a^I} \ \to max$$

while, in the above example, the attribute a^I might jointly characterize the case of resources y_1^I, y_2^I. The above ratios are a natural generalization and provide very practical effectiveness indicators. The same applies to ratios built on differential aggregates; a good example of such combined ratio may be:

$$\frac{a_1^O \ - \ a_1^I}{a_2^I \ - \ a_2^O} \ \to \ \max$$

where denominator expresses resources consumed (such as net energy balance) while the numerator expresses resources obtained (such as added value).

We can conclude this discussion by following remarks:

- Assumptions 1, 2 and 3 provide for a simple analysis of substitution of elementary resources (of I,O type) based on a chosen set of intensity ratios.
- The area of substitution is limited by the assumptions of optimization of chosen indicators.

When formulating a development thesis in the multiobjective industrial development analysis (MIDA) for a given PDA, the key issue is that of a critical resource. The decision maker may, for the sake of the formulation of the development thesis, assign a status of critical resource to any of the resources of O or I type as well as their aggregates (Dobrowolski et al. 1984) According to the MIDA methodology, a critical resource is defined as single resource or an aggregate attribute which obtained this status through the decision maker's choice, as it was considered by him as crucial (critical) for the implementation of the development thesis.

In the context of the substitution model considered here, the status of a critical resource could be assigned e.g. to the resource that was chosen in order to examine possibility and effectiveness of its substitution by another resource. A PDA model is then used to enable the analysis of substitution effectiveness as a part of analysis resulting from development thesis.

2.2. Towards decision support system tool for multiobjective evaluation of industrial development strategy.

Before discussing a real life example let us make some additional observations. The intensity ratios built on the input and output aggregates when used as criteria for multiobjective problem, attain their extreme values in vertices of Pareto optimal surface. We have proposed (Dobrowolski, Zebrowski, 1984) to define this area denoted by the efficient vertices in the criteria space as Attainable Performance Area (APA).

Let us remind also that a technological repertoire or a set of technologies is naturally divided into two subsets. The first represents existing technologies while the second - potentially available technologies that require additional investment. Therefore two types of respective industrial structures must be distinguished: the first, that assures the attainability of current production goals by existing technologies and the second that comprises those technologies that can be attained by means of investment.

The formal framework presented above provides a good point of departure for devising a DSS tool for the evaluation of industrial development strategies in terms of intensity ratios which are practically used and well interpretable by decision makers. This approach becomes then naturally a part of MIDA methodology (Dobrowolski et al. 1985).

To make this extension applicable, following methodological steps are to be considered. The starting one is the evaluation of APA which is to be represented by selected set of intensity ratios and which is to be agreed with the decision maker as a complete set for a given stage of analysis. All the necessary conditions for the evaluation of the **existing** state in terms of its performance such as production goal as well as a set of other defining conditions (see an example below and also (Dobrowolski, Zebrowski, 1985)) are to be known. Another distinguished structure is the one that may be called an **ultimate** or **goal** structure. This is defined by a similar set of parameter and conditions as those describing the existing structure with the fundamental difference that they correspond to the aspirations of the decision maker.

The above provides a first step in the analysis. Next comes the analysis of Attainable Performance Area (APA). There is no unique way of analyzing this type of Pareto set. It should be adjusted to the particular needs of a decision maker when solving a particular case; however, the necessary information, that is, the efficient vertices must be designated. Then a feasible method for acquiring knowledge about the properties of APA is to be chosen. APA, let us remind, reflects properties of the available repertoire of technologies assembled into alternative industrial structures. These structures represent potential alternatives for industrial development strategy that are Pareto optimal with respect to intensity ratios (criteria) accepted for their evaluation. A practical feasible indication for devising such a method is to define some cross sections of APA, parameterized by a ratio representing a decision variable which is a driving force for the development. This could be represented for example, by intensity ratio built on investment. Obviously such a ratio is indispensable when a set of intensity ratios is being considered by a decision maker.

Next comes, as a natural step, the evaluation of various development trajectories expressed in terms of horizon of implementation of alternative development strategies. This gives to the decision maker an idea about the dynamics of achieving the goal structure as represented by a completion of a chosen industrial development strategy. From that stems a natural mode of parameterizing an implementation horizon by the period of investment return.

Having all the above information, the decision maker can also obtain the resulting values of critical resources which are to be consumed or can be obtained with respective strategies. This knowledge compared with corresponding data from the analysis performed for the existing state is a very practical method of evaluating substitution of critical resources due to the substitution of technologies within respective industrial structures. All the above considerations will be now illustrated by means of a practical industrial example.

3. Example of multiobjective evaluation of IDS - Industrial Development Strategy

3.1. Feedstocks and Fuels PDA - an example of substitution model analysis.

Based on the formal background presented above, the problem may be referred to a particular PDA. We choose an area that was described before in many works (Dobrowolski, Zebrowski, 1985), and may serve as a good illustration.

The necessary information about the PDA may be ordered as follows:

1 Technological repertoire is given (described by respective parameters and production capacities).

2. Substitution of critical resources that are primary energy carriers (a basic feedstock) is to be evaluated. These are:
   - crude oil,
   - hard coal,
   - lignite,
   - natural gas.

3. It is assumed that the range of substitution is limited through demand for the output critical resources imposed by the strategy of higher level (macroeconomic production goals). This comprises following products:
   - ethylene,
   - benzene,
   - methanol,
   - diesel oil,
   - carbonizate from coal,
   - carbonizate from lignite.

   This demand describes a goal production level while existing production levels are also defined.

4. The following values are assumed to be the attributes that enable for aggregation:
   - prices of I, O resources (including technological energy, that is, steam and electric energy),
   - heating values of raw materials and products existing in the PDA.
5. The following aggregates are defined:
   - IE - input energy - technological energy used by PDA and energy contained in raw materials,
   - OE - output energy - energy gained in the products,
   - II - input investment,
   - IV - input value value of purchase of raw materials and energy,
   - OV - output value - value of sale of the products.

   Additionally a differential aggregate can be formulated:

   AV = OV - IV - which is interpreted as the Added Value.
6. Effectiveness ratios expressed in terms of the defined aggregates are as follows:
   - OE/IE - Energy Conversion Efficiency
   - OV/IV - economic effectiveness ratio,
   - AV/II - effectiveness of investment, which is equivalent to RI - Return of Investment.

Let us consider the multiobjective problem based on the following three effectiveness ratios:

$OE/IE \rightarrow \max$;

$OV/IV \rightarrow \max$;

$AV/II \rightarrow \max$;

A set of states of the PDA-FF model defining possible substitution will be a solution to this problem. Observe that:

- Assumptions 1 and 2 are satisfied for the representation of the assumed attributes;
- The problem is formulated in the Pareto sense (equivalence of criteria) and, therefore, Assumptions 3 is also satisfied;
- Effectiveness of substitution is considered in the sense of assumed criteria.

Therefore, the set of solutions comprises only such alternatives of substitution, for which the three effectiveness ratios have greater value than the rejected ones. Such set of solutions can be presented as a surface in the criteria space, which will be shown in the next paragraph. In such a way, the substitution area for all input and output resources existing in PDA-FF was obtained. It can be expressed in terms of assumed effectiveness criteria and in absolute values of critical resources gained or consumed. These two types of parameters describe industrial structure rather adequately from point of view of a decision maker since they provide information about the scale or level of operation and its intensity (Skocz, Zebrowski, 1986). At the same time, this information serves best for the comparison of various development alternatives. The completeness of ratios as well as list of critical resources taken into consideration may vary from case to case but the principle remains unchanged.

In the case discussed here the surface representing the set of solutions of the problem of optimal substitution is spanned by three vertices of this Pareto surface. They may be interpreted as follows:

OE/IE (max)denotes the state of PDA-FF with the greatest energetical effectiveness.
OV/IV (max)denotes the state of PDA-FF with the greatest economical effectiveness.
AV/II (max) denotes the state of PDA-FF with the best investment effectiveness.

3.2. Experiments with the model.

Experiments with a PDA-FF model are summarized in Tables 1-4 and visualized by Fig.1.

APA or Attainable Performance Area of PDA-FF is shown in Table 1. This gives an idea of the substitution flexibility of the technological repertoire as expressed in terms of the three selected ratios. Another important information given in that table it the cost of investment (II) which is to be involved to attain a structure corresponding to respective vertices. The difference in calculated values of II is more than 10 fold, while the corresponding ratios are not so dramatically different however the experienced decision maker knows that even several per cent difference in intensity ratios should not be underestimated. Table 4 - which is a summary table - provides a relation between the existing industrial structure and APA. It shows also the consumption of critical resources for respective industrial structures and their substitution as related to their intensity ratios.

