Contributions to Methodology and Techniques of Decision Analysis (First Stage)

Andrzej Ruszczyński, Tadeusz Rogowski,
Andrzej P. Wierzbicki
Editors

CP-90-008
November 1990

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

Telephone: (0 22 36) 715 21 •0 □ Telex: 079 137 iiasa a □ Telefax: (0 22 36) 71313

Foreword

This collaborative volume reports on the results of a contracted study agreement between the System and Decision Analysis Program and its project on the Methodology of Decision Analysis at IIASA and a group of Polish institutes working in this area. This group includes: the Institute of Automatic Control, Warsaw University of Technology (coordinating the study on the Polish side), the Systems Research Institute of the Polish Academy of Sciences in Warsaw, the Institute of Computing Science of the Technical University of Poznan, the Institute for Control and Systems Engineering of the Academy of Mining and Metallurgy in Cracow, and the Institute of Informatics, University of Warsaw.

The study includes research in four directions: mathematical programming techniques for decision support, applications of decision support systems new methodological developments in decision support, dissemination of results and educational activities. The papers in this volume are organized in parts somewhat differently.

Part 1. *Theoretical and Methodological Contributions* contains four papers on mathematical programming for decision support and three papers on methodological issues. In the first four papers, specific features of certain mathematical programming problems are investigated in order to achieve fast and reliable computational algorithms that might be used in interactive decision support systems. Known algorithms for multistage stochastic programming are typically too slow for repeated optimization runs in interactive decision support; A. Ruszczyński in the first paper, investigates parallel decomposition methods for such problems. Irregular two-dimensional cutting problems result in discrete optimization of complexity characterized by strong NP-completeness; J. Błazewicz et al. in the second paper, analyze - between other issues - heuristic algorithms that lead to an implementation of a decision support system. Large problems of minimizing piece-wise quadratic functions with bounds on variables require special, fast computational algorithms, analyzed by J. Sosnowski[1] in the third paper; this class of problems has found applications in the multiobjective linear and dynamic programming package HYBRID. Multiobjective linear programming problems with fuzzy coefficients and bounds are an especially useful way of describing uncertainty in multiobjective decision support; the paper of P. Czyżak and R. Słowiński describes an interactive package for solving such types of problems.

The second group of methodological papers starts with the contribution of E. Nawarecki and G. Dobrowolski towards applications of min/max estimates for decision making under uncertainty. The next paper, by J. Paczyński, examines the issue of using indexed variables in user-friendly model generators, that is, preprocessors that help to define mathematical models of decision situations in a natural, user-friendly format. Another paper, by J. Granat, investigates the opposite side of interaction of a user with a decision support system - the issue of various graphic tools for representing the results of computerized decision analysis to the user.

Part II *Examples of prototype decision support systems (DSS) and applications* contains five papers on various prototypes of decision support systems together with their applications and two papers on special applications of such systems. Three papers give doc-

[1]This paper was published as the IIASA Collaborative Paper CP-90-003

umentation for enhanced versions of three prototype DSS developed during the previous scientific cooperation documented in IIASA Working Paper WP-88-071 (later published as a volume in Springer Verlag) *Theory, Software and Testing Examples in Decision Support Systems.* These are software packages: IAC-DIDAS-N (for nonlinear multiobjective programming models in decision support) by T. Kreglewski et al., DINAS (for multiobjective network analysis) by W. Ogryczak et al., and MCBARG (for multiobjective analysis of multi-person bargaining situations) by L. Kruś et al.[2]. The next two papers concern prototype decision support systems for multiobjective project scheduling (which took also part in an international comparative study on this subject, organized by IIASA): MPS by R. Słowiński et al. and MIPS by T. Ryś et al. In all these papers, examples of applications are also included. The last two papers are directly concerned with applications of DSS: multiobjective optimization of water releases from a retention reservoir during a flood by A. Karbowski and multiobjective design of a multipurpose batch plant in chemical industry by T. Ryś.

The study was supported partly by IIASA national currency funds in Poland, partly by the funds of a research program R.P.I.02 by the Polish Ministry of Education. It is a part of a long-term cooperation of the Systems and Decision Sciences Program with the Polish scientific community working on problems of optimization and decision support.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program.

[2]This paper was published as the IIASA Collaborative Paper CP-90-006

Contents

Contributions to Methodology and Techniques of Decision Analysis (First Stage)

Andrzej Ruszczyński, Tadeusz Rogowski,
Andrzej P. Wierzbicki
Editors

Part I
Theoretical and Methodological Contributions

Parallel Decomposition Methods for Linear Multistage Stochastic Programming Problems Theory and Computational Results

Andrzej Ruszczyński

Institute of Automatic Control

Warsaw University of Technology

00-665 Warsaw, Poland

Abstract

A new decomposition method for multistage stochastic linear programming problems is proposed. A multistage stochastic problem is represented in a tree-like form and with each node of the decision tree of a a certain linear or quadratic subproblem is associated. The subproblems generate proposals for their successors and some backward information for their predecessors. The subproblems can be solved in parallel and exchange information in an asynchronous way through special buffers. After a finite time the method either finds an optimal solution to the problem or discovers its inconsistency. An analytical illustrative example shows that parallelization can speed up computation over every sequential method. Computational experiments indicate that for large problems we can obtain substantial gains in efficiency with moderate numbers of processors.

Keywords: Stochastic Programming, Dynamic Programming, Decomposition, Parallel Computing.

1 Introduction

Parallel computing systems offer increased computing power but require new methods that take advantage of their capabilities. The principal objective of this paper is to present an algorithm that allows parallel decomposition of a class of particularly difficult optimization problems: *multistage stochastic linear programming problems.*

Let Ω be a finite probability space with elementary events ω and probabilities p_ω. Next, let $D_\omega(t)$ and $H_\omega(t)$, $t = 1,\ldots,T$ be sequences of random $m_b \times m_x$ matrices and $b_\omega(t)$ and $c_\omega(t)$, $t = 1,\ldots,T$, be sequences of random vectors in R^{m_b} and R^{m_x}, respectively. We shall call each sequence $s_\omega(t) = (D_\omega(t), H_\omega(t), b_\omega(t), c_\omega(t))$ corresponding to some event $\omega \in \Omega$ a *scenario*. The problem is to find a sequence $x_\omega(t)$, $t = 1,\ldots,T$, $\omega \in \Omega$, of

random vectors in R^{m_s} (a *policy*), which minimizes the linear form

$$\sum_{\omega\in\Omega} p_\omega \sum_{t=1}^{T} c_\omega(t)^* x_\omega(t) \tag{1.1}$$

subject to the constraints

$$D_\omega(t)x_\omega(t-1) + H_\omega(t)x_\omega(t) = b_\omega(t),\ t = 1,\ldots,T,\ \omega \in \Omega, \tag{1.2}$$

$$x_\omega(t) \geq 0,\ t = 1,\ldots,T,\ \omega \in \Omega, \tag{1.3}$$

with $x(0) = x_0$ fixed, and an additional *nonanticipativity constraint*, which can be formulated as follows: for all $\omega^1, \omega^2 \in \Omega$ and any $t \in \{1,\ldots,T\}$

$$x_{\omega^1}(t) = x_{\omega^2}(t) \ \ \textit{if} \ \ s_{\omega^1}(\tau) = s_{\omega^2}(\tau) \ \ \textit{for} \ \ \tau = 1,\ldots,t. \tag{1.4}$$

In other words, decisions corresponding to scenarios which are indistinguishable up to time t should be equal (see [23] for an extensive discussion of this issue).

Two important special cases of (1.1)-(1.4) are the *deterministic control problem* (with one scenario) and the *two-stage stochastic programming problem* ($T = 2$, $s_\omega(1)$ identical for all $\omega \in \Omega$).

Although in principle (1.1)-(1.4) is a linear programming problem, its size may be too large for standard linear programming techniques [19]. For this reason a variety of specialized approaches have been developed for the two cases mentioned earlier.

The first group are special versions of general-purpose LP methods which take advantage of the structure of the constraint matrix of the problem to improve basis factorization techniques and pricing strategies in the simplex method [5, 9, 10, 12, 15, 20, 30, 26, 32] or direction-finding in interior point methods [17].

The second group are techniques coming down from the decomposition principle of Dantzig and Wolfe [3, 6, 7, 14, 31, 32, 33].

The third group are nonlinear methods specialized to this particular class of problems: the finite generation method [22], the progressive hedging algorithm [23] and the regularized decomposition method [24, 25, 27, 28]. The last one is of special interest for us, because it shares the finite convergence property of pure linear approaches.

The objective of our paper is twofold. First, we shall extend the regularized decomposition method to multistage stochastic programs, while retaining properties observed in the two-stage case. Secondly, we shall show that the subproblems into which (1.1)-(1.4) is decomposed can be solved in parallel and can exchange information in an asynchronous manner. We hope that this is of interest in its own right and brings new quality even to the earlier two-stage version of [24]: the subproblems *and* the master can operate in parallel. In the multistage case our approach may mitigate the effort required by nested formulations [3, 14, 33] by allowing fast transmission of information between the stages. For computers on which true multitasking is not yet possible our results eliminate restrictions on the order in which the subproblems are processed. Similar observations have been also made in a recently completed dissertation [8], where parallel decomposition of staircase linear programs was discussed. For the progressive hedging algorithm of [23] a parallel implementation was described in [18].

In section 2 we restate the problem in a tree-like form and give a general outline of the method. In section 3 we study in detail fundamental objects of our method: regularized subproblems and we describe how they generate information for the other subproblems. Section 4 contains a formal description of the method and in section 5 we prove its finite

convergence. In section 6 we analyse a simple example and illustrate the operation of three methods: the nested decomposition method, the optimal serial algorithm and the parallel method. Finally, in section 7 we present computational results obtained for a number of test problems, both deterministic and stochastic, with synchronous and asynchronous versions of our method.

2 Outline of the method

More insight into the structure of problem (1.1)-(1.4) can be gained by restating it in a tree-like form. Namely, with the set of scenarios $s_\omega(t)$, $t = 1,\dots,T$, $\omega \in \Omega$, we can associate a tree $\mathcal{T} = \{\mathcal{N}, \mathcal{A}\}$, where $\mathcal{N}$ is a set of nodes and $\mathcal{A}$ is a set of arcs of $\mathcal{T}$. The set of nodes $\mathcal{N}$ is divided into subsets (levels) $\mathcal{N}_t$, $t = 1,\dots,T$; the nodes $n \in \mathcal{N}_t$ at level t correspond to different subscenarios $\{s^n(1),\dots,s^n(t)\}$. At level 1 there are so many nodes as many different realizations of $s(1)$ can occur; at level 2 the nodes correspond to different pairs $\{s(1), s(2)\}$, etc. The number of nodes at level T is equal to the number of scenarios $|\Omega|$. The arcs join nodes from neighboring levels in such a way that a node n at level t corresponding to subscenario $s^n = \{s^n(1),\dots,s^n(t)\}$ is connected with all nodes m at level $t+1$ whose subscenarios $s^m = \{s^m(1),\dots,s^m(t+1)\}$ equal s^n up to time t. Let us denote by $\pi(n)$ the *predecessor* of node n, i.e. the node at the previous level with which n is connected and by $\mathcal{S}(n)$ the set of *successors* of n, $\mathcal{S}(n) = \{m : n = \pi(m)\}$. Next, let $\Pi(n) = \{n, \pi(n), \pi(\pi(n)), ...\}$ be the *path* from n to level 1. Taking account of the fact that for $n \in \mathcal{N}_t$ we have $s^n = \{s^{\pi(n)}, s^n(t)\}$, it is sufficient to associate with each node $n \in \mathcal{N}_t$ only the last element of its subscenario, $s_n = s^n(t)$; the whole subscenario can be recovered by backtracking the path $\Pi(n)$.

A node n at level t corresponds to the bundle Ω_n of scenarios which are indistinguishable up to time t. By the nonanticipativity condition (1.4) all decisions $x_\omega(t)$, $\omega \in \Omega_n$ must be equal. We denote their value by x_n.