Table 2 contains cross sections of APA as visualized on Fig.1. Cross sections are defined by the fixed value of the rate of return on investment, namely AV/II.

With the above knowledge it is useful to perform the following evaluation of APA (illustrated by Table 3.). This evaluation is done in order to figure out the implementation horizon and the rate of return on investment as parameterized by a return period. This is a better methodological approach than the one based only on a pre-defined return of investment ratio.

The reader is invited to get a closer insight by evaluating the data himself. It may be added that - with such simple calculations - a good feeling of dynamic properties of the evaluated strategies can be achieved by a decision maker. This is meant in the sense of feasibility of a given strategy in time related to the indispensable investment level. Naturally the value of the information on development dynamics lies rather in the comparison of various industrial development strategies than in the calculation of static indices. Without a preliminary comparison of basic indices, however, the choice of alternatives for further investigation cannot be carried out.

Finally, Table 4 provides information which summarizes the results of the analysis.

4. Conclusions.

We can conclude that multiobjective industrial development analysis (MIDA), owing to the concepts presented in this paper, is extended with the new tool of decision support type. The simple theoretical framework provides a good communication means between the system and a decision maker.

The concept of APA utilized in this paper proved to be very useful owing to its practical and clear interpretation. The progress described here demanded, however, a very substantial effort on the side of software implementation, specifically a linear fractional programming solver developed by Dr G. Dobrowolski, which will be implemented and embedded in the MIDA system. Similarly, an 'Optimist' software package implemented by T. Rys was an important software tool enabling MIDA to be used for the type of analysis described in this paper.

V, I, E	vertices
VE	projection of Pareto surface on plane of economic efficiency (OV/IV) and energy conversion efficiency (OE/IE)
VI-1, EI-1	lines corresponding to constant value of return of investment
	VI-2,E
	VI-3, VE-3
	VI-4, VE-4

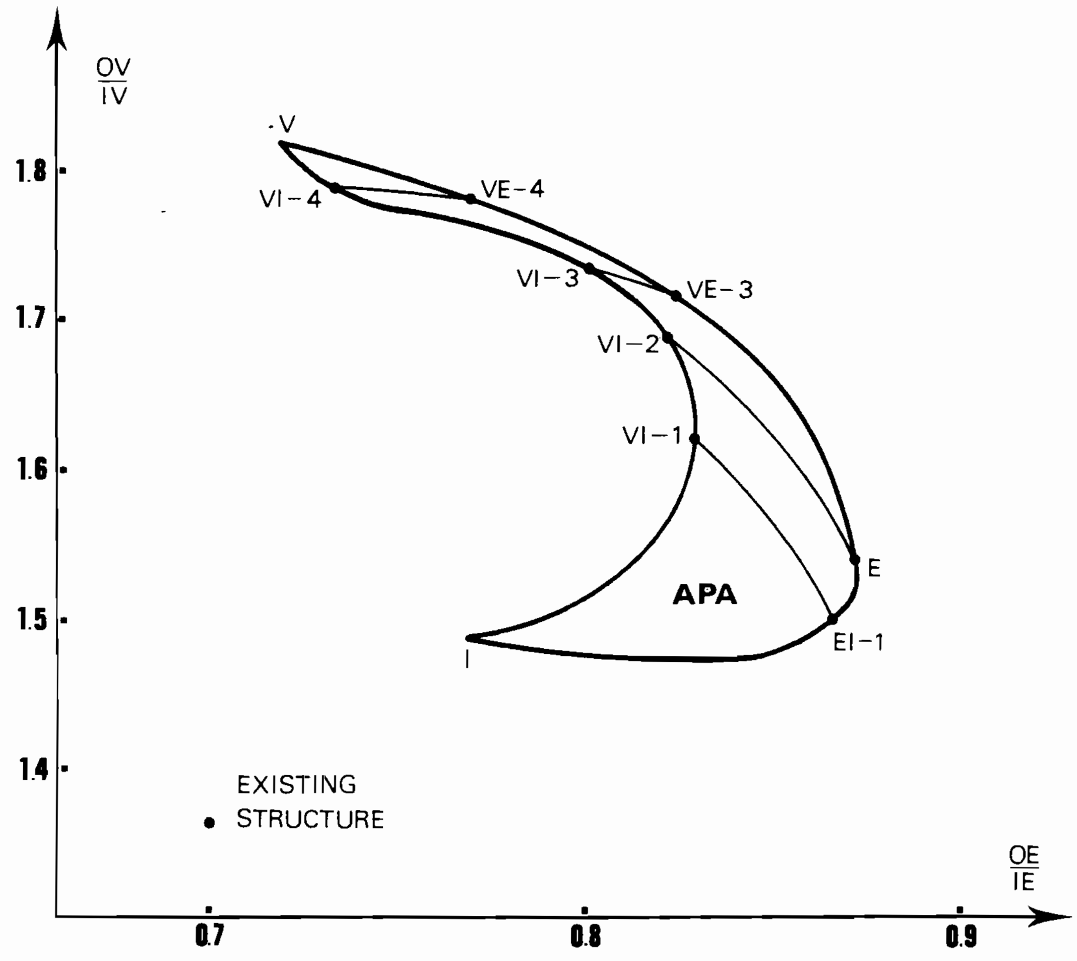


Figure 1. Pareto set for substitution model.

Acknowledgements.

Author feels deeply indebted to his co-workers from JSRD and would like to express his special gratitude to Dr G. Dobrowolski who helped in conceiving the substitution model. W. Ziembla helped significantly in the last stage of completion of the paper when the concept of APA was utilized for IDS evaluation. The above does not take off the responsibility from the author for any faults that may occur in the presented paper.

References.

Borek A.,Dobrowolski G., Zebrowski M., (1979). *Applications of System Analysis in Management of Growth and Development of the Chemical Industry.* Report CHEM/SEM.8/R.16 Chemical Industry Committee of the United Nations.

Dobrowolski G.,Kopytowski J., Wojtania J., Zebrowski M. (1984). *Alternative Routes from Fossil Resources to Chemical Feedstocks.* IIASA Research Report RR-84-19, Laxenburg, Austria.

Dobrowolski G., Zebrowski M., (1985). *Decision Support in Substitution Analysis for IDS - Industrial Development Strategy Exemplified by the Fuel and Feedstocks Sector of the Chemical Industry. The Application of DIDAS.* in "Theory, Software and Test Examples for Decision Support Systems", A. Lewandowski and A. Wierzbicki eds., Laxenburg, Austria.

Dobrowolski G., Rys T., Zebrowski M., (1985). *MIDA - Multiobjective Interactive Decision Aid in the Development of the Chemical Industry.* in "Theory, Software and Test Examples for Decision Support Systems", A. Lewandowski and A. Wierzbicki eds., Laxenburg, Austria.

Skocz M., Zebrowski M. (1986). An Extended Resources Allocation Method in Design of Industrial Development Strategy. *Proceedings of IFAC Symposium on Large Scale Systems, Zurich.*

TAB 1. APA - Attainable Performance Area - Vertices

Symbol of experiment	Criterion name	OV / IV	OE / IE	AV / II	II (Input Investment)
V	Economic efficiency (OV / IV)	1.80516	0.72361	0.25237	15576
E	Energy conversion efficiency (OE / IE)	1.51682	0.87435	0.62281	4340
I	Return of Investment (AV / IV)	1.48720	0.77076	2.02212	1130

TAB 2. APA - Attainable Performance Area - Cross Sections parameterized by value of AV/II

No	Value of	Symbol of	OV / IV	OE / IE	II	Symbol of	OV / IV	OE / IE	II
	AV / II	Experiment	|	mln $	Experiment	|	mln $		
1	1.00000	VI - 1	1.61349	0.83056	2646	EI - 1	1.48593	0.86807	2573
2	0.62281	VI - 2	1.68658	0.8260	4724	E	1.51682	0.87435	4340
3	0.45372	VI - 3	1.73178	0.79939	7348	VE - 3	1.70738	0.82508	6612
4	0.30815	VI - 4	1.78349	0.7291	12545	VE - 4	1.77041	0.77374	11728

TAB 3. Implementation Horizon (years) and Rate of Return of Investment (%/year) of Development Strategy (parameterization by return period).

Symbol of experiment	I	VI - 1 EI - 1	VI - 2 E	VI - 3 VE - 3	VI - 4 VE - 4	V
Return period						
4 years %/yr	177.21	75.00	37.28	20.37	5.82	0.24
yr	0.56	1.33	2.68	4.91	17.19	421.94
5 years %/yr	182.21	80.00	42.28	25.37	10.82	5.24
yr	0.55	1.25	2.37	3.94	9.24	19.09
6 years %/yr	185.55	83.33	45.61	28.71	14.15	8.57
yr	0.54	1.20	2.19	3.48	7.07	11.67

TAB 4. BASIC PARAMETERS and VALUES DESCRIBING:

1. Existing Industrial Structure
2. APA - Attainable Performance Area

Symbol of experiment	Unit	Existing Structure	I AV/II->max	E OE/IE->max	V OV/IV->max
Criterion Value	----	-------	2.022	0.873	1.805
Basic Results:					
Added Value	mln $	1026	2285	2703	3931
Output Value	mln $	3799	6975	7933	8811
Input Value	mln $	2772	4690	5230	4881
Input Investment	mln $	----	1130	4340	15576
Output Energy	mln Gcal117	232	341	288	
Input Energy	mln Gcal167	301	390	398	
Energy Balance	mln Gcal51	69	49	109	
Consumption of Critical Feedstocks:					
Gas	mln m3	1613.5	1453.7	543.8	------
Crude Oil	th t	13390.1	22797.0	18151.1	13655.4
Coal	th t	478.5	7444.7	4911.6	42358.2
Lignite	th t	-------	-------	44715.1	26473.6
Pirite Coal	th t	-------	1050.0	18375.0	1050.0

Spatial PDA Modelling for Industrial Development with Respect to Transportation Costs

Maciej Skocz, Wieslaw Ziembla

Joint Systems Research Department,
Institute for Control and Systems Engineering,
Academy of Mining and Metallurgy, Cracow
Industrial Chemistry Research Institute, Warsaw.

ABSTRACT

A modular approach for programming development of a spatially distributed industrial system is proposed in the paper. The problem comprises two decision tasks: technological development of the system with multiobjective goals, and an allocation of production units according to minimization of transportation cost. For such a two-fold problem, a two-level decomposition method is applied. A heuristic coordination procedure aims at giving an easy insight into the solving process and enables on-line verification during computations of assumptions and data selected in the course of the problem identification. The method is being implemented as an interactive decision support system to be applied to programming development of the so-called Spatial Production-Distribution Area (PDAS) in the chemical industry.

1. THE PROBLEM OVERVIEW

Let us call a Spatial Production-Distribution Area a large technological network-like system comprising a set of locally concentrated production networks named local PDAS. The set of local PDAS is arbitrarily determined as a possibly widest set of technological alternatives reasonably preselected according to spatially dependent conditions. A final choice of locations as well as technologies to be developed in the given locations is the ultimate goal of the development programming procedure.

The problem of programming development without regard to spatial allocation of technology has been widely described (Borek et al.,1978; Dobrowolski et al., 1982, 1984, 1985; Rys et al., 1986; Skocz et al., 1986). This problem will be also of concern here, as a major decision task incorporated in the framework of programming development of PDAS. Moreover, the concepts delineated in the works quoted above will be maintained and used in the approach presented here, so that this contribution can be viewed as an extension including transportation factors into the methodology worked out earlier for programming the development of chemical industry.

It is important to note that transportation factors are only one of many others associated with possible spatial distribution of PDAS. Among the others factors worth mentioning are spatially allocated untransportable resources (ground, infrastructure etc.)

Before presenting the problem and its solution procedure in a formal way, let us look at the basic assumptions that resulted from problem identification:

a) Model decision variables of diverse kind and range are to be distinguished in PDAS, such as technological variables, transportation variables, etc.

b) Transportation cost (TC) is considered as an active factor that influences the location of production units and as such is one of driving forces of PDAS structure development. Hence, the arising optimization problem (min TC) does not constitute a classical transportation problem but is a kind of location problem.

c) The transportation variables should be carefully selected in order to delete variables of negligible influence on spatial allocation of technological units. Similarly, parameters of transportation network should be aggregated to assure consistency between the scale of technological model and the scale of transportation model.

d) The PDAS model and the solution procedure should make it possible to analyse parameters that determine final solution comprising a selected technological repertoire and the location of units. In other words, the model should better explain how technological and transportation conditions influence the spatial structure of PDAS to be developed.

The above assumptions imposed strong requirements on the formal description of the problem. According to the points a), c), the proposed model has a hierarchical structure where the submodels are oriented towards technological and transportation data.

The coordination scheme of solving the whole problem enables interactive insight into the solving process. Therefore, on-line verification and modification of selected parameters during computations is possible. The idea of the two-level solving procedure is as follows:

Level 1

The problem of an overall PDAS development is formulated and solved. The identification of the problem comprises possible technological alternatives, global constraints either of technological or economic type, as well as strategic goals of development. Location of technological units is not taken into account at this stage of the algorithm, so only the selection of technologies that best suit conditions and goals forms a solution to the problem. More specifically, given a foreseen demands for some products, availability of resources and economic conditions, the development program is worked out. This program is expressed in terms of capacities of production processes. Moreover, various characteristics of the solution are given according to the assumed goals (e.g overall profit, consumption of resources).

Level 2

The overall development program of the PDAS is decomposed into a series of development programs concerning local PDAS. The decomposition is performed according to minimization of the transportation cost objective. At this stage, capacities of local technological processes are determined subject to local constraints (both of PDA-type and transportation-type). This implies an interactive use of two models; a PDA-like model and a transportation model. A mode of interaction, as well as of communication with the Level 1, is proposed following a coordination procedure that makes the algorithm convergent to a suboptimal solution.

The proposed approach results in a modular structure of a decision support system that has numerous advantages from the theoretical and practical point of view.

2. MATHEMATICAL MODELS

Let us now present the idea described above in a formalized way.

Level 1

At this level, the problem of development programming is formulated within a framework of so-called PDA concept (Dobrowolski et al., 1982, 1984, 1985). For the sake of comprehensive presentation of the whole algorithm, we are recalling a basic form of the PDA model. The model takes into account:

- the processing and flows of chemicals with specification of hazardous substances and hazardous processes within the PDA,
- the flows of chemicals (with specification of hazardous substances) into and out of other areas and industries representing the marketing or business activity of the PDA,
- the flow of investment, revenue and other resources such as energy, manpower, etc.

It should be emphasized that the version of the model described here has been constructed according to the other main assumptions of the basic model, i.e.:

1. it represents the equilibrium state of the PDA,
2. it includes only easily quantifiable physical elements of the system (without taking into account important but not quantifiable social or political factors).

Before describing the network of the PDA, we define its links with the environment (Fig.1). From this figure we can write the following equation describing the outflow of any chemical j:

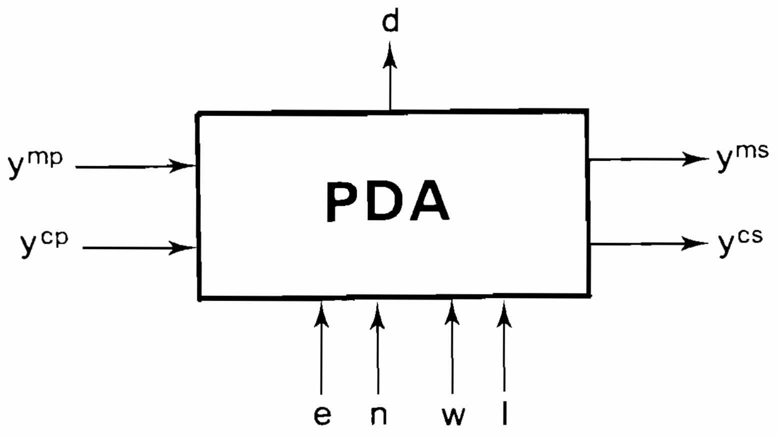


Figure 1. The links between a Production Distribution Area and its environment.

$$y_j = y_j^{ms} - y_j^{mp} + y_j^{cs} - y_j^{cp}\ , \quad j \in J \quad , J \supset J_t \tag{1}$$

y_j^{ms} - market sale of chemical j,
y_j^{mp} - market purchase of chemical j,
y_j^{cs} - coordinated sale of chemical j,
y_j^{cp} - coordinated purchase of chemical j,
J - set of indices representing chemicals of the PDA,
J_t - set of indices of substances that cause transportation cost of substantial value.