To complete the reformulation of the problem, with nodes $n \in \mathcal{N}$ we shall associate probabilities $\bar{p}_n$ defined as follows: for each *terminal node* $n \in \mathcal{N}_T$ we set $\bar{p}_n = p_\omega$, where $\omega \in \Omega$ is the event that corresponds to leaf n. For other nodes we define $\bar{p}_n = \sum_{m \in \mathcal{S}(n)} \bar{p}_m$.

Using this notation we can rewrite (1.1)-(1.4) as follows:

$$\textit{minimize} \sum_{n \in \mathcal{N}} \bar{p}_n c_n^* x_n \tag{2.1}$$

$$D_n x_{\pi(n)} + H_n x_n = b_n,\ n \in \mathcal{N}, \tag{2.2}$$

$$x_n \geq 0,\ n \in \mathcal{N}, \tag{2.3}$$

where for $n \in \mathcal{N}_1$ we define $x_{\pi(n)} = x_0$. We shall assume throughout this paper that (2.1)-(2.3) is bounded.

The tree structure makes it possible to develop a hierarchical method for solving (2.1)-(2.3). For each pair of nodes (m, n), $m \in \mathcal{S}(n)$, we define the conditional probability $p_{mn} = \bar{p}_m / \bar{p}_n$ and use it for recursive definition of the *value function*

$$f_n(x_{\pi(n)}) = min\ \{c_n^* x_n + \sum_{m \in \mathcal{S}(n)} p_{mn} f_m(x_n) :\ H_n x_n = b_n - D_n x_{\pi(n)},\ x_n \geq 0\}. \tag{2.4}$$

Solving the problem is equivalent to calculating

$$\sum_{n \in \mathcal{N}_1} \bar{p}_n f_n(x_0).$$

Since each component of this sum can be computed independently, with no loss of generality we can assume that there is only one node $n = 1$ at level 1 and our aim is to find $f_1(x_0)$. This can be done by the *nested decomposition method*: a recursive procedure of dynamic programming type in which problems (2.4) at various levels of recursion are solved by a cutting plane method (cf. [3, 14, 33]).

We shall modify the hierarchical approach in two directions.

First, instead of the pure cutting plane method we admit the use (for some nodes n) of its regularized version analysed in the two-stage case in [24, 25]. With each node n of the tree $\mathcal{T}$, except for the terminal nodes, we associate the following *regularized subproblem*

$$\textit{minimize}\ \ \eta_n = \tfrac{1}{2}\rho_n\|x_n - \xi_n\|^2 + c_n^* x_n + \sum_{m\in S(n)} p_{mn}\bar{f}_m(x_n) \tag{2.5}$$

$$H_n x_n = b_n - D_n x_{\pi(n)}, \tag{2.6}$$

$$x_n \geq 0. \tag{2.7}$$

Here $\rho_n \geq 0$ is a penalty coefficient, ξ_n is a certain regularizing point and $\bar{f}_m(\cdot)$, $m \in S(n)$ are convex piecewise linear outer approximations of the value functions $f_m(\cdot)$:

$$\bar{f}_m(x_n) \leq f_m(x_n)\ \ \textit{for all}\ \ x_n. \tag{2.8}$$

If $\rho_n = 0$ then (2.5)-(2.7) becomes identical with the master problem of the *multicut method* of [4] (a special version of Benders decomposition for stochastic programs) and for $\rho_n > 0$ we get the master problem of the *regularized decomposition method* of [24, 25]. The use of proximal terms $\frac{1}{2}\rho_n\|x_n - \xi_n\|^2$ in (2.5), where ξ_n is the best point found so far, stabilizes the sequence of trial points x_n generated by (2.5)-(2.7). For larger problems, where identification of essential pieces of $\bar{f}_m(\cdot)$ takes many tries, regularization may save time, especially when a good initial approximation to the solution is available [24, 25]. Our analysis will treat the general case covering the purely linear Benders decomposition and the regularized version.

With each terminal node $n \in \mathcal{N}_T$ of $\mathcal{T}$ we associate the linear problem

$$\textit{minimize}\ \ l_n = c_n^* x_n \tag{2.9}$$

$$H_n x_n = b_n - D_n x_{\pi(n)}, \tag{2.10}$$

$$x_n \geq 0. \tag{2.11}$$

In the method we link subproblems (2.5)-(2.7) and (2.9)-(2.11) in the same way in which the nodes of $\mathcal{T}$ are linked. They exchange information along the arcs by passing the solutions x_n to their successors and receiving some backward information used to correct the approximations $\bar{f}_m(\cdot)$. The backward information has the form of *cuts*, i.e. some linear functions used to describe pieces of $\bar{f}_m(\cdot)$ or facets of their domains.

Our principal objective, however, is parallelization. In our method we allow *all* subproblems to be solved in a parallel asynchronous manner. Their logical dependence, implied by the tree structure of the problem, is reflected only in the communication structure of the distributed method, but does not condition the order in which the subproblems are processed. To this end we separate subproblems by buffers which store primal solutions passed from antecedent problems and cuts generated by the successors. Each subproblem takes some (possibly outdated) information from the buffers, generates its primal solution and a cut, passes them to the neighboring buffers, etc., until no new information appears.

We shall discuss all these issues in sections 3 and 4, but let us at first illustrate the structure of the method on two typical examples.

Example 1. Consider the deterministic dynamic problem

$$\textit{minimize} \ \sum_{t=1}^{T} c_t^* x_t$$

$$D_t x_{t-1} + H_t x_t = b_t, \ t = 1, \ldots, T,$$

$$x_t \geq 0, \ t = 1, \ldots, T.$$

Graph $\mathcal{T}$ is in this case a chain and the corresponding network of subproblems and buffers takes on the form shown in Figure 1. It corresponds to the nested decomposition method, but our subproblems are quadratic and are solved in parallel thus allowing for fast exchange of information between the stages (see [1, 2] for another parallel approach to dynamic programming).

Example 2. Consider now the stochastic two-stage problem

$$\textit{minimize} \ c_1^* x_1 + \sum_{l=2}^{L} p_l c_l^* x_l$$

$$H_1 x_1 = b_1,$$

$$D_l x_1 + H_l x_l = b_l, \ l = 2, \ldots, L,$$

$$x_l \geq 0, \ l = 1, 2, \ldots, L.$$

Graph $\mathcal{T}$ is a star with root 1 and leaves $2, \ldots, L$. The corresponding network of subproblems and buffers is shown in Figure 2. It is similar to the structure of the Dantzig-Wolfe method, but our master is different, and the master *and* the subproblems are solved in parallel, which significantly differs our approach from that of [13].

For stochastic dynamic problems the structure of the network of subproblems is a combination of these two extreme cases.

We end this section by stressing that the fact that we distinguish tasks which can be solved in a parallel asynchronous manner does not imply that we need different processors for different tasks. It simply leaves us full freedom in assigning tasks to processors.

3 Cuts

Let $a_{mj} + g_{mj}^* x_n$, $j \in J_m$, be a collection of linear functions such that

$$f_m(x_n) \geq a_{mj} + g_{mj}^* x_n, \ \textit{for all} \ x_n, \ j \in J_m^1, \tag{3.1}$$

and

$$\text{dom } f_m \subseteq \{x_n : a_{mj} + g_{mj}^* x_n \leq 0\}, \ j \in J_m^2, \tag{3.2}$$

where J_m^1 and J_m^2 are disjoint subsets of J_m. We shall call (3.1) *objective cuts* and (3.2) *feasibility cuts*. The cuts can be used to define functions $\bar{f}_m$ in (2.5) as follows: if x_n satisfies the feasibility cuts we set

$$\bar{f}_m(x_n) = \min\{v_{mn} : v_{mn} \geq a_{mj} + g_{mj}^* x_n, \ j \in J_m^1\};$$

otherwise we set $\bar{f}_m(x_n) = +\infty$. It is clear that $\bar{f}_m$ is convex and piecewise linear and satisfies (2.8).

Using the cuts we can reformulate (2.5)-(2.7) in a more explicit fashion. Let us introduce aggregate vectors and matrices: $p_n = (p_{mn})_{m\in S(n)}$, $v_n = (v_{mn})_{m\in S(n)}$, $a_n = (a_{mj})_{m\in S(n), j\in J_m}$, $G_n = (g_{mj})_{m\in S(n), j\in J_m}$. With this notation (2.5)-(2.7) can be equivalently formulated as follows:

$$\textit{minimize}\ \ \eta_n = \tfrac{1}{2}\rho_n\|x_n - \xi_n\|^2 + c_n^* x_n + p_n^* v_n \tag{3.3}$$

$$a_n + G_n^* x_n - E_n^* v_n \leq 0, \tag{3.4}$$

$$H_n x_n = b_n - D_n x_{\pi(n)}. \tag{3.5}$$

Here E_n is a zero-one matrix, whose j-th column has 1 at position corresponding to v_{mn} if the j-th cut in (3.4) is an objective cut for the function $f_m(\cdot)$. The columns corresponding to feasibility cuts are zero. For simplicity we include direct constraints (2.7) into (3.4) as feasibility cuts. We assume that there is at least one cut for each $f_m(\cdot)$, $m \in S(n)$, among (3.4), so that E_n has full row rank.

To describe the way in which cuts for the predecessor can be generated let us fix our attention on a specific class of methods for solving (3.3)-(3.5): the *active set methods* which proved useful for linear quadratic problems of similar structure (cf. [16, 24, 25]). Their main idea is to choose a subset of linearly independent constraints from (3.4)-(3.5), solve the equality constrained subproblem obtained and revise the active set if optimality conditions for the whole problem are not satisfied. In case of $\rho_n = 0$ the methods reduce to the dual simplex method. For $\rho_n > 0$ the algorithms are more involved, because the number of active constraints may vary, but the simplicity of the quadratic term in (3.3) and the special form of E_n allow efficient implementation [24, 25].

Each active set defines some submatrices G, E, H, D of G_n, E_n, H_n, D_n and subvectors a, b of a_n, b_n, which are used in equality constraints:

$$G^* x_n - E^* v_n = -a, \tag{3.6}$$

$$H x_n = b - D x_{\pi(n)}. \tag{3.7}$$

The necessary and sufficient conditions of optimality for (3.3),(3.6), (3.7) have now the form:

$$E\lambda = p_n,$$

$$\rho_n x_n + G\lambda + H^*\mu = \rho_n\xi_n - c_n.$$

We can always choose an active set so that E is of full row rank and $\begin{bmatrix} E \\ G \end{bmatrix}$ H^*] is of full column rank. There can be many specific ways in which the active set can be altered [16, 24, 25], but there are always only two possible situations in which the method terminates: optimality with $\lambda \geq 0$ and (x_n, v_n) satisfying (3.4) and (3.5), or inconsistency of the active cuts with a certain inactive cut. These two cases determine the type of information that can be passed to the preceding problem.

Lemma 1 *Let (3.3)-(3.5) be solvable for some $x_{\pi(n)}$ with the final active set (3.6)-(3.7). If the system of equations*

$$E\lambda = p_n, \tag{3.8}$$

$$G\lambda + H^*\mu = -c_n, \tag{3.9}$$

has a solution (λ, μ) *with* $\lambda \geq 0$, *then*

$$f_n(x_{\pi(n)}) \geq g^* x_{\pi(n)} + \alpha \ \ for\ all\ \ x_{\pi(n)}, \tag{3.10}$$

where

$$g = D^* \mu, \tag{3.11}$$

$$\alpha = a^* \lambda - b^* \mu. \tag{3.12}$$

Proof. Consider the linear problem

$$minimize\ \ l_n = c_n^* x_n + p_n^* v_n \tag{3.13}$$

$$G^* x_n - E^* v_n \leq -a, \tag{3.14}$$

$$H x_n = b - D x_{\pi(n)}. \tag{3.15}$$

It is a relaxation of (2.4), so the optimal value satisfies for each $x_{\pi(n)}$ the inequality

$$\hat{l}_n(x_{\pi(n)}) \leq f_n(x_{\pi(n)}).$$

On the other hand $\lambda \geq 0$ and μ satisfying (3.8)-(3.9) form a feasible dual solution to (3.13)-(3.15). Thus for each $x_{\pi(n)}$

$$\hat{l}_n(x_{\pi(n)}) \geq \lambda^* a + \mu^* (D x_{\pi(n)} - b).$$

Combining the last two inequalities we obtain the required result.
Remark If $\rho_n = 0$ in (3.3) then the system (3.8)-(3.9) is always solvable and (λ, μ) are the optimal Lagrange multipliers associated with (3.4)-(3.5).