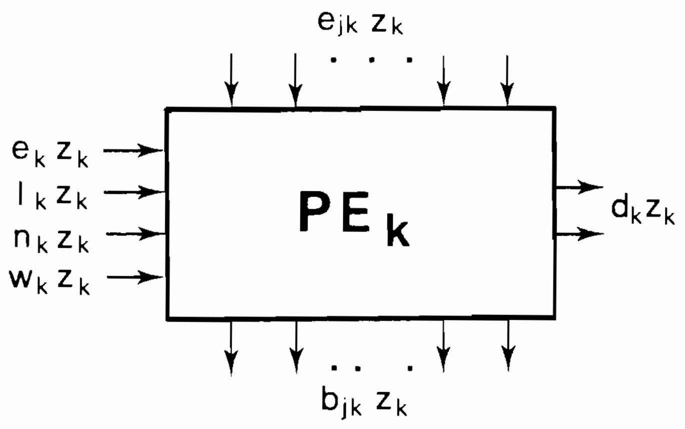


Figure 2. Processes element PE_k and the associated variables and parameters.

The variables in Fig.1-2 may be defined as follows:

z_k - production level of PE_k,

$\hat{z}_k$ - production capacity of PE_k,

$a_{jk}\ z_k$ - quantity of chemical j consumed by PE_k,

$b_{jk}\ z_k$ - quantity of chemical j produced by PE_k,

$d_{lk}\ z_k$ - quantity of waste l produced by PE_k,

$q_k(z_k)$ - necessary resources,

$e_k\ z_k$ - quantity of energy consumed by PE_k,

$w_k\ z_k$ - quantity of water consumed by PE_k,

$l_k\ z_k$ - quantity of labor consumed by PE_k,

$n_k\ z_k$ - investment for PE_k,

For the balance nodes, the following equations are satisfied:

$$y_j = \sum_{k \in K} b_{jk}\ z_k - \sum_{k \in K} a_{jk}\ z_k$$

By combining the above results with (1) we obtain:

$$y^{ms} - y^{mp} + y^{cs} - y^{cp} = (\ B - A\ )\ z \tag{2}$$

To complete this description of the network we have to add the constraints imposed on production capacity:

$$z \leq \hat{z} \tag{4}$$

For multiprocess installations, where processes run simultaneously, instead of the latter equation we use the following constraint:

$$\sum_{k \in K_i} z_k \leq \hat{z}_i \quad , i \in I \tag{4a}$$

where I, K_i denote correspondingly the sets of indices of installations and processes run on i-th installation. In addition, the model describes redevelopment of installations, i.e. substitution of an old process by a new one run on the same installation. For a given k-th process it is formulated as follows:

$$\frac{1}{\hat{z}_k^o} z_k^o + \frac{1}{\hat{z}_k^n} z_k^n \leq 1 \tag{5}$$

where: $\hat{z}_k^o$, $\hat{z}_k^n$ denote capacities of an old and new k-th process, z_k^o , z_k^n denote production levels of an old and new k-th process.

The introduction of new technologies is fundamental to this approach for development programming, as it opens the way to technological restructuralization of the PDA.

It is obviously necessary to add some additional constraints on resource availability or waste production limits and a set of criteria which reflect preference or goals of the decision maker. Since formulation of the constraints seems to be straightforward, here we present only some optimization criteria which may be of interest in the analysis.

First, we assume that for a fixed production goal we are interested in maximizing the added value (or revenue) that leads us to the problem:

$$Q_{rev} = \sum_{j \in J} c_j^s \left(y_j^{ms} + y_j^{cs} \right) - c_j^p \left(y_j^{mp} + y_j^{cp} \right) \quad \rightarrow \quad \max \tag{6}$$

with constraints given by market conditions and production capacities.

Following another decision strategy, resource consumption may be minimized, which results in the set of criteria:

$$Q_{ener} = \sum_{k \in K} e_k z_k \quad \rightarrow \quad \min \tag{7a}$$

$$Q_{inv} = \sum_{k \in K} n_k z_k \quad \rightarrow \quad \min \tag{7b}$$

$$Q_{labor} = \sum_{k \in K} l_k z_k \quad \rightarrow \quad \min \tag{7c}$$

From the above objective functions, it is possible to derive other useful optimization problems based on linear fractional functions (e.g. Q_{rev} / Q_{ener}) as well as various multiobjective problems. Of course, any objective (6) - (8) can be transposed onto a corresponding constraint. Discussions of the problem of choice of relevant objectives go, however, beyond this formal description of the model.

For brevity, it will be sometimes convenient to represent the above described model as an optimization problem in a compact form called P-PDA:

$$Q\ (z,y) \quad \rightarrow \quad \min \tag{8}$$

s.t.

$$G\ (z,y)\ \geq\ 0$$

where y $= \{\ y_j\ ,\ j \in J \supset J_t\ \}$, z $= \{\ z_k\ ,\ k \in K\ \}$, J is a set of indices representing chemicals, J_t denotes set of indices of substances that cause transportation cost of substantial value.

Note also that every y_j can be split into the following components:

y_j^{ms} - market sale of chemical j,
y_j^{mp} - market purchase of chemical j,
y_j^{cs} - coordinated sale of chemical j,
y_j^{cp} - coordinated purchase of chemical j,

As usual, z_k denotes production level of process k, K is a set of all technological processes in the PDAS.

To summarize, as an output of the optimization procedure at *Level 1* one obtains outflows of chemicals y_j split into y_j^{ms}, y_j^{mp}, y_j^{cs}, y_j^{cp}, $j \in J$. These variables and associated with them z_k , $k \in K$ represent global directions of the whole PDAS development.

It should be emphasized that the above model includes all the assumptions of the basic PDA model, which will not be quoted here. It is obvious, however, that both the objective function(s) and type of constraints that occur in the PDAS version are implementation-specific.

Models of local PDAS have an analogous form. They are distinguished from the general PDAS model with superscript l designating all components of the model.

Level 2

Let us now denote by L a set of possible locations of local PDAS. Chemical substances that are consumed in technological processes in a given location can be either produced in the same local PDAS, other local PDAS, or can be purchased on the market beyond the whole PDAS. Moreover, chemicals can be transported between given sites along different routes that cause different transportation cost.

The entire transportation cost in the whole PDAS can be evaluated according to the following formula:

$$TC = \sum_{j \in J_t} \sum_{l \in L} [\ \sum_{n \in L} \sum_{r \in R_{nl}} x_{jr}\ d_r\ P_{jr} + \sum_{m \in M} \sum_{r \in R_{lm}} v_{jr}\ d_r\ P_{jr}\ ] \tag{9}$$

where:

J_t	- set of chemicals that cause meaningful transportation cost,
R_{nl}, R_{lm}	- set of routes that connect locations n,l and l,m
x_{jr}	- amount of substance j transported between sites by the route r,
M	- set of markets,
v_{jr}	- amount of substance j imported and/or exported by the route r,
d_r	- distance along r,
P_{jr}	- price for transportation of unit product j along route r.

In order to formulate an optimization problem for minimization of TC, the following constraints are taken into account:

$$\sum_{l \in L} \sum_{r \in R_{ml}} v_{jr} \leq V_j^m \tag{10}$$

$$\sum_{l \in L} \sum_{r \in R_{lm}} v_{jr} \leq S_j^m \tag{11}$$

$$\sum_{j \in J_t} \sum_{l \in L} \sum_{n \in L} \sum_{r \in R_{lm}} x_{jr} \leq T \tag{12}$$

$$\sum_{l \in L} \sum_{n \in L} \sum_{r \in R_{lm}} x_{jr} \leq t_j \; , j \in J_t \tag{13}$$

$$\sum_{j \in J_t} x_{jr} \leq R_r \tag{14}$$

where:

V_j^m - availability of substance j from market m,

S_j^m - salability of substance j from market m,

T - total transportation potential,

t_j - transportability of substance j,

R_r - maximal transportation flow along route r

It is important to observe that the above model has a natural connection with the PDA model, because the following equations hold:

$$\sum_{n \in L} \sum_{r \in R_{ln}} x_{jr} = y_j^{l\,cs}, \tag{15}$$

$$\sum_{m \in M} \sum_{r \in R_{lm}} v_{jr} = y_j^{l\,ms}, \tag{16}$$

$$\sum_{n \in L} \sum_{r \in R_{nl}} x_{jr} = y_j^{l\,cp}, \tag{17}$$

$$\sum_{m \in M} \sum_{r \in R_{ml}} v_{jr} = y_j^{l\,mp}, \tag{18}$$

where $y_j^{l\,cs}$, $y_j^{l\,cp}$, $y_j^{l\,ms}$, $y_j^{l\,mp}$, $l \in L$, denote spatial coordinates of vectors y_j^{cs}, y_j^{cp}, y_j^{ms}, y_j^{mp}.