Lemma 2 *For a given set of constraints (3.4)-(3.5) the number of different objective cuts (3.10)-(3.12) is finite.*

Proof. Each cut (3.10)-(3.12), if it exists, is uniquely defined by the active set, and there can be only finitely many different active sets.

Lemma 3 *If the solution* x_n *to (3.3)-(3.5) at* $x^0_{\pi(n)}$ *is equal to* ξ_n, *then the cut (3.10)-(3.12) exists and supports the epigraph of the function*

$$\tilde{f}_n(x_{\pi(n)}) = min\{c_n^* x_n + \sum_{m \in S(n)} p_{mn} \tilde{f}_m(x_n) \ \mid \ H_n x_n = b_n - D_n x_{\pi(n)},\ x_n \geq 0\}$$

at $(x^0_{\pi(n)}, \tilde{f}_n(x^0_{\pi(n)}))$.

Proof. At $x_n = \xi_n$ the necessary and sufficient conditions of optimality for (3.13)-(3.15) and (3.3)-(3.5) are identical, so the cut must exist. Next, the constraints not included into the active set are satisfied at (x_n, v_n). Therefore $\hat{l}_n(x^0_{\pi(n)}) = \tilde{f}_n(x^0_{\pi(n)})$. Since (g, α) supports $\hat{l}_n(\cdot)$ at $x^0_{\pi(n)}$ it supports $\tilde{f}_n(\cdot)$ at $x^0_{\pi(n)}$, too.

Lemma 4 *Suppose that (3.4)-(3.5) are inconsistent for some* $x^0_{\pi(n)}$. *Then there exists an active set (3.6)-(3.7) such that one of the following conditions holds.*

(i) *There is a feasibility cut* $\alpha + g^* x_n \leq 0$ *among (3.4) and multipliers* $\lambda \geq 0$ *and* μ *such that*

$$E\lambda = 0, \tag{3.16}$$

$$g + G\lambda + H^*\mu = 0, \tag{3.17}$$

$$\alpha + \lambda^* a + \mu^*(Dx^0_{\pi(n)} - b) > 0. \tag{3.18}$$

(ii) *There is an equation* $hx_n = \beta - dx_{\pi(n)}$ *among (3.5) and multipliers* $\lambda \geq 0$, μ *and* $\epsilon = \pm 1$ *such that*

$$E\lambda = 0, \tag{3.19}$$

$$\epsilon h + G\lambda + H^*\mu = 0, \tag{3.20}$$

$$\epsilon(dx^0_{\pi(n)} - \beta) + \lambda^* a + \mu^*(Dx^0_{\pi(n)} - b) > 0. \tag{3.21}$$

Proof. Suppose that the cut $\alpha + g^* x_n \leq 0$ is violated at the solution of the equality constrained subproblem and cannot be introduced into the active set. Then (3.16)-(3.18) with $\lambda \geq 0$ follow from [21, thm. 22.1]. If an equality constraint $hx_n = \beta - dx_{\pi(n)}$ is inconsistent with active cuts, in a similar way we get (3.19)-(3.21).

Using lemma 4 we can obtain cuts which must be satisfied by any $x_{\pi(n)}$. If case (i) holds, multiplying (3.14) by λ^* and adding (3.15) multiplied by μ^* we see that

$$\lambda^* G^* x_n + \mu^* H x_n - \lambda^* E^* v_n \leq -\lambda^* a + \mu^*(b - Dx_{\pi(n)}),$$

and, since x_n must satisfy $\alpha + g^* x_n \leq 0$,

$$\alpha + \lambda^* a + \mu^*(Dx_{\pi(n)} - b) \leq 0.$$

In case (ii) in a similar fashion we obtain the cut

$$\lambda^* a + \epsilon(dx_{\pi(n)} - \beta) + \mu^*(Dx_{\pi(n)} - b) \leq 0.$$

The new cut is violated at $x^0_{\pi(n)}$. These two cases can be put in one format

$$\bar{\lambda}^* a_n + \bar{\mu}^*(D_n x_{\pi(n)} - b_n) \leq 0, \tag{3.22}$$

by assigning zero multipliers to inactive cuts and multiplier ± 1 to the violated cut. We can summarize it in the following lemma.

Lemma 5 *At any* $x^0_{\pi(n)}$ *for which (3.4)-(3.5) are inconsistent we can construct by (3.22) a feasibility cut*

$$\bar{\alpha} + \bar{g}^* x_{\pi(n)} \leq 0, \tag{3.23}$$

$$\bar{g} = D_n^* \bar{\mu}, \tag{3.24}$$

$$\bar{\alpha} = a_n^* \bar{\lambda} - b_n^* \bar{\mu}. \tag{3.25}$$

The number of such cuts possible is finite and they fully describe the set of $x_{\pi(n)}$ *for which (3.4)-(3.5) are consistent.*

Proof. Formulae (3.23)-(3.25) follow directly from (3.22). Each such cut is defined uniquely by the active set and the violated constraint, because (3.16)-(3.17) or (3.19)-(3.20) define uniquely (λ, μ) by the full column rank of $\begin{bmatrix} E \\ G \end{bmatrix}$ H^*]. The number of possible active sets for (3.4)-(3.5) is finite and for each active set there can be only finitely many violated constraints. Therefore, one can generate only finitely many cuts (3.23)-(3.25). If $x^0_{\pi(n)}$ satisfies them, then it must satisfy (3.4)-(3.5), since otherwise we would be able to construct a new cut by lemma 4. The proof is complete.

For problem (2.9)-(2.11) associated with a terminal node the cuts simplify slightly: there are no terms $a^*\lambda$ and $a_n^*\bar{\lambda}$ in (3.12) and (3.25).

4 Tasks

As we mentioned in section 2, our method for solving (2.1)-(2.3) consists of a number of tasks which can be executed in parallel and can exchange information in an asynchronous manner. With each node n of the tree $\mathcal{T}$ we asociate a task $SUB(n)$ whose function is to solve the regularized subproblem (3.3)-(3.5) corresponding to node n. The task $SUB(n)$ communicates with other tasks through two channels: $BOX(n)$ and $PIPE(n)$. Let us describe the channels and the tasks in more detail.

BOX(n)

In $BOX(n)$ the last solution x_n of (3.3)-(3.5) is stored. Only $SUB(n)$ may change its contents by overwriting x_n. The tasks $SUB(m)$ for $m \in \mathcal{S}(n)$ may read x_n without destroying it. If $BOX(n)$ is empty and $SUB(m)$ attempts to read x_n, $SUB(m)$ waits until there will be new information available.

PIPE(n)

Through $PIPE(n)$ cuts generated by the tasks $SUB(m)$, $m \in \mathcal{S}(n)$ are transmitted to $SUB(n)$. $PIPE(n)$ has a finite capacity which allows for storing at least one cut. When $SUB(n)$ takes a cut from $PIPE(n)$, the cut is deleted and new space in $PIPE(n)$ is created. If $PIPE(n)$ is full, the tasks $SUB(m)$, $m \in \mathcal{S}(n)$ which attempt to put cuts to $PIPE(n)$, wait until room for the next cut will be available.

The tasks $SUB(n)$ have three different forms: for the starting node, for terminal nodes $n \in \mathcal{N}_T$ and for intermediate nodes. $SUB(n)$ operates in two modes: 'go' and 'optimal' and updates the solution of (3.3)-(3.5) each time new information is available in the buffers. To simplify our description we asume that at the beginning every $SUB(n)$, $n \notin \mathcal{N}_T$, has at least one objective cut for each $f_m(\cdot)$, $m \in \mathcal{S}(n)$ (e.g. $f_m(\cdot) \geq -C$ for a sufficiently large C). The tasks start in mode 'go'.

We start the description of active tasks from the first task which is responsible for detecting optimality or infeasibility and terminating the whole method.

SUB(1)

Step 1. Get a cut from $PIPE(1)$. If $PIPE(1)$ is empty then go to Step 4; otherwise go to Step 2.

Step 2. Solve the subproblem (3.3)-(3.5) and delete from (3.4) the cuts that were inactive at the solution. If (3.3)-(3.5) was infeasible then go to Step 7. If (3.3)-(3.5) was solvable then go to Step 3.

Step 3. Write x_1 into $BOX(1)$ and go to Step 1.

Step 4. If the tasks $SUB(m)$ for all $m \in \mathcal{S}(1)$ read the last x_1 from $BOX(1)$ and are in mode 'optimal' and $PIPE(1)$ is still empty, then go to Step 5; otherwise go to Step 1.

Step 5. If $\rho_n > 0$ and $x_1 \neq \xi_1$ then set $\xi_1 \Leftarrow x_1$ and go to Step 2; otherwise go to Step 6.

Step 6. Terminate (*optimal solution found*).

Step 7. Terminate (*the problem is infeasible*).

Before proceeding to the other cases let us briefly comment on the above algorithm. There is only one external source of changes in the solution of (3.3)-(3.5): new cuts. Only when no new cuts can be expected, because the sons are in mode 'optimal' (Step 4), we update the regularizing point ξ_1. If this is exploited too, $SUB(1)$ terminates. If (3.3)-(3.5) has no feasible solutions, the original problem is infeasible, because the fasibility cuts approximate the domains of $f(m)$, $m \in \mathcal{S}(1)$ from outside.

SUB(n) for $n \neq 1$ and $n \notin \mathcal{N}_T$

Step 1. Read $x_{\pi(n)}$ from $BOX(\pi(n))$.

Step 2. Get a cut from $PIPE(n)$.

Step 3. If $x_{\pi(n)}$ did not change and $PIPE(n)$ was empty, go to Step 4; otherwise set mode to 'go' and go to Step 5.

Step 4. If mode='optimal' go to Step 1; otherwise go to Step 8.

Step 5. Solve the subproblem (3.3)-(3.5) and delete from (3.4) the cuts that were inactive at the solution. If (3.3)-(3.5) was infeasible then go to Step 6. If (3.3)-(3.5) was solvable then go to Step 7.

Step 6. Clear $BOX(n)$, generate the feasibility cut (3.22), put it into $PIPE(\pi(n))$ and go to Step 1.

Step 7. Write x_n into $BOX(n)$ and generate the objective cut (3.10)-(3.12), if possible. If the objective cut exists then put it into $PIPE(\pi(n))$. Go to Step 1.

Step 8. If the tasks $SUB(m)$ for all $m \in \mathcal{S}(n)$ read the last x_n from $BOX(n)$ and are in mode 'optimal' and $PIPE(n)$ is still empty, then go to Step 9; otherwise go to Step 1.

Step 9. If $\rho_n > 0$ and $x_n \neq \xi_n$ then set $\xi_n \Leftarrow x_n$ and go to Step 5; otherwise change mode to 'optimal' and go to Step 1.

Now there are two external sources of changes in the solution of (3.3)-(3.5): changes in $x_{\pi(n)}$ and new cuts. Only if both possibilities are exploited, we update the regularizing point ξ_n. If this is exploited too, we change the mode to 'optimal' to let our predecessor know that nothing new can be expected from us.

Each $SUB(n)$ processes many cuts and most of them become soon outdated. However, owing to the deletion rule of Step 5 (Step 2 for $SUB(1)$), the size of (3.3)-(3.5) is bounded. The set of cuts that are stored (the *committee*) need not have more than $m_x + |\mathcal{S}(n)| + 1$ members : no more than $m_x + |\mathcal{S}(n)|$ active cuts and one new cut read from $PIPE(n)$. A specialized algorithm for updating the solution of (3.3)-(3.5) when a new cut is added has been developed in [24, 25].

The tasks for terminal nodes are simpler: there are no cuts to process and the problem is linear.

SUB(n) for $n \in \mathcal{N}_T$

Step 1. Read $x_{\pi(n)}$ from $BOX(\pi(n))$.

Step 2. If $x_{\pi(n)}$ is different from the last $x_{\pi(n)}$ set mode to 'go' and go to Step 3; otherwise set mode to 'optimal' and go to Step 1.

Step 3. Solve the subproblem (2.9)-(2.11). If (2.9)-(2.11) was solvable then go to Step 4; otherwise go to Step 5.

Step 4. Generate the objective cut (3.10)-(3.12), put it into $PIPE(\pi(n))$ and go to Step 1.

Step 5. Generate the feasibility cut (3.22), put it into $PIPE(\pi(n))$ and go to Step 1.

If $SUB(1)$ terminates, all other tasks terminate, too; their last solutions contain then the solution to the original problem.

5 Convergence

Our aim in this section is to prove that the method after a finite time either discovers inconsistency in the problem or finds its optimal solution (recall that we assume throughout this paper that the problem is bounded). We shall use τ to denote time that passed from the start of the method.

To avoid deadlocks and races we shall need two additional assumptions.

(A1) If a new x_n is written into $BOX(n)$, then after a finite time each $SUB(m)$, $m \in \mathcal{S}(n)$ will get acess to $BOX(n)$.

(A2) If $SUB(m)$ for $m \in \mathcal{S}(n)$ reads x_n from $BOX(n)$, then the mode of $SUB(m)$ is changed to 'go' before $SUB(n)$ checks it at Step 8.

Let us introduce two notions concerning asymptotic behavior of our subproblems.

Definition 1 *We say that $SUB(n)$ for $n \neq 1$ is* stable from above *if there exists a finite time τ_n such that the contents of $BOX(\pi(n))$ does not change for $\tau \geq \tau_n$. The task $SUB(1)$ is* stable from above *if it is feasible for all $\tau \geq 0$.*

Definition 2 *We say that $SUB(n)$ is* terminally optimal *if there exists a finite time $\hat{\tau}_n$ such that $SUB(n)$ stays in mode 'optimal' for all $\tau \geq \hat{\tau}_n$.*

We are now ready to carry out our analysis. We shall at first assume that $\rho_n > 0$ for $n \notin \mathcal{N}_T$.

Lemma 6 *Suppose that $SUB(n)$ is in mode 'optimal' at time τ. Then the tasks $SUB(m)$ for $m \in \mathcal{S}(n)$ ar in mode 'optimal' at time τ.*

Proof. Our assertion is trivial for terminal nodes $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in \mathcal{S}(n)$. We shall prove it for n. Let $SUB(n)$ be in mode 'optimal' at time τ. Then at some time $\tau_n \leq \tau$ $SUB(n)$ entered Step 8 (Step 4 for $n = 1$) and the tasks $SUB(m)$, $m \in \mathcal{S}(n)$ were at mode 'optimal' at time instants $\tau_m \in [\tau_n, \tau]$. Each $SUB(m)$ can change its mode only after receiving a new x_n from $BOX(n)$ or a new cut from $PIPE(m)$. In the interval $[\tau_m, \tau]$ the solution x_n does not change, because $SUB(n)$ stays in mode 'optimal'. Next, by our inductive assumption $SUB(j)$, $j \in \mathcal{S}(m)$ are in mode 'optimal', so $PIPE(m)$ remains empty. Consequently, $SUB(m)$, $m \in \mathcal{S}(n)$ stay in mode 'optimal' in the intervals $[\tau_m, \tau]$.

Lemma 7 *Suppose that $SUB(n)$ is in mode 'optimal' at time τ. Then $\bar{f}_m(x_n) = f_m(x_n)$ for all $m \in \mathcal{S}(n)$ and x_n solves the linear problem (2.4).*

Proof. Our assertion is obvious for terminal nodes $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in \mathcal{S}(n)$. We shall prove it for n. Let $\tau_n \leq \tau$ be the last time at which x_n changed. By lemma 6, all $SUB(m)$, $m \in \mathcal{S}(n)$ are in mode 'optimal' at time τ. On the other hand, by Step 3 each $SUB(m)$ changed its mode to 'go' at a certain $\tau_m \in [\tau_n, \tau]$. So, each $SUB(m)$, $m \in \mathcal{S}(n)$ executed at least once Step 5 in the time interval $[\tau_m, \tau]$. Let $\hat{\tau}_m$ be the last time in this interval at which Step 5 was executed by $SUB(m)$. Since $SUB(m)$ is in mode 'optimal' at τ we must have had $x_m = \xi_m$ at $\hat{\tau}_m$. By lemma 3 the last objective cut generated by $SUB(m)$ supported $\bar{f}_m(\cdot)$ at x_n. By our inductive assumption, for every $l \in \mathcal{S}(\Updownarrow)$ we had $\bar{f}_l(x_m) = f_l(x_m)$, so $\bar{f}_m(x_n) = f_m(x_n)$. Since $\bar{f}_m(\cdot) \leq f_m(\cdot)$, the cut supported $f_m(\cdot)$ at x_n. By *(A2)*, $SUB(n)$ did not stop before his successors read x_n. But x_n did not change in $[\hat{\tau}_m, \tau]$; hence $\bar{f}_m(x_n) \geq f_m(x_n)$. Since $\bar{f}_m \leq f_m$ we obtain $\bar{f}_m(x_n) = f_m(x_n)$ for all $m \in \mathcal{S}(n)$. Consequently, x_n solves (2.4). If $m \in \mathcal{S}(n)$ is a terminal node, the analysis is simpler, because each objective cut is then a supporting cut.

Lemma 8 *There are finitely many possible committees for each $SUB(n)$.*

Proof. Our assertion is true for terminal nodes $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in \mathcal{S}(n)$. We shall prove it for n. Each committee is a set of cuts generated by the tasks $SUB(m)$, $m \in \mathcal{S}(n)$. By our inductive assumption each successor of n may have only finitely many committees. By lemmas 2 and 5 each committee may define only finitely many cuts. Therefore only finitely many committees for $SUB(n)$ can be formed from these cuts.

Lemma 9 *Suppose that $SUB(n)$ is stable from above. Then ξ_n is changed only finitely many times.*

Proof. Let $x_{\pi(n)}$ be fixed for $\tau \geq \tau^0$ and let ξ_n be changed at time instants $\tau^k \geq \tau^0$, $k = 1, 2, \ldots$. Let x_n^k denote the solution to (3.3)-(3.5) at τ^k. By Step 9, the regularizing point in the interval $[\tau^k, \tau^{k+1}]$ is given by $\xi_n^{k+1} = x_n^k$. It is changed at τ^{k+1}, so $x_n^{k+1} \neq \xi_n^{k+1}$. Let

$$\hat{\tau}^k = \inf\{\tau \in [\tau^k, \tau^{k+1}] : \; x_n(\tau) = x_n^{k+1}\}.$$

Consider $SUB(m)$, $m \in \mathcal{S}(n)$. By Step 8 of $SUB(n)$, each $SUB(m)$ reads x_n^{k+1} at some time instant $\tau_m^k \geq \hat{\tau}^k$, changes mode to 'go', and reaches mode 'optimal' at some time instant $\sigma_m^k \in (\tau_m^k, \tau^{k+1}]$. By Lemma 7, at time σ_m^k we have $\bar{f}_l(x_m) = f_l(x_m)$ for all $l \in \mathcal{S}(\Updownarrow)$ and x_m solves (2.4) (with n replaced by m). Thus $\bar{f}_m(x_n^{k+1}) = f_m(x_n^{k+1})$. Summing up, at each time instant τ^k the following relations hold

(i) x_n^{k+1} solves (2.5)-(2.7) with $\xi_n = \xi_n^k$;

(ii) $\bar{f}_m(x_n^{k+1}) = f_m(x_n^{k+1})$ for all $m \in \mathcal{S}(n)$.

These two conditions imply that at τ^k an *exact serious step* of the regularized decomposition method of [24] for solving the problem

$$\text{minimize} \;\; F_n(x_n) = c_n^* x_n + \sum_{m \in \mathcal{S}(n)} p_{mn} f_m(x_m)$$

$$H_n x_n = b_n - D_n x_{\pi(n)},$$

$$x_n \geq 0$$

is executed ($x_{\pi(n)}$ is fixed). It follows from the theory developed in [24] that after finitely many such steps the minimum of F_n will be reached and no more steps will be possible.

Lemma 10 *If $SUB(n)$ is stable from above then all its successors $SUB(m)$ for $m \in \mathcal{S}(n)$ are stable from above.*

Proof. By lemma 9, ξ_n can be changed only finitely many times. Hence there is τ_0 such that for $\tau \geq \tau_0$ both $x_{\pi(n)}$ and ξ_n remain constant. The solution x_n to (3.3)-(3.5) does not change when inactive cuts are deleted. It is unique for a given committee, owing to the existence of the quadratic regularizing term in (3.3). Consequently, x_n may change only by introduction of a cut which cuts-off the previous solution. In this case the minimum value of (3.3)-(3.5) increases. By lemma 8 there can be only finitely many different committees at $SUB(n)$, which implies that x_n may be changed only finitely many times. The proof is complete.

Lemma 11 *If $SUB(n)$ is stable from above then it is terminally optimal.*

Proof. Our assertion is obvious for terminal nodes $n \in \mathcal{N}_T$. Suppose that it is true for all $m \in \mathcal{S}(n)$. We shall prove it for n. By lemma 10, the successors $SUB(m)$, $m \in \mathcal{S}(n)$ are stable from above. By our inductive assumption they are terminally optimal. Let τ_0 be such a time instant that for $\tau \geq \tau_0$ $x_{\pi(n)}$ and x_n do not change and $SUB(m)$, $m \in \mathcal{S}(n)$ are in mode 'optimal'. If $SUB(n)$ were in mode 'go' at some time $\tau \geq \tau_0$ it would have to enter Step 8. But x_n does not change for $\tau \geq \tau_0$, so we would have $x_n = \xi_n$, mode would be set to 'optimal' and $SUB(n)$ would start infinite cycling between Steps 1 and 4.

It is now easy to prove our main result.

Theorem 1 *After a finite time the method either discovers inconsistency in the problem and stops at Step 6 of $SUB(1)$ or finds an optimal solution and stops at Step 5 of $SUB(1)$. In the latter case the solution is given by x_n, $n \in \mathcal{N}$.*

Proof. Suppose that $SUB(1)$ is not stable from above. Then after a finite time it stops at Step 6 with inconsistent feasibility cuts. The cuts approximate the domain of $f_1(\cdot)$ from outside, so the problem is infeasible in this case. Suppose now that $SUB(1)$ is stable from above. By lemma 11 it is terminally optimal and after a finite time it stops at Step 6. Then by lemmas 6 and 7 all tasks are in mode 'optimal' with $x_n = \xi_n$, $n \in \mathcal{N}$ solving the corresponding problems (2.4). The proof is complete.

Let us now briefly discuss the case of $\rho_n = 0$ for some n. Lemmas 6, 7 and 8 remain unchanged. Lemma 9 still holds for n with $\rho_n > 0$ and is inessential for $\rho_n = 0$. A substantial difficulty, however, arises in Lemma 10. With $\rho_n = 0$ we cannot guarantee that the minimum value of (3.3)-(3.5) increases after introducing a new cut and a theoretical possibility of cycling occurs, especially when inactive cuts are dropped. To avoid this we need an additional indirect assumption:

(AS) If $\rho_n = 0$ and $x_{\pi(n)}$ does not change for $\tau \geq \tau_0$ then a change of x_n for $\tau \geq \tau_0$ implies the increase of the minimum value of (3.3)-(3.5).