As follows from the above observation, the solution to the transportation problem can provide a feasible decomposition of the PDA model outputs (y^{cs}, y_j^{cp}, y_j^{ms}, y_j^{mp}), as far as the conditions:

$$\sum_{l \in L} y_j^{l\,cs} \leq y_j^{cs}, \tag{19}$$

$$\sum_{l \in L} y_j^{l\,ms} \leq y_j^{ms}, \tag{20}$$

$$\sum_{l \in L} y_j^{l\,cp} \leq y_j^{cp}, \tag{21}$$

$$\sum_{l \in L} y_j^{l\,mp} \leq y_j^{mp}, \tag{22}$$

are assumed to hold in the transportation problem.

Concluding, the transportation problem consists of the objective (9), transportation constraints (10)-(14) and linking constraints with the PDAS model (15)-(22). The latter are put down in a redundant form to better visualize the link between the two models.

It is important to note, however, that the above spatial decomposition of the global solution takes into account only transportation factors and as such neglects local conditions that may influence technological development of the local PDAS. For the above reason, a coordination procedure that combines solutions derived from the PDAS and the transportation model is proposed.

3. SOLVING ALGORITHM

The proposed algorithm consists of the following steps:

Step 1.

Generation of the overall PDAS development program, i.e determination of the variables $\{y_j,\ j \in J\}$, $\{z_k,\ k \in K\}$. For further considerations the set K (potential technologies) will be always limited to those selected by non-zero variables z_k (K = {k; z_k 0}.

Step 2

Solving the problems regarding development of local $PDAS_l,\ \ l \in L$);

$$Q^l\ (z^l,y^l) \quad \rightarrow \quad \min \tag{23}$$

s.t.

$$G^l\ (z^l,y^l)\ \geq\ 0$$

where G^l represents also a set of local constraints on resources (manpower, water, energy etc.). The solution obtained at this step is $\hat{y}^l$, $\hat{z}^l$.

Step 3

Solving the transportation problem:

$$TC(x,y,z) \quad \rightarrow \quad \min \tag{24}$$

s.t.

$$y = \left\{ y_j^l,\ \sum_{l \in L} y_j^l \leq y_j,\ j \in J_l \ \text{ and } \ y_j^l \leq \hat{y}_j^l \right\}$$

The solution to the above problem $y_j^l = \tilde{y}_j^l$ will be used in the next step.

Step 4

Choice of the $PDAS_l$ that satisfies the following condition:

$$\|\ \hat{y}_j^l - \tilde{y}_j^l \| \quad \rightarrow \quad \min \tag{25}$$

The solution is $l = s$.

Step 5

Generating eventual optimal structure of the $PDAS^l$ through solving the PDA^s, s.t. additional constraint $y_j^s \leq \tilde{y}_j^s$. The solution is denoted $\bar{y}_j^s$, then y_j obtained from solving P-PDA is updated (y_j <-- $y_j - \bar{y}_j^s$). The selected set of technologies K^s will be substracted from the set K ($K\ :=\ K\backslash K^s$). Similarly, L will be diminished by $s(L{:=}L\backslash\{s\})$.

If $L \neq 0$, go to step 2, otherwise the procedure is terminated.

4. COMMENTS AND CONCLUSIONS

The concept of a hierarchical coordination of programming of the development of a spatially allocated PDA, described in this paper, is suggested as a basis for a decision support system for these purposes. The preliminary character of this concept should be stressed, however; many questions require further analysis and research before constructing such decision support system. The role of the decision maker interacting with the coordination algorithm at its both levels should be clarified, the convergence of such a hierarchical interactive procedure should be analysed, alternative procedures of interactive hierarchical coordination should be taken into account. Nevertheless, the example presented in this paper stresses the importance of research on hierarchical formulations of models in decision support systems.

References.

Borek A., Dobrowolski G., Zebrowski M. (1978). *GSOS - Growth Strategy Optimization System for the Chemical Industry.* Proc. of MECO-78, Athens.

Dobrowolski G., Kopytowski J., Lewandowski A., Zebrowski M. (1982). *Generating Efficient Alternatives for Development in the Chemical Industry.* Collaborative Paper CP-82-54, IIASA, Austria.

Dobrowolski G., Kopytowski J., Wojtania J., Zebrowski M. (1984). *Alternative Routes from Fossil Resources to Chemical Feedstocks.* Research Report RR-84-19, IIASA, Austria.

Dobrowolski G., Kopytowski J., Rys T., Zebrowski M. (1985). *MIDA: Multiobjective Interactive Decision Aid in the Development of the Chemical Industry.* in Theory, Software and Practical Examples for Decision Support Systems, IIASA, pp.235-251.

Rys T., Skocz M., Ziembla W., Zebrowski M. (1986). *Modelling the Chemical Industry for Hazardous Waste Management.* Proc. of AMSE Conf. on Modelling and Simulation, Sorento, Italy.

Skocz M., Zebrowski M., Ziembla W. (1987) *A Method for Design of Industrial Investment Strategy* Paper accepted for X IFAC Congress in Munich, 1987.

Ranking and Selection of Chemical Technologies Application of SCDAS Concept

Grzegorz Dobrowolski and Maciej Zebrowski

Joint Systems Research Department,
Institute for Control and Systems Eng.,
Industrial Chemistry Research Institute

1. INTRODUCTION

In the design of Industrial Development Strategy (IDS) two phases may be distinguished. The first is generation of efficient development alternatives, the second is selection and ranking of these alternatives. Regardless of particular line of responsibility in a given organization the decision process in this case involves a group of experts. They should be able to arrive at the conclusive results of their search so that a decision could be made.

The second phase starts when a given number of alternatives have been generated as a result of methodologically ordered process. The methodology considered is MIDA or Multiobjective Interactive Decision Aid (Dobrowolski et al., 1985; Zebrowski, 1986). This process is performed on two consecutive levels of MIDA hierarchy. The higher level covers a global IDS referring to the whole industrial branch such as described by so called PDA or Production Distribution Area model (Dobrowolski et al. 1984). The alternatives are understood here as development strategies. On the second, lower level called Comparative Study of Technologies individual technologies are considered (Dobrowolski et al., 1984, Gorecki et al., 1984).

In the decision support systems designed for MIDA the process of selection and ranking of alternatives is built in a form of heuristic procedures based on judgement of a decision maker and his experts. It is to be explored how we could support these procedures with formally constructed decision support tools.

We have selected an approach called SCDAS or Selection Committee Decision Analysis and Support System (Lewandowski et al., 1986).

In this paper the first attempt to apply SCDAS type of approach to the group decision in the area of the design of IDS is reported. We begin with description how SCDAS can be incorporated in MIDA decision support system. The problem is exemplified by selection and ranking of alternative technologies for methanol production. The paper concludes with proposal for continuation of the effort reported here aimed at developing a specialized tool for group decision support that is foreseen to be incorporated as MIDA module.

2. THE PROBLEM AREA

Design of IDS for the chemical industry is a mainstream of our research since almost a decade (e.g. Borek et al., 1978; Dobrowolski et al., 1984). From this experience came a strong demand for development of tools that could efficiently help in ranking and selection of development alternatives. Development of advanced decision support system tools based on the efficient multiobjective optimization software (e.g. DIDAS see Dobrowolski et al. 1985), made the generation of decision alternatives well equipped. The easier is generation process, the richer is the array of possibilities to be evaluated and selected by a decision maker and his experts.

The problem discussed here i.e. ranking of a finite set of alternatives and selection of one of them by a group of experts is a classical decision problem. Without going into details we may conclude that strong need exists for developing simple but advanced decision support system of tools for evaluation, ranking and selection of decision alternatives.

We decided to consider here an approach that led to development of the package called SCDAS. Several other approaches to the above problem may be considered. One of the very promising, which deserves more attention is the one described by Siskos (Siskos et al., 1986). This is a multicriteria ranking of alternatives utilizing method called Electra (Roy,1978; Skalka et al.,1983).

The SCDAS can be located in the quasi-satisficing framework of decision support (see e.g. Wierzbicki 1982) which is an extension of the satisficing framework (as introduced by Simon 1958). As it was assumed in the SCDAS approach, there are several alternatives and a group of experts who are to make a selection. They form a committee with equal or varying voting power. The voting power as well as procedural terms of reference for the case to be considered should be formulated in the committee's charter. Each alternative is described by a set of attributes that may be expressed by numbers. The procedural variant preferred in SCDAS is an *aspiration level-led* process: each expert (committee member) is asked to specify anchor levels (aspiration and reservation levels) for all attributes. The aspiration and reservation levels are used to obtain an achievement function – which can be interpreted as an approximate multivariable cardinal utility function that is further averaged and maximized in the system (Wierzbicki, 1986).