With *(AS)* Lemma 10 is still valid and Theorem 1 remains true. However, the use of $\rho_n > 0$ for selected n is not a theoretical trick to exclude cycling, but may have a stabilizing effect on (3.3)-(3.5) preventing from rapid changes of trial points x_n.

6 Illustrative Example

Subproblems of our method process many cuts and exchange information in both directions without waiting till optimality of their descendants. Most of the cuts are thus inexact. On the other hand, a true support of $f_n(\cdot)$ at a given point only occurs when all subproblems have been optimized (see lemmas 3 and 7). So, in a multistage case doubts may arise as to real advantages offered by parallelisation over the nested approach. For the two-stage stochastic case of Example 2 advantages are rather clear, because the lower level subproblems are independent, but in the other extreme case of the dynamic problem of Example 1 the answer is in no way obvious. We shall present a simple analytical example which shows that a gain in computation time is possible in this case too.

Let s_t denote the state of a linear dynamic system and u_t and w_t denote control variables, $t = 1, \ldots, T$. Both state and controls are scalars and the problem is to

$$minimize \sum_{t=1}^{T} (c_t u_t - q_t w_t) \tag{6.1}$$

subject to the state equation

$$s_t = s_{t-1} + u_t - w_t, \; t = 1, \ldots, T, \tag{6.2}$$

bounds on state variables and controls

$$s_t \geq 0, \; t = 1, \ldots, T, \tag{6.3}$$

$$0 \leq u_t \leq \bar{u}_t, \; t = 1, \ldots, T, \tag{6.4}$$

$$w_t \geq 0, \; t = 1, \ldots, T, \tag{6.5}$$

with initial state $s(0)$ and terminal condition

$$s_T \geq s_T^{min}. \tag{6.6}$$

Numerical values of coefficients are the following

$$T = 4,$$

$$c = (2, 4, 3, 5),$$

$$q = (\tfrac{1}{2}, \tfrac{1}{2}, 1, 1),$$

$$\bar{u} = (4, 4, 4, 10),$$

$$s_4^{min} = 5.$$

It is obvious that the above problem can be put into the general format by defining $x_t = (s_t, u_t, w_t)$. The optimal trajectory is clearly $\hat{s} = (5, 5, 5, 5)$, but we shall see that it is rather difficult to find by the nested decomposition method.

We shal compare on our example three methods:

- The nested decomposition method, in which cuts for the predecessor are generated only when all successors are optimized.
- The optimal serial algorithm, where the order of processing the subproblems is best possible for a given problem.

- The parallel method in its simplest synchronous version.

In Figure 3 we illustrate the operation of the nested decomposition method. For each cycle of the method we fill-in the box in the column corresponding to the task $SUB(t)$ which is solved at a given cycle and we give the current state value and the cut (if any). Empty boxes denote problems that do not generate any new information.

Figure 4 presents in the same way the optimal serial algorithm. It is clearly more efficient than the nested method, because the order of subproblems has been determined *a posteriori*. In our case it turned out to be a compressed version of the nested method. All heuristic strategies for processing subproblems within the nested decomposition method cannot be better than that of Figure 4 (e.g. the *fast-forward-fast-back algorithm* discussed in [3] is quite good here and requires only two cycles more).

In Figure 5 we show the operation of the parallel method assuming that $\rho_t = 0$, $t = 1, 2, 3$ and that the method operates synchronously: odd subproblems are processed at odd cycles and even subproblems at even cycles. We also assume that at the beginning the tasks have artificial cuts $f_{t+1}(x_t) \geq -C$ and that they do not generate cuts for their predecessors if the artificial cuts are active. For every cycle we display only the tasks that receive or generate new information. In each box we give the information generated at a given cycle (empty boxes denote tasks that do not generate anything new) and we use arrows to denote the flow of the information.

We see that for our problem the parallel method is faster than the optimal serial algorithm, so a significant novelty is introduced by parallelization. This differs our approach from the parallel dynamic programming algorithms of [2] which in fact find the best sequence of subproblems by considering in parallel all sequences. The reason is that in our method information flows in both directions: forwards in form of trial points and backwards in form of cuts. Therefore it is possible that a good trial point and a significant cut are generated in parallel and both contribute to the final solution. In our example such a situation occured in cycles 6-9 and resulted in a relevant cut $f_3 \geq 5 - s_2$ generated at cycle 9 (the flow of significant information is indicated in Figure 5 by thick arrows). In the serial algorithm the same cut could be obtained only at the 13th cycle. In our example the advantages are not high, but the analysis of this phenomenon indicates that the potential of palallelization may grow with the increase of the number of time stages and with the increase of the dimensionality of subproblems.

Clearly, asynchronism and the use of regularization in some subproblems might complicate the picture considerably, but our aim in this section was not the comparison of methods but illustration of the mechanism that can make the parallel method attractive.

7 Simulation results

We also carried out a series of simulation tests for both synchronous and asynchronous versions of our decomposition method. We used the multitasking facility for PC computers of [29].

The first series of experiments were carried out for deterministic inventory control problems of form (6.1)-(6.6), with $T = 5$, $T = 10$, $T = 15$ and $T = 20$, and with other coefficients generated from the uniform distribution in [0,10]. If the problem was infeasible (which is easy to test) the generation was repeated. For each T we generated in this way 40 problems an we compared for them three times:

τ_0 - the processing time for the serial algorithm in a single-processor computer,

τ_s - the processing time for the synchronous method in a multi-processor computer,

τ_a - the processing time for the asynchronous method in a multi-processor computer. The times τ_s and τ_a were computed in the following way: all tasks were processed cyclically with a very small time quantum (0.056 s) devoted to each of them at every call; one full cycle of this procedure was assumed to take only the time quantum involved. To make the quanta negligible with respect to the time needed to solve a subproblem we introduced to the subproblems some delays, so the times that we present below have only relative meaning. To eliminate the effect of indeterminism in the asynchronous method, for each problem the method was run three times, so we made in total 200 runs for every T.

In Table 1 we present for each T results for three characteristic problems: the best one (from the point of view of acceleration introduced by parallelization), a typical one and the worst one; we give also average results of all 40 randomly generated problems. It follows from Table 1 that parallelization decreases the computation time and that the improvement is bigger for longer time horizons T. This is particularly interesting for the special class of inventory-type problems that we tested and shows that for larger T the phenomenon discussed in our analytical example of the previos section becomes a rule. Asynchronism does not significantly change the results with respect to the synchronous case.

More insight into the behavior of the method can be drawn from the optimal state trajectories and from relative activities of tasks $A(t) = 100\tau_a(t)/\tau_a$, where $\tau_a(t)$ is the time in which task $SUB(t)$ was active. In Table 2 we provide these data for the three cases previously selected for $T = 20$.

From the optimal state trajectories we see that the advantages of full parallelization can be higher for problems where the dynamic structure is substantial (our best case) than for easier ones which are in fact composed of a number of short-horizon problems (as our worst case). Similar relations were observed in all experiments.

Another interesting conclusion follows from the analysis of the activity of tasks. Most of the time the tasks are suspended, waiting for new information, especially in our worst case. There is also a significant correlation of the activity of the tasks and the optimal state trajectory: the activity is higher at the end of each period of accumulation. This suggested us to analyse how many tasks are simultaneously active in the course of calculation.

In Table 3 we show for the 12 cases selected earlier the relative time of simultaneous activity for each number of tasks. Acceleration is strongly correlated with essential parallelization, i.e. with high percentage of time when many tasks are active. However, the load of the tasks is not equal, so a question arises how many processors we really need to process these tasks efficiently in a time close to the ideal case studied here.

Such an analysis has been carried out for our 12 problems with the synchronous method: for each cycle of the method we processed the tasks of this cycle by a given number of processors and the next cycle was initiated only after completing this cycle's work. Two task scheduling schemes were used: the *static* and the *dynamic* one.

In the static method a subset of subproblems was preassigned to each processor. The processors solved subproblems in their own queues before reaching the synchronization barrier. We have chosen sub-chains of neighboring tasks as the groups allocated to processors to minimize data transfer between them.

In the dynamic task scheduling method all tasks of a given cycle were ordered in a job queue and subsequently assigned to currently idle processors. Each processor after solving a subproblem returned for more work until the queue was empty, which initiated the next cycle.

In Tables 4 and 5 we present the speedup factors over the serial method thus obtained. These factors are defined as τ_0/τ_s and τ_0/τ_a, respectively, with τ_s and τ_a obtained with

the given number of processors.

A number of interesting conclusions can be drawn from these results. First, it is obvious that dynamic task scheduling is much better than the static one. Secondly, for the dynamic assignment a significant acceleration can be achieved with a modest number of processors, especially for larger T. In fact, usually 3 processors provide reasonable results. Both these phenomena follow from the fact that at each cycle only a small but variable subset of tasks really need be solved again, while the other ones remain optimal.

Summing up these experiments we can say that in the presence of dynamic structure our parallel scheme already has a potential of accelerating the processing over the serial method. The speedup is not high but can be achieved with a small number of processors. This fully agrees with the results recently obtained in [8].

The second series of experiments were carried out with stochastic versions of the inventory control problem (6.1)-(6.6):

$$\textit{minimize} \quad \sum_{\omega\in\Omega} p_\omega \sum_{t=1}^{T} (c_\omega(t)u_\omega(t) - q_\omega(t)w_\omega(t)) \tag{7.1}$$

subject to the state equation

$$s_\omega(t) = s_\omega(t-1) + u_\omega(t) - w_\omega(t),\ t = 1,\dots,T, \tag{7.2}$$

bounds on state variables and controls

$$s_\omega(t) \geq 0,\ t = 1,\dots,T, \tag{7.3}$$

$$0 \leq u_\omega(t) \leq \bar{u}_\omega(t),\ t = 1,\dots,T, \tag{7.4}$$

$$w_\omega(t) \geq 0,\ t = 1,\dots,T, \tag{7.5}$$

with initial state $s(0)$ and terminal condition

$$s_\omega(T) \geq s_\omega^{\min}(T), \tag{7.6}$$

The difference in comparison with (6.1)-(6.6) is that

$$((c_\omega(t), q_\omega(t), \bar{u}_\omega(t)),\ t = 1,\dots,T,\ s_\omega^{\min}(T)),\ \omega \in \Omega,$$

is now a finite collection of scenarios and our decisions

$$(s_\omega(t), u_\omega(t), w_\omega(t)),\ t = 1,\dots,T,$$

must depend on scenarios in a nonanticipative way. We assume that the tree of scenarios (introduced in section 2) has one node at level $\mathcal{N}_1$ and that nodes at levels $\mathcal{N}_t$, $t = 1,\dots,T-1$ have L sons each. The total number of nodes (tasks in the decomposition method) is then equal to $(L^T - 1)/(L - 1)$, which can be quite large even for moderate T and L.

The following pairs of T and L were tested: (4,2), (4,3), (4,4), (5,3), (6,2) and (8,2), with the number of nodes (tasks) ranging from 15 to 255. The coefficients for each scenario were generated from the uniform distribution in [0,10] and the probabilities of scenarios were generated from [0,1] and normalized afterwards. For each pair (T, L) we generated in this way 10 problems and we tested on them the serial decomposition method and two versions of the parallel decomposition: the synchronous and the asynchronous one.

As before, the asynchronous method was run three times for each problem to reduce the effects of indeterminism (which in fact turned out to be inessential).

In Table 6 we present for each (T, L) the speedup factors of the parallel decomposition methods over the serial algorithm for three characteristic problems: the best one (in this series) a typical one and the worst one; we give also average results of all 10 randomly generated problems (only relative times are given because our tricks with multitasking, delays and communication made real times completely meaningless). The speedup factors, as before, are defined as τ_0/τ_s and τ_0/τ_a, respectively.