Next the experts enter the evaluation process. Having in mind their aspiration and reservation levels, the experts are to assign levels of attributes for all alternatives. While using the achievement function, either individual or the group ranking is obtained. A decision analysis process results in final ranking and selection of the alternatives in a way that is rational, understandable and acceptable to the committee members. The procedural framework for the decision process assumes that set of attributes as well as alternatives may be changed by the experts.

Two levels in MIDA hierarchy were selected for application of that new tool for selection and ranking of development alternatives. First comes the level aimed at evaluation and selection of an IDS for the whole industrial branch as described by the PDA type of the model (Dobrowolski et al., 1985). Thus development strategies become the alternatives. Second would be the lower level which is closely related to the above. That would be a Comparative Study of Technologies (Dobrowolski et al., 1984; Gorecki et al., 1984). Here technologies will be ranked. We can describe jointly tasks for both levels.

The result of SCDAS session is to be presented to a decision maker in form of ranking of alternatives obtained from the earlier stage of MIDA predecision analysis. Most of the attributes would be common for both cases under consideration. This results from the MIDA methodology and reflects real life decision environment: attributes must be homogenous throughout the process of predecision analysis and at least some of them must assure continuity and coherence when results are obtained from level to level.

There are two types of attributes. First is represented by intensive parameters such as rate of return, investment, production capacity, wastes etc. They are easy for comparison since they are usually backed by engineering and economic type of data. Some of them are expressed in absolute numbers and some in relative values. In terms of industrial experts these attributes describe scale (volume) or intensity of industrial operation.

Second type of attributes are those describing some trends or character of the phenomena. These are e.g. availability of resources, marketing forecasts, etc. They are much less reliable, more difficult for quantification and therefore the role of expert's experience and judgement is decisive at the stage of attributes' evaluation.

Following is a proposed list of attributes with some comments; in any individual case, a selection or an aggregation of this list should be made in order to limit the number of attributes to a reasonable one.

1. Rate of return of investment.
   This parameter should be related to the current rate of interest.
2. Productivity of the capital.
3. Thermal efficiency.
   It is equivalent to ECE or Energy Conversion Efficiency calculated from the lower heating values of media in question.
4. Investment to manpower intensity.
   It is to express cost of creating a new work place. As investment cost battery limits investment is considered.
5. Cost of investment.
   The following elements of the cost should be analyzed: battery limits, construction cost, offsides as well as working capital.
6. Terms of trade impact.
7. Income.
8. Cost of production.
   It may be calculated using yields and consumption coefficients and assumed prices. Overheads, taxes, insurance and other factors may be also incorporated.
9. Availability of raw materials.
   It should reflect the proportion of the demand for the raw material due to the new investment related to the markets share.
10. Availability of technology.
    This may be related either to the number of potential suppliers and/or to potential necessity of imports as opposed to domestic source of technology.
11. Environmental impacts.
    It serves the purpose of evaluation of particular dangers to the environment expressed in amounts of wastes.
    In individual case it can be also expressed in terms of emission of a particular waste agent per unit of main product.

The above would be a brief description of the background of the problem area. The idea emerging from it can be explained further by an example.

3. RANKING AND SELECTION OF TECHNOLOGY. THE CASE OF METHANOL.

This case was selected for its simplicity and generality. It was also used by us previously; therefore, it provides a good material for testing and comparison (Dobrowolski et al., 1984). The software used was an original SCDAS package version (Lewandowski et al., 1986).

The scope.

Suppose that, from the higher level of MIDA analysis, it became evident that production of methanol is to be located in a particular region. The estimated demand for methanol production was 500,000 MTPY.

The region in question is rich in low grade coal which is not suitable for power generation. Since its mining cost is low due to its easy accessibility it is to be evaluated as a feedstock for the chemical processing. The cost of this coal is estimated at 22 USD per MT as opposed to the average market price of coal being 34 USD per MT of coal used for power generation. One of the possible locations for methanol production is a large chemical plant which, among many other products, produces carbide.

The acethylene obtained from carbide is a very valuable chemical feedstock. As a by-product from carbide plant so called carbide gas (rich in carbon monoxide) is obtained. This can be used in combination with methane for chemical synthesis, specifically for methanol production.

Another potential location to be considered is a site next to the open lignite pit. The lignite at the price of 15 USD per MT is therefore another possible domestic feedstock. Other feedstocks would have to be imported.

There are two conflicting strategies for the future development of the region.

The first assumes autarchic development that strongly favors domestically available raw materials. It also represents restrained approach to the investment. The autarchic approach leads to depreciation of economic and market factors. Consequently the environmental effects may become relatively less important.

The second strategy as opposed to the previous one is an open approach, economy and market oriented. Nevertheless factors such as environmental impacts are also to be considered.

Alternatives.

Methanol is usually produced on the industrial scale from syngas, a 1:2.2 mixture of carbon monooxide and hydrogen, in the presence of a catalyst and under specific temperature and pressure conditions.

Syngas may be obtained from any raw material containing carbon, and therefore all fossil resources are potential starting materials. The following are the most commonly used:

- natural gas;
- natural gas combined with a source of carbon monooxide, such as the residual gas from carbide ovens;
- heavy fractions and the heavy residue from distillation of crude oil;
- hard coals and lignite.

The choice of raw material depends largely on the resources and conversion technologies available.

Nine technologies of methanol based on the above raw materials are the alternatives presented to the experts. The design office prepared technical information for all units for the same capacity of 500,000 MTPY (metric tones per year). Instead of numeration, 3-character symbols are provided here for the sake of identification during the SCDAS session.

NG. From natural gas to methanol through steam reforming of methane.

N+C As above, but with the addition of (carbon monooxide containing) residual gas from carbide ovens.

HR. From heavy residues to methanol through partial oxidation.

LG. From coal to methanol through medium-pressure gasification of the coal, followed by SNG (synthetic natural gas) production and subsequent steam reforming of methane.

FT. Coal gasification followed by Fisher-Tropsch synthesis.

TEX From coal through pressure gasification of coal-water paste in oxygen atmosphere - TEXACO process.

HTC High temperature (1400 K) Winkler process for hard coal.

HTL As above, but for lignite.

KT. From coal to methanol through low-pressure gasification of the coal (Koppers-Totzek-type process).

So called Fisher-Tropsch technology falls out of the line of others due to the scale of its operation since it produces very large quantities of of SNG (Synthetic Natural Gas) and 500,000 MTPY of methanol is, in fact, not the main product of this plant. It produces also other feedstock apart from SNG. But due to the specific role of coal in the region it was also included as one of the technological alternatives.

To give a good foundation for ranking and selection all the committee members have been equipped with the description of the alternative processes. Table 1 contains technological parameters for the first considered technology. The reader can find the data of this type for all alternatives in Appendix B. Such data ought to be accepted by the whole committee as long as local aspects of implementation of the technologies can be neglected at this stage.

TABLE 1: Information sheet for NG. technology.

General			
Battery Limits		184220	th.USD
Manpower		105	men
Capacity		500000	tons/year
Inputs		unit/year	
electric energy	8.000	4000000	kWh
steam	0.080	40000	Gcal
water	11.000	5500000	m3
natural gas	0.922	461000	th.m3
Co-Mo catalyst	0.030	15000	kg
ZnO catalyst	0.100	50000	kg
reforming catalyst	0.033	16500	kg
catalyst of synthesis	0.200	100000	kg
Outputs		unit/year	
waste water	5.048	2524000	m3
methanol	1.000	500000	tons

Attributes

The set of the attributes for the case has been proposed. They are enumerated in Table 2.

TABLE 2: List of attributes.

	Techno-economical:
RoR	Rate of Return
TEf	Thermal Efficiency
Pro	Productivity
B/L	Battery Limits / Labor Intensity
Inv	Cost of Investment
	Miscellaneous:
Raw	Availability of Raw Materials
FoT	Impact on Foreign Trade (Import)
Env	Environmental Impacts

In the Appendix C formula for calculating the above attributes are given.

As the auxiliary material values of the intensive parameters for the alternatives have been calculated and presented to the experts.

TABLE 3: Intensive parameters for the technologies.

	Intensive parameters:
UCo	Unit Cost
UIn	Unit Income
RoR	Rate of Return
Pro	Productivity
TEf	Thermal Efficiency
B/L	Battery Limits / Labour Intensity

TABLE 4: Values of the intensive parameters.

	UCo [USD/t]	UIn [USD/t]	RoR [%]	Pro [%]	TEf [%]	B/L [ml.USD/man]
NG.	214.82	26.18	7	65	62	1.754
N+C	201.79	43.67	12	65	72	1.676
HR.	330.92	-82.32	-13	39	50	1.611
LG.	347.09	-70.73	-8	33	37	1.498
KT.	404.31	-155.52	-16	26	30	1.558
FT.	1595.88	-870.75	-15	13	49	2.494
TEX	232.92	15.59	3	44	42	1.405
HTC	272.07	-23.38	-3	35	37	1.764
HTL	282.60	-34.58	-5	33	44	1.854

The calculation was based on assumed prices and average lower heating values which are reproduced in Appendix A. Negative numbers for UIn mean simply that a technology in question brings loss, consequently RoR has a negative value.

Ranking obtained.

Having obtained all the above information the group of five experts enter upon the stage of evaluation of the methanol technologies and they express their views through evaluating each alternative.* Final ranking that was generated by SCDAS basing on the votes is given in Table 5.

TABLE 5: Final ranking

1	2	3	4	5	6	7	8	9
N+C	NG.	TEX	HR.	LG.	HTC	HTL	KT.	FT.

The result can be commented briefly as follows. All technologies bringing loss in monetary or economical terms are ruled out and therefore the score is in full agreement with the technological and economical data provided to the experts. It can be seen that the second type of attributes practically does not play significant role in evaluation and selection of alternatives except for environmental impacts. The TEX technology being the third choice proves this point.

Although both development strategies for the region – namely the autarchic one and the open market oriented one – were represented by the experts in the voting process, it can be seen that the latter one became prevalent. To illustrate better the SCDAS application another case may be considered when aspiration levels and scores for attributes: availability of resources and impact on foreign trade were modified as if the experts tended to an autarchic development of the region. The resulting order is presented in Table 6.

TABLE 6: Technology ranking (autarchic tendency)

1	2	3	4	5	6	7	8	9
N+C	TEX	LG.	KT.	HTC	HTL	FT.	NG.	HR.

Comparing the results presented in both above tables it can be seen that in the first case (i.e. of market oriented strategy) the impact of techno-economic attributes is decisive, while in the second (hypothetical) case the other attributes referring to knowledge and experience of the experts are essential. It should be stressed, however, that the subjective factor remains present in the analysis with all its negative and positive aspects.

4. CONCLUSIONS AND PROPOSAL FOR THE FUTURE DEVELOPMENT.

The experience gained from application of SCDAS package for extending MIDA methodology proves its usefulness, although the package was originally developed not for this kind of application. The conclusions formulated here are obtained from the point of view of the proposed new application of SCDAS concept – that is, ranking and selection of alternatives in the process of design of industrial development strategies.

This kind of application requires an open construction of the decision support system. In MIDA system free access to various databases such as technological database is

*We do not reproduce here the 360 numbers which were the scores given by experts.

assumed . Moreover, results can be stored for future processing and evaluation. Therefore, special commands in SCDAS enabling data transfer would be useful. Additionally, a possibility for calling in the command interpreter of the operating system would be very practical.

It may be also concluded that an important area of future research will be the identification and methodological structuring of selection and ranking processes for the various attributes: their impacts, subjectivity versus objectivity etc. A basic condition for a successful application of the proposed approach will be an explicit and clear understanding by experts and decision makers of the above problems and their impact on the selection and ranking process. This is necessary in order to assure fairness of representation of experts views and knowledge as well as to avoid hidden biases of results owing to lack of understanding of the method.

It would be also useful to trace the progress in ranking by separate scoring of various subsets of attributes; for example, first ranking according only to ratios, then according to subset of attributes expressed in absolute numbers, etc. This would show the impact of attributes considered on the judgement of committee members and could provide a feedback which may help to arrive at conclusive results. Such an analysis may be especially useful in a more complex case, for example, when alternatives for IDS on the higher level are to be considered.

With the help of the functions proposed above we intend to develop various procedures related to designating aspiration and reservation levels, evaluating alternatives, etc. It may be concluded that further identification and methodological structuring of the selection and ranking process is an important area of future research. All this would provide for an useful application of SCDAS concept in MIDA environment.

References.

Borek A., Dobrowolski G., Zebrowski M. (1978) *GSOS - Growth Strategy Optimization System for the Chemical Industry.* Proceedings of MECO-78, Athens, Vol. 3.

Gorecki H. et al. (1984). *Multiobjective Approach to Project Formulation. Design of the Chemical Installation.* Springer Verlag, Proceedings of MECO-78, Athens, Vol. 3.

Dobrowolski G., Kopytowski J., Wojtania J., Zebrowski, M. (1984) *Alternative Routes from Fossil Resources to Chemical Feedstock.* Research Report RR-84-19, International Institute for Applied Systems Analysis, Laxenburg, Austria

Dobrowolski G., M. Zebrowski (1985) *Decision Support for Substitution Analysis in IDS (Industrial Development Strategy) Exemplified by Fuel and Feedstock Sector of the Chemical Industry; An Application of DIDAS.* Theory, Software and Test Examples for Decision Support Systems, Editors A.Lewandowski and A.Wierzbicki, pp. 219-234, International Institute for Applied Systems Analysis, 2361 Laxenburg, Austria.

Dobrowolski G., J. Kopytowski, T. Rys, M. Zebrowski (1985) *MIDA (Multiobjective Interactive Decision Aid) in the Development of Chemical Industry.* Theory, Software and Test Examples for Decision Support Systems, Editors A.Lewandowski and A.Wierzbicki, pp. 235-251, International Institute for Applied Systems Analysis, 2361 Laxenburg, Austria.

Keeney R.L.,Lathrop J.F, Sicherman A. (1985) *An analysis of Baltimore Gas and Electric Company technology choice.* J. Oper. Res., vol.34, No. 1.

Lewandowski A., S. Johnson, A. Wierzbicki (1986) *A Prototype Selection Committee Decision Analysis and Support System SCDAS: Theoretical Background and Computer Implementation.* Working paper WP-86-27, International Institute for Applied Systems Analysis, 2361 Laxenburg, Austria.

Siskos J., Hubert Ph. (1983) ***Multicriteria analysis of the impacts of energy alternatives: a survey and a new comparative approach.*** Eur. J. Oper. Res., 13, pp.278-299.

Siskos J., Lombard J. and Oudiz A. (1986) *The use of Multicriteria Outranking Methods in the Comparison of Control Options Against Chemical Pollutant.* J.Op.Res. Soc., vol.37, No 4, pp. 357-371.

Roy B. (1978) ELECTRE III: Un algorithme de classement fonde sur une representation floue des preferences en presence des criteries multiples. *Cah. Cent. Etud. Rech. Opl. 20.* pp. 3-24.

Skalka J.M., Boyussou D., Bernabeu Y.A. (1983). ELECTRE III et IV: aspects methodologiques et guide d'utilisation. Document LAMSADE No 25, LAMSADE, Universite Paris-Dauphine, Paris.

Skocz M., Ziembla W., Zebrowski M., (1986). *Multiobjective Design of Dynamic Industrial Development Trajectories. An Extension of MIDA Methodology.* in Theory,Software and Test Examples for Decision Support Systems Phase II. Eds A.Lewandowski & A. Wierzbicki, IIASA, Laxenburg, Austria. (in print).

Wierzbicki A. P. (1982). *A Mathematical Basis for Satisfying Decision Making.* Math. Modeling, vol.3, pp. 391-405.

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

Zebrowski M. (1986). *MIDA Application - Multiobjective Evaluation of Industrial Structure. The Case of Chemical Industry.* in Theory, Software and Test Examples for Decision Support Systems. Phase II. Eds A.Lewandowski & A.Wierzbicki, IIASA Laxenburg, Austria. (in print)

APPENDIX A

In this appendix, prices and lower heating values for each media involved in the analysis are reproduced. These parameters were used for calculation of all techno-economical indicators of the technologies.

TABLE A.1:Prices and LHV (Lower Heating Values) for the media under consideration.