It follows from Table 6 that the gains from parallelization are much higher than in the deterministic case and that they are closely related to the number of nodes of the tree (which is quite large in some cases). The asynchronous method is 10-30% better than the synchronous one. Presumably, for broader trees the possibility of faster transmission of information along some branches becomes relevant. This effect would probably increase for more realistic problems with differentiated solution times of subproblems. It should be stressed, however, that our results have been obtained under an assumption that the number of processors is equal to the number of nodes of the scenario tree.

Therefore, similarly to the deterministic case, we analysed the possibility of processing the tasks by a smaller number of processors without losing the advantages of parallelization. To this end we used the synchronous method with dynamic task scheduling with an increasing number of processors. The results for the 21 problems selected earlier are given in Table 7. For the largest problems they are additionally illustrated in Figure 6 (the vertical range is equal to the maximum speedup from Table 6).

If the number of processors P is small in comparison with the number of nodes of the tree the speedup is almost linear in P and processor efficiency is high (ca 75%). From this point of view the fact that our approach decomposes the problem into a very large number of small tasks becomes an advantage. Clearly, our strategy for assigning tasks to processors was far from being optimal: we just processed alternately the odd and the even levels with task queuing used only within each cycle. Strategies that take advantage of the potential of asynchronism could do even better. On the other hand, the subproblems at a given level usually exhibit a lot of similarities (see, e.g., [11]), so advantage can be taken of synergies. This creates the need for a reasearch on scheduling strategies in our decomposition scheme.

Summing up, our simulation results indicate that for stochastic problems our parallel method can significantly accelerate computation over the serial approach. It is also rather flexible allowing for the use of any number of processors.

8 Conclusions

Our decomposition approach differs from earlier methods in two ways.

- All subproblems can be solved in parallel and exchange information in an asynchronous manner. This may speed up the flow of information between stages.
- The method admits the use of regularizing quadratic terms in subproblems to stabilize their solution and allow deleting inactive cuts.

In spite of these modifications, the method shares the finite termination property of classical approaches. Our simple illustrative example proves that parallelization introduces a substantial novelty into decomposition methods and that a parallel version can be faster than any serial algorithm.

The numerical experience gained so far indicates that

- advantages of parallelization increase with the growth of the scenario tree,
- for deterministic problems a small number of processors is sufficient to exploit the advantages offered by parallelization,
- for stochastic problems the speedup is close to the number of processors, if it is small compared with the number of nodes of the decision tree,
- asynchronism helps when the scenario tree is broad.

These conclusions suggest that our parallel decomposition scheme can become a powerful tool for solving structured problems with replications due to dynamics and stochastics.

Acknowledgement. The author expresses his gratitude to Mr. Radosław Zakrzewski and Mr. Jarosław Romańczuk for their assistance in developing the simulation program and in testing the methods.

References

[1] D.P. Bertsekas, "Distributed dynamic programming", *IEEE Transactions on Automatic Control* AC-27(1982) 610-616.

[2] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation* (Prentice-Hall, Englewood Cliffs, 1989).

[3] J.R. Birge, "Decomposition and partitioning methods for multistage stochastic linear programs", *Operations Research* 33(1985) 989-1007.

[4] J.R. Birge and F.V. Louveaux, "A multicut algorithm for two-stage stochastic linear programs", *European Journal of Operations Research* 34(1988) 384-392.

[5] J. Bisschop and A. Meeraus, "Matrix augmentation and partitioning in in the updating of the basis inverse", *Mathematical Programming* 30(1984) 71-87.

[6] G.B. 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 I* (University of California Press, Berkeley, 1961) pp. 165-176.

[7] G.B. Dantzig and P. Wolfe, "Decomposition principle for linear programs", *Operations Research* 8(1960) 101–111.

[8] R. Entriken, "The parallel decomposition of linear programs", technical report SOL 89-17, Department of Operations Research, Stanford University, November 1989.

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

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

[11] H.I. Gassmann, "MSLiP: A computer code for multistage stochastic linear programming problem", working paper 64, School of Business Administration, Dalhousie University, Halifax 1987.

[12] J. Gondzio and A. Ruszczyński, "A sensitivity method for solving multistage stochastic linear programming problems", in: A Lewandowski and A. Wierzbicki, eds., *Aspiration Based Decision Support Systems, Lecture Notes in Economics and Mathematical Systems 331*, (Springer-Verlag, Berlin, 1989) pp. 68-79.

[13] J.K. Ho, T.C. Lee and R.P. Sundarraj, "Decomposition of linear programs using parallel computation", technical report, College of Business Administration, University of Tennessee (Knoxville, 1987).

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

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

[16] K.C. Kiwiel, "A dual method for certain positive semidefinite quadratic programming problems", technical report, Systems Research Institute, Warsaw 1987.

[17] I.J. Lustig, J.M. Mulvey and T.J. Carpenter, "Formulating stochastic programs for interior point methods", technical report SOR 89-16, Department of Civil Engineering and Operations Research, Princeton University, Princeton 1989.

[18] J.M. Mulvey and H. Vladimirou, "Evaluation of a parallel hedging algorithm for stochastic network programming", technical report SOR 88-14, Department of Civil Engineering and Operations Research, Princeton University, Princeton 1988.

[19] B. Murtagh, *Advanced Linear Programming* (McGraw-Hill, 1981).

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

[21] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton 1970).

[22] R.T. Rockafellar and R.J.-B. Wets, "A Lagrangian finite generation technique for solving linear quadratic problems in stochastic programming", *Mathematical Programming Study* 28(1986) 63-93.

[23] R.T. Rockafellar and R.J.-B. Wets, "Scenarios and policy aggregation in optimization under uncertainty", WP-87-119, IIASA (Laxenburg, 1987).

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

[25] A. Ruszczyński, "Regularized decomposition of stochastic programs: algorithmic techniques and numerical results", technical report, Institute of Automatic Control, Warsaw University of Technology, 1986.

[26] A. Ruszczyński, "Modern techniques for linear dynamic and stochastic programs", in: A. Lewandowski and A. Wierzbicki, eds., *Aspiration Based Decision Support Systems, Lecture Notes in Economics and Mathematical Systems 331*, (Springer-Verlag, Berlin, 1989) pp. 48-67.

[27] A. Ruszczyński, "Regularized decomposition and augmented Lagrangian decomposition for angular linear programming problems", in: A. Lewandowski and A. Wierzbicki, eds., *Aspiration Based Decision Support Systems, Lecture Notes in Economics and Mathematical Systems 331*, (Springer-Verlag, Berlin, 1989) pp. 80-91.

[28] A. Ruszczyński, "An augmented Lagrangian decomposition method for block diagonal linear programming problems", *Operations Research Letters* 8(1989) 287-294.

[29] J. Sobczyk, "Multitasking in TURBO PASCAL", technical report, Institute of Automatic Control, Warsaw University of Technology, 1989.

[30] 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.

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

[32] R.J.-B. Wets, "Large scale linear programming techniques", in: Yu. Ermoliev and R.J.-B. Wets, eds, *Numerical Methods in Stochastic Programming*, (Springer-Verlag, Berlin, 1988) pp. 65-94.

[33] R. Wittrock, "Dual nested decomposition of staircase linear programs", *Mathematical Programming Study* 24(1985) 65-86.

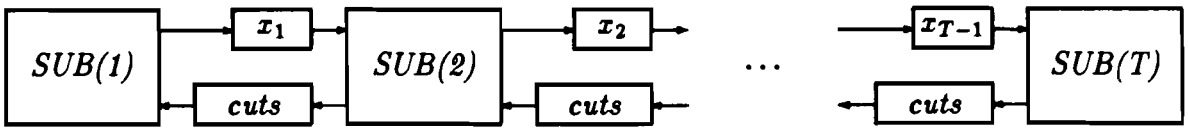


Figure 1: The network of tasks for the deterministic dynamic problem.

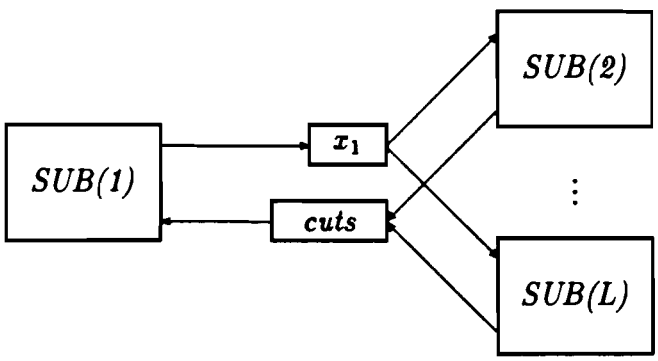


Figure 2: The network of tasks for the stochastic two-stage problem.

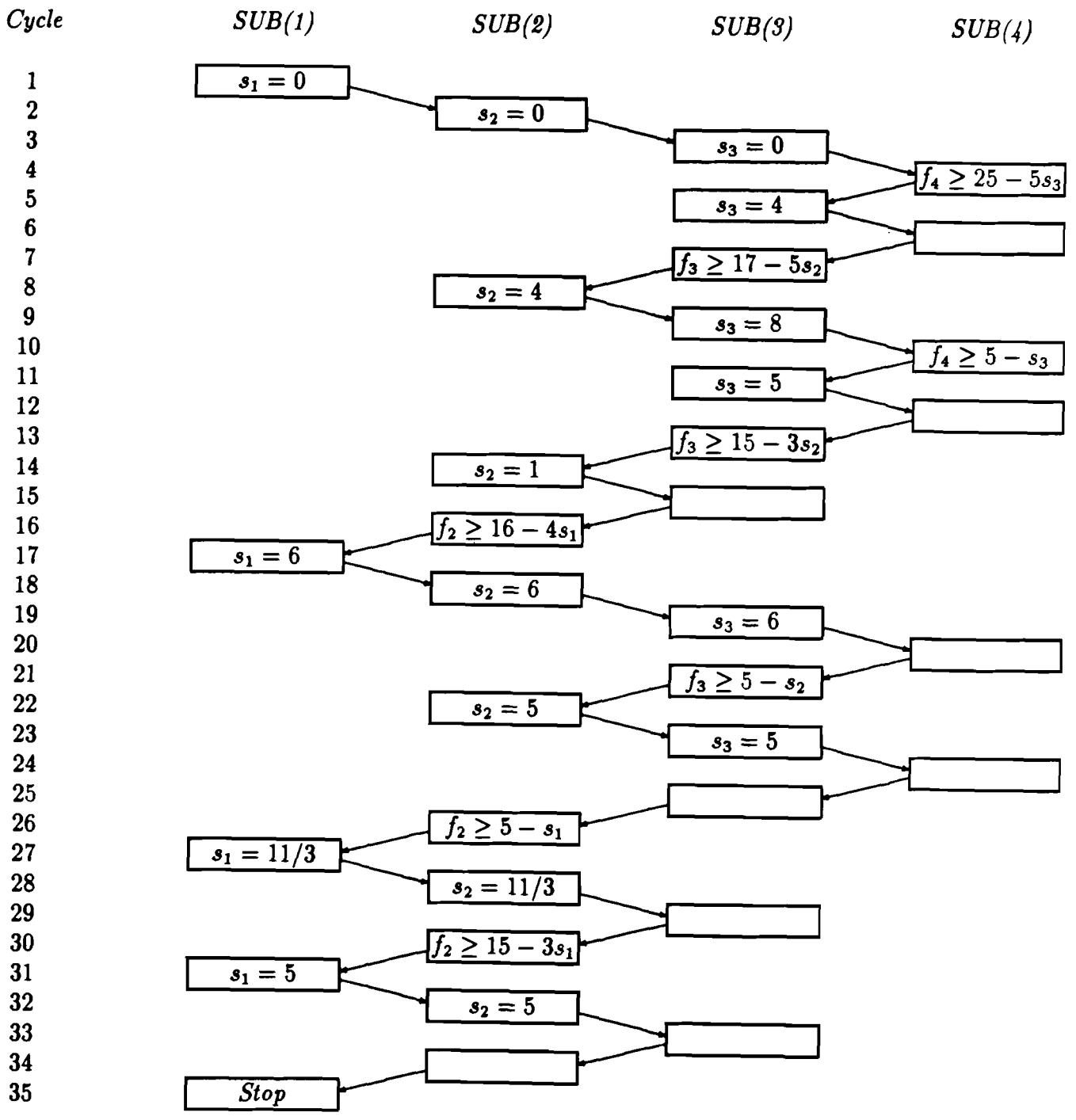


Figure 3: The nested decomposition method.