	Price [unit]	Heating [Gcal/unit]	unit
ammonia	209.80	4.45	tons
carbide gas	27.50	2.50	th.m3
coal	22.00	4.07	tons
diesel oil	306.20	10.00	tons
electric energy	0.06	0.00068	kWh
heating gases	68.70	5.07	th.m3
heating oil	142.30	9.60	tons
heavy residue	142.00	9.60	tons
lignite	15.00	2.15	tons
liquid gases LPG	353.70	10.80	th.m3
methanol	241.00	4.60	tons
natural gas	110.00	8.00	th.m3
raw phenols	546.60	7.50	tons
steam	13.50	1.00	Gcal
sulphur	95.20	2.20	tons
superior alcohols	120.50	6.44	tons

APPENDIX B.

This appendix contains basic information on the technologies of methanol. The information is given on the special sheets that are presented to the committee members.

TABLE B.1:Information sheet for NG. technology.

General			
Battery Limits		184220	th.
Manpower		105	men
Capacity		500000	tons/year
Inputs		unit/year	
electric energy	8.000	4000000	kWh
steam	0.080	40000	Gcal
water	11.000	5500000	m3
natural gas	0.922	461000	th.m3
- catalyst	0.030	15000	kg
catalyst	0.100	50000	kg
reforming catalyst	0.033	16500	kg
catalyst of synthesis	0.200	100000	kg
Outputs		unit/year	
waste water	5.048	2524000	m3
methanol	1.000	500000	tons

TABLE B.2:Information sheet for N+C technology.

General			
Battery Limits		187810	th.
Manpower		112	men
Capacity		500000	tons/year
Inputs		unit/year	
electric energy	40.000	20000000	kWh
steam	0.260	130000	Gcal
water	11.000	5500000	m3
natural gas	0.730	365000	th.m3
carbide gas	0.224	112000	th.m3
Co-Mo catalyst	0.030	15000	kg
ZnO catalyst	0.136	68000	kg
reforming catalyst	0.033	16500	kg
catalyst of synthesis	0.200	100000	kg
Outputs		unit/year	
waste water	4.636	2318000	m3
methanol	1.000	500000	tons
superior alcohols	0.037	18500	tons

TABLE B.3:Information sheet for HR. technology.

General			
Battery Limits		315810	th.
Manpower		196	men
Capacity		500000	tons/year
Inputs		unit/year	
electric energy	63.000	31500000	kWh
steam	0.296	148000	Gcal
water	15.000	7500000	m3
heavy residue	0.992	496000	tons
conversion catalyst	0.080	40000	kg
catalyst of synthesis	0.200	100000	kg
air	3.520	1760000	th.m3
Outputs		unit/year	
waste water	6.910	3455000	m3
gas waste	0.116	58000	th.m3
methanol	1.000	500000	tons
sulphur	0.033	16500	tons
superior alcohols	0.037	18500	tons

TABLE B.4:Information sheet for LG. technology.

General			
Battery Limits		421100	th.
Manpower		281	men
Capacity		500000	tons/year
Inputs		unit/year	
electric energy	123.000	61500000	kWh
steam	4.160	2080000	Gcal
water	18.000	9000000	m3
coal	3.032	1516000	tons
air	3.637	1818500	th.m3
catalyst of synthesis	0.200	100000	kg
Outputs		unit/year	
waste water	8.132	4066000	m3
gas waste	1.028	514000	th.m3
ashes	0.355	177500	tons
methanol	1.000	500000	tons
ammonia	0.021	10500	tons
raw phenols	0.014	7000	tons
heating oil	0.109	54500	tons
sulphur	0.035	17500	tons
superior alcohols	0.037	18500	tons

TABLE B.5:Information sheet for FT. technology.

General			
Battery Limits		473710	th.
Manpower		304	men
Capacity		500000	tons/year
Inputs		unit/year	
electric energy	300.000	150000000	kWh
steam	6.170	3085000	Gcal
water	21.000	10500000	m3
coal	2.444	1222000	tons
air	4.792	2396000	th.m3
catalyst of synthesis	0.200	100000	kg
heating gases	0.024	12000	th.m3
conversion catalyst	0.070	35000	kg
Outputs		unit/year	
methanol	1.000	500000	tons
superior alcohols	0.037	18500	tons
sulphur	0.035	17500	tons
waste water	13.506	6753000	m3
gas waste	1.126	563000	th.m3
ashes	0.306	153000	tons

TABLE B.6:Information sheet for TEX technology.

General			
Battery Limits		2990610	th.
Manpower		1199	men
Capacity		520000	tons/year
Inputs		unit/year	
coal	17.162	8924240	tons
electric energy	2.000	1040000	kWh
water	20.000	10400000	m3
Outputs		unit/year	
natural gas	3.113	1618760	th.m3
liquid gases LPG	0.065	33800	tons
diesel oil	0.203	105560	tons
heating oil	0.060	31200	tons
superior alcohols	0.146	75920	tons
ammonia	0.060	31200	tons
sulphur	0.220	114400	tons
ashes	2.308	1200160	tons
methanol	1.000	520000	tons

TABLE B.7:Information sheet for HTC technology.

General			
Battery Limits		281110	th.
Manpower		200	men
Capacity		500000	tons/year
Inputs		unit/year	
coal	2.488	1244000	tons
air	5.218	2609000	th.m3
electric energy	210.000	105000000	kWh
water	16.000	8000000	m3
steam	1.450	725000	Gcal
Outputs		unit/year	
superior alcohols	0.037	18500	tons
methanol	1.000	500000	tons
sulphur	0.032	16000	tons
ashes	0.335	167500	tons

TABLE B.8:Information sheet for HTL technology.

General			
Battery Limits		352890	th.
Manpower		200	men
Capacity		500000	tons/year
Inputs		unit/year	
coal	2.633	1316500	tons
air	3.144	1572000	th.m3
electric energy	120.000	60000000	kWh
water	5.000	2500000	m3
steam	2.400	1200000	Gcal
Outputs		unit/year	
methanol	1.000	500000	tons
superior alcohols	0.037	18500	tons
sulphur	0.034	17000	tons
ashes	0.319	159500	tons

TABLE B.9:Information sheet for KT. technology.

General			
Battery Limits		370830	th.
Manpower		200	men
Capacity		500000	tons/year
Inputs		unit/year	
lignite	3.837	1918500	tons
air	3.771	1885500	th.m3
electric energy	120.000	60000000	kWh
water	5.000	2500000	m3
steam	2.660	1330000	Gcal
Outputs		unit/year	
methanol	1.000	500000	tons
superior alcohols	0.037	18500	tons
sulphur	0.027	13500	tons
ashes	0.422	211000	tons

APPENDIX C.

The way of intensive parameter calculations is explained here. Although it constitutes only an example, the below described method of cost approximation is widely used for the development problem; we assume that members of the committee are acquaitanted with this method and agree with it, at least generally. If they disagree, they can always modify their scores for all the attributes corresponding to the production cost.

UCo Unit Cost is calculated for one ton of methanol based on the prices and consumption coefficients, the labor cost and the battery limits construction cost (see appendices A and B) for all raw materials. The whole procedure looks like:

$$UCo = a \times material\_cost + b \times labor\_cost + c \times blcc\_cost \tag{C.1}$$

The unit material cost is calculated as follows:

$$material\_cost = \sum (price \times consumption\_coef.) \tag{C.2}$$

The coefficients a, b, c in eq. C.1 are obtained from well-known factors the values of which are imposed by the local conditions.

$$a = 1 + general\_overhead \tag{C.3}$$

$$b = 1 + general\_overhead + direct\_overhead \tag{C.4}$$

$$c = maintenance\_cost \times (1 + general\_overhead) + insurance\&tax + blcc\_depreciation \tag{C.5}$$

UIn Unit Income is a difference between the unit value of production (see equation below) and Unit Cost. Methanol as well as all by-products are taken into account.

$$unit\_production\_value = \sum (price \times yield\_coef.) \tag{C.6}$$

RoR Rate of Return arises as a relative factor of the shape:

$$RoR = \frac{UIn \times capacity}{battery\_limits} \tag{C.7}$$

Pro Productivity is calculated using eq. C.6 as follows:

$$Pro = \frac{unit\_production\_value \times capacity}{battery\_limits} \tag{C.8}$$

TEf Thermal Efficiency is a relative factor calculated on the base of the lower heating value as the energy equivalent for chemicals. Exact values are reproduced in appendix A. The energy equivalent of methanol and by-products is divided by the energy equivalent of raw materials.

$$TEf = \frac{\sum \; (\; heating\_value \times yield\_coef. \;)}{\sum \; (\; heating\_value \times consumption\_coef. \;)} \tag{C.9}$$

B/L This parameter is calculated as follows:

$$B/L = \frac{battery\_limits}{manpower} \tag{C.10}$$

It is assumed in the calculations above that the investment cost is sufficiently represented by the battery limits construction cost.