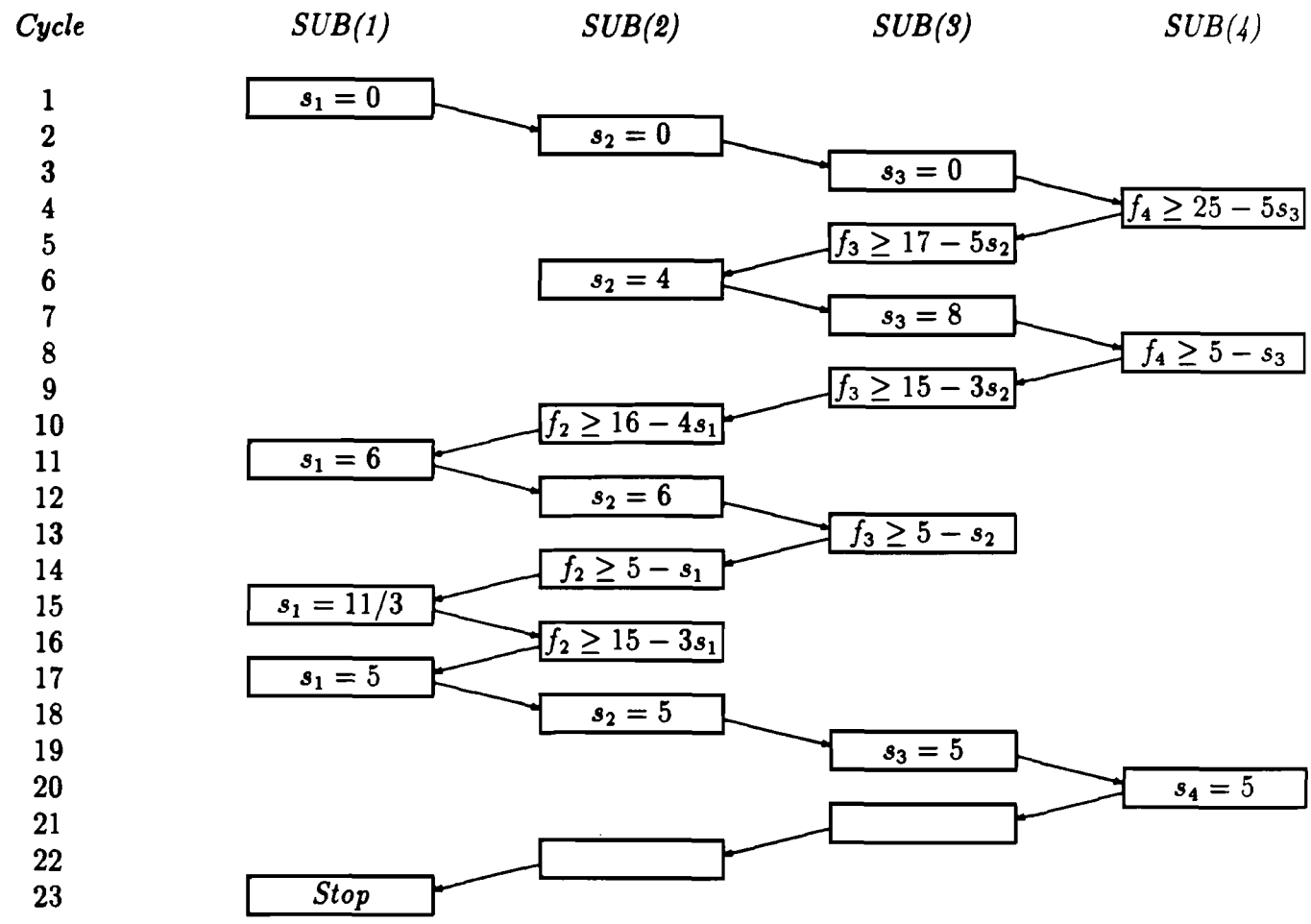


Figure 4: The optimal serial algorithm.

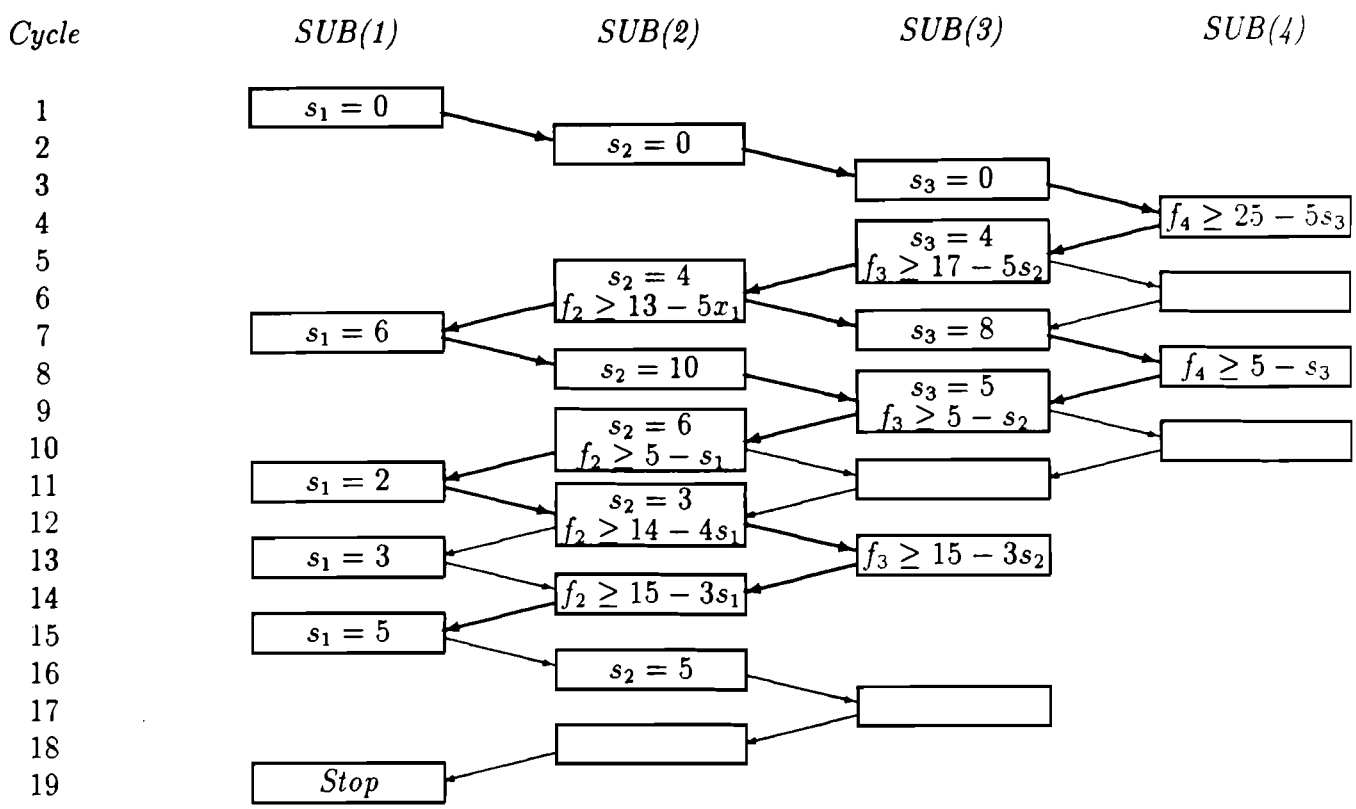


Figure 5: The parallel method.

N	Problem	Serial	Synch.	Asynch.
5	best	6.2	4.3	3.6
5	typical	8.5	6.5	7.0
5	worst	6.0	6.8	6.9
5	average	5.4	4.5	4.4
10	best	19.5	10.0	8.9
10	typical	16.6	12.7	10.3
10	worst	9.5	8.0	8.0
10	average	15.1	9.6	9.3
15	best	43.5	14.4	14.2
15	typical	29.2	13.2	13.0
15	worst	16.0	12.6	12.6
15	average	30.6	14.1	13.7
20	best	87.3	23.3	23.2
20	typical	36.6	16.6	15.7
20	worst	29.1	19.0	19.3
20	average	44.2	19.2	18.9

Table 1: Execution times of decomposition methods for deterministic problems.

	best		typical		worst	
t	$s(t)$	$A(t)$	$s(t)$	$A(t)$	$s(t)$	$A(t)$
1	4.48	3.7	7.70	9.9	2.97	4.3
2	6.08	5.4	13.91	12.1	11.16	6.5
3	14.17	7.2	0.00	12.0	14.45	8.4
4	16.60	9.0	0.64	9.9	15.19	10.5
5	21.46	10.7	9.42	12.2	20.67	12.6
6	25.73	12.5	17.35	14.7	24.92	14.6
7	35.36	14.3	25.60	16.9	32.48	16.8
8	44.19	16.1	31.20	19.1	0.00	18.3
9	52.26	17.7	38.21	19.1	0.00	4.3
10	58.07	19.4	42.49	21.5	8.21	4.4
11	61.70	21.1	44.69	26.1	13.77	6.4
12	71.35	23.0	48.69	28.2	0.00	8.3
13	77.38	24.7	51.16	30.6	0.00	4.3
14	87.11	26.4	56.50	33.2	0.00	4.3
15	87.36	28.1	0.00	34.4	4.94	4.4
16	93.44	29.9	0.00	9.8	4.94	6.4
17	93.44	31.5	0.00	9.8	0.00	8.5
18	98.89	31.7	0.00	9.9	0.00	6.5
19	104.49	32.6	0.39	14.1	2.80	6.5
20	2.79	1.7	0.39	8.9	7.43	5.0

Table 2: Optimal state trajectories and activity of tasks ($N = 20$).

N	Problem	Number of active tasks									
		1	2	3	4	5	6	7	8	9	10
5	best	60.1	31.8	8.1	-	-	-	-	-	-	-
5	typical	60.8	33.9	5.3	-	-	-	-	-	-	-
5	worst	30.4	49.7	19.9	-	-	-	-	-	-	-
10	best	58.6	13.7	15.2	12.5	-	-	-	-	-	-
10	typical	48.0	14.3	16.0	17.2	4.5	-	-	-	-	-
10	worst	78.8	21.2	-	-	-	-	-	-	-	-
15	best	40.2	9.6	9.5	9.6	9.5	9.5	9.5	2.6	-	-
15	typical	60.1	7.9	7.9	10.8	10.7	2.6	-	-	-	-
15	worst	80.5	13.9	5.6	-	-	-	-	-	-	-
20	best	49.3	5.9	4.4	6.0	5.9	6.0	6.5	7.1	7.0	1.9
20	typical	44.6	8.1	15.8	9.3	4.3	10.2	5.1	2.6	-	-
20	worst	66.3	18.4	8.7	6.6	-	-	-	-	-	-

Table 3: Times of simultaneous activity of tasks [%].

N	Problem	Number of processors							
		1	2	3	4	5	6	7	8
5	best	0.98	1.32	1.43	-	-	-	-	-
5	typical	0.92	1.18	1.31	-	-	-	-	-
5	worst	0.51	0.72	0.87	-	-	-	-	-
10	best	0.98	1.26	1.37	1.45	1.72	-	-	-
10	typical	0.69	0.90	1.07	1.16	1.33	-	-	-
10	worst	0.99	1.07	1.11	1.27	1.28	-	-	-
15	best	0.96	1.30	1.51	1.97	1.97	2.39	2.92	3.00
15	typical	0.97	1.16	1.34	1.60	1.60	2.08	2.20	2.20
15	worst	0.98	1.03	1.11	1.17	1.17	1.17	1.28	1.32
20	best	0.96	1.30	1.65	1.91	2.31	2.39	2.36	2.71
20	typical	0.85	1.11	1.17	1.28	1.51	1.61	1.60	1.79
20	worst	0.97	1.03	1.10	1.16	1.20	1.24	1.40	1.40

Table 4: Speedup factors for deterministic problems with static task scheduling.

N	Problem	Number of processors							
		1	2	3	4	5	6	7	8
5	best	0.98	1.33	1.42	-	-	-	-	-
5	typical	0.92	1.25	1.30	-	-	-	-	-
5	worst	0.51	0.80	0.87	-	-	-	-	-
10	best	0.98	1.37	1.55	1.72	-	-	-	-
10	typical	0.69	1.00	1.15	1.29	1.33	-	-	-
10	worst	0.99	1.28	-	-	-	-	-	-
15	best	0.96	1.58	2.00	2.32	2.50	2.70	2.94	3.00
15	typical	0.97	1.47	1.75	1.93	2.14	2.20	-	-
15	worst	0.98	1.24	1.32	-	-	-	-	-
20	best	0.96	1.62	2.10	2.43	2.72	2.88	3.05	3.24
20	typical	0.85	1.34	1.69	1.90	1.98	2.12	2.21	2.25
20	worst	0.97	1.35	1.48	1.57	-	-	-	-

Table 5: Speedup factors for deterministic problems with dynamic task scheduling.

N	L	Tasks	Problem	Synch.	Asynch
4	2	15	best	3.48	3.67
4	2	15	typical	2.84	3.74
4	2	15	worst	2.58	3.41
4	2	15	average	2.89	3.53
4	3	40	best	5.40	7.62
4	3	40	typical	4.90	6.64
4	3	40	worst	4.04	5.24
4	3	40	average	4.84	6.67
4	4	85	best	9.41	11.49
4	4	85	typical	8.18	10.93
4	4	85	worst	6.45	6.44
4	4	85	average	8.00	8.68
4	5	156	best	13.89	19.24
4	5	156	typical	11.42	11.86
4	5	156	worst	7.76	7.34
4	5	156	average	11.41	13.22
5	3	121	best	12.70	15.14
5	3	121	typical	10.53	16.38
5	3	121	worst	9.97	12.48
5	3	121	average	10.85	13.97
6	2	63	best	8.16	8.91
6	2	63	typical	7.09	8.17
6	2	63	worst	5.58	6.77
6	2	63	average	7.11	8.55
8	2	255	best	23.80	24.54
8	2	255	typical	20.01	23.74
8	2	255	worst	17.50	22.87
8	2	255	average	20.24	23.20

Table 6: Maximum speedup factors for stochastic problems.

T	L	Problem	Number of processors							
			1	2	3	4	5	6	7	8
4	2	best	1.05	1.80	2.38	2.59	3.28	3.33	3.33	3.33
4	2	typical	0.98	1.74	2.28	2.64	2.84	2.84	2.84	2.84
4	2	worst	0.87	1.57	1.99	2.33	2.48	2.50	2.50	2.50
4	3	best	0.97	1.84	2.66	3.26	3.64	4.18	4.73	4.91
4	3	typical	0.86	1.63	2.33	2.86	3.27	3.65	3.97	4.26
4	3	worst	0.91	1.74	2.37	2.84	3.19	3.58	3.90	4.01
4	4	best	1.02	2.00	2.84	3.66	4.52	5.08	5.66	6.04
4	4	typical	0.95	1.82	2.57	3.33	3.87	4.60	4.89	5.45
4	4	worst	0.93	1.78	2.52	3.24	3.76	4.37	4.89	5.17
4	5	best	1.00	1.98	2.90	3.85	4.75	5.65	6.41	7.06
4	5	typical	0.93	1.82	2.66	3.46	4.32	5.01	5.73	6.30
4	5	worst	0.82	1.61	2.34	3.09	3.73	4.37	5.00	5.46
5	3	best	1.00	1.97	2.89	3.80	4.58	5.34	6.14	6.79
5	3	typical	0.95	1.88	2.76	3.60	4.28	5.09	5.72	6.27
5	3	worst	0.88	1.73	2.51	3.30	3.99	4.56	5.15	5.67
6	2	best	0.87	1.70	2.42	3.10	3.79	4.42	4.86	5.18
6	2	typical	0.82	1.59	2.29	2.94	3.48	3.92	4.42	4.74
6	2	worst	0.78	1.49	2.11	2.66	3.10	3.49	3.83	4.20
8	2	best	0.95	1.89	2.80	3.70	4.55	5.37	6.21	6.96
8	2	typical	0.85	1.70	2.52	3.34	4.10	4.84	5.55	6.24
8	2	worst	0.80	1.60	2.37	3.12	3.82	4.52	5.14	5.76

Table 7: Speedup factors for stochastic problems with dynamic task scheduling.

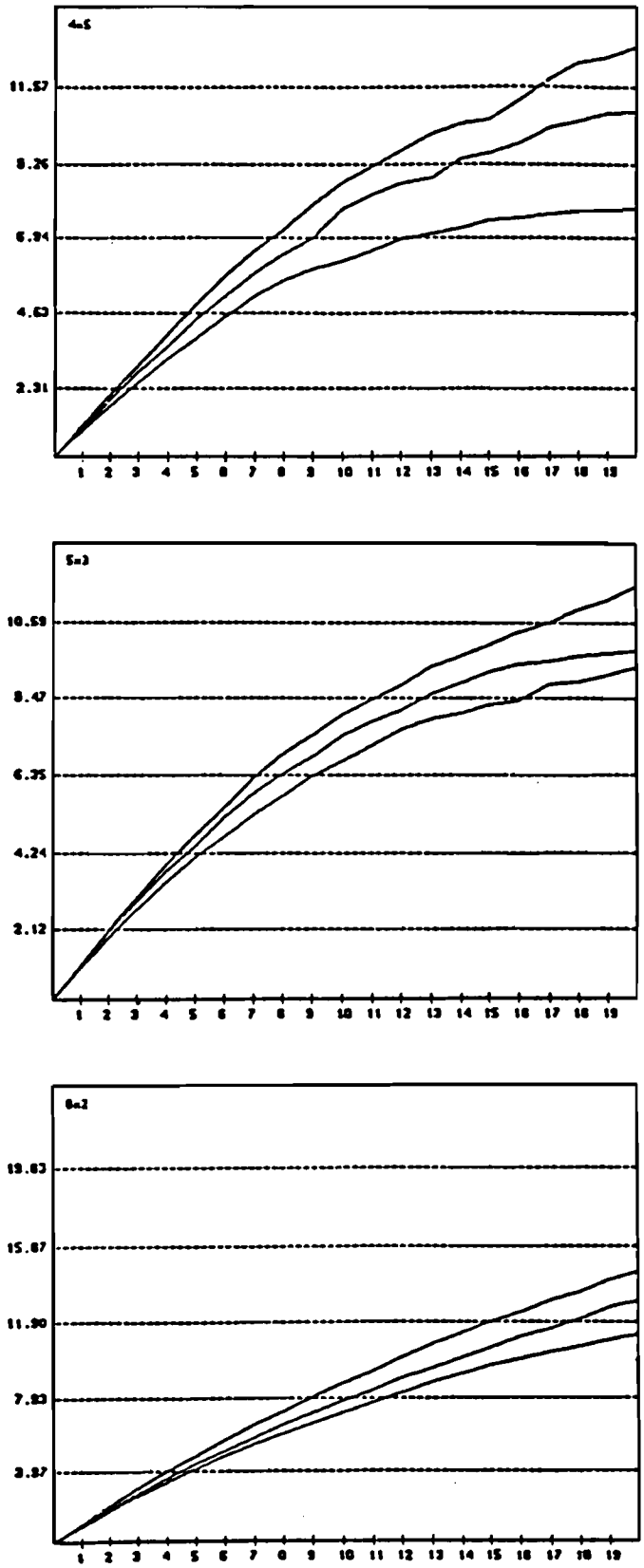


Figure 6: Speedup versus processors for (T, L)=(4,5) (top), (T, L)=(5,3) (center) and (T, L)=(8,2) (bottom).

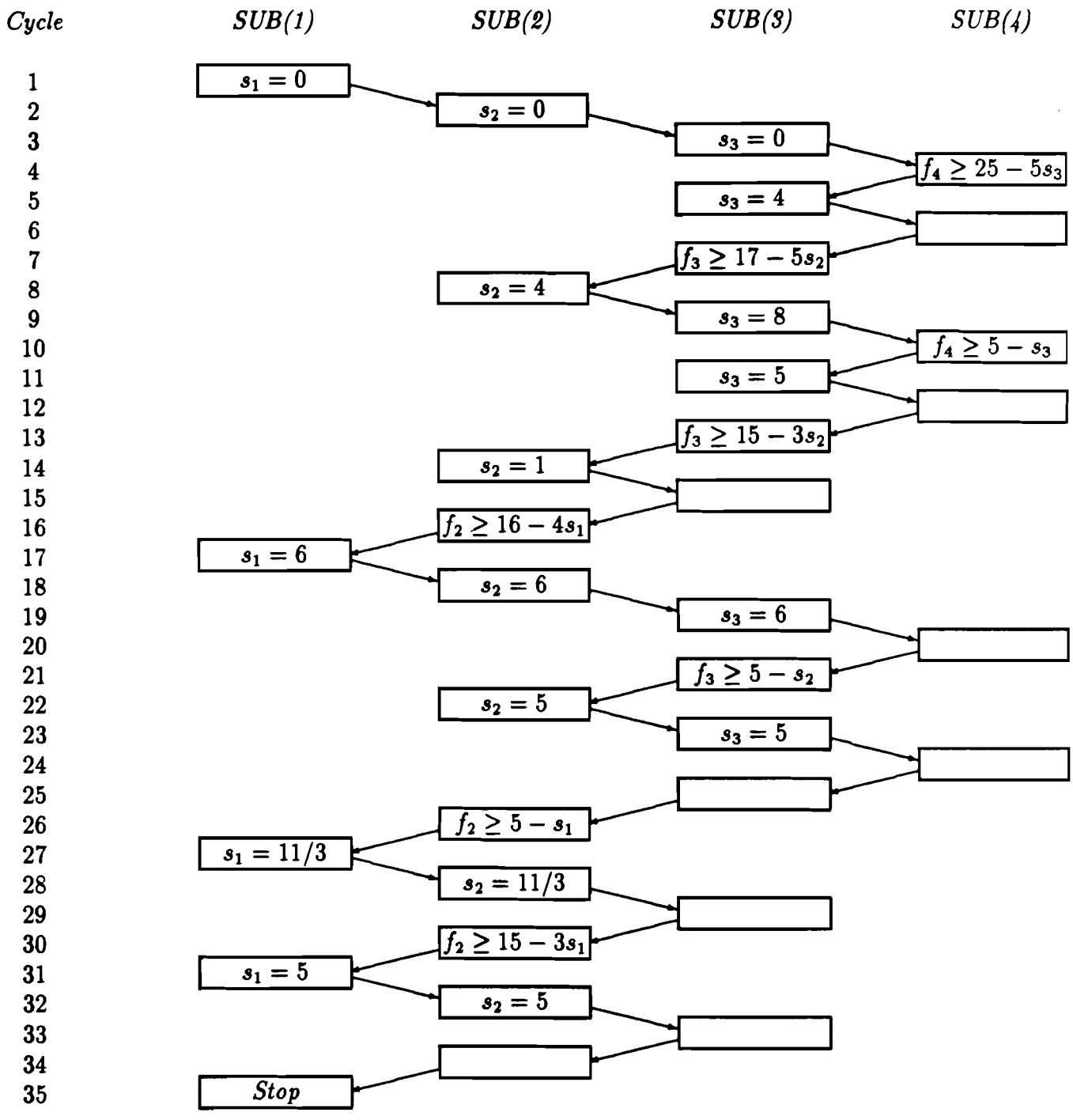


Figure 7: The nested decomposition method.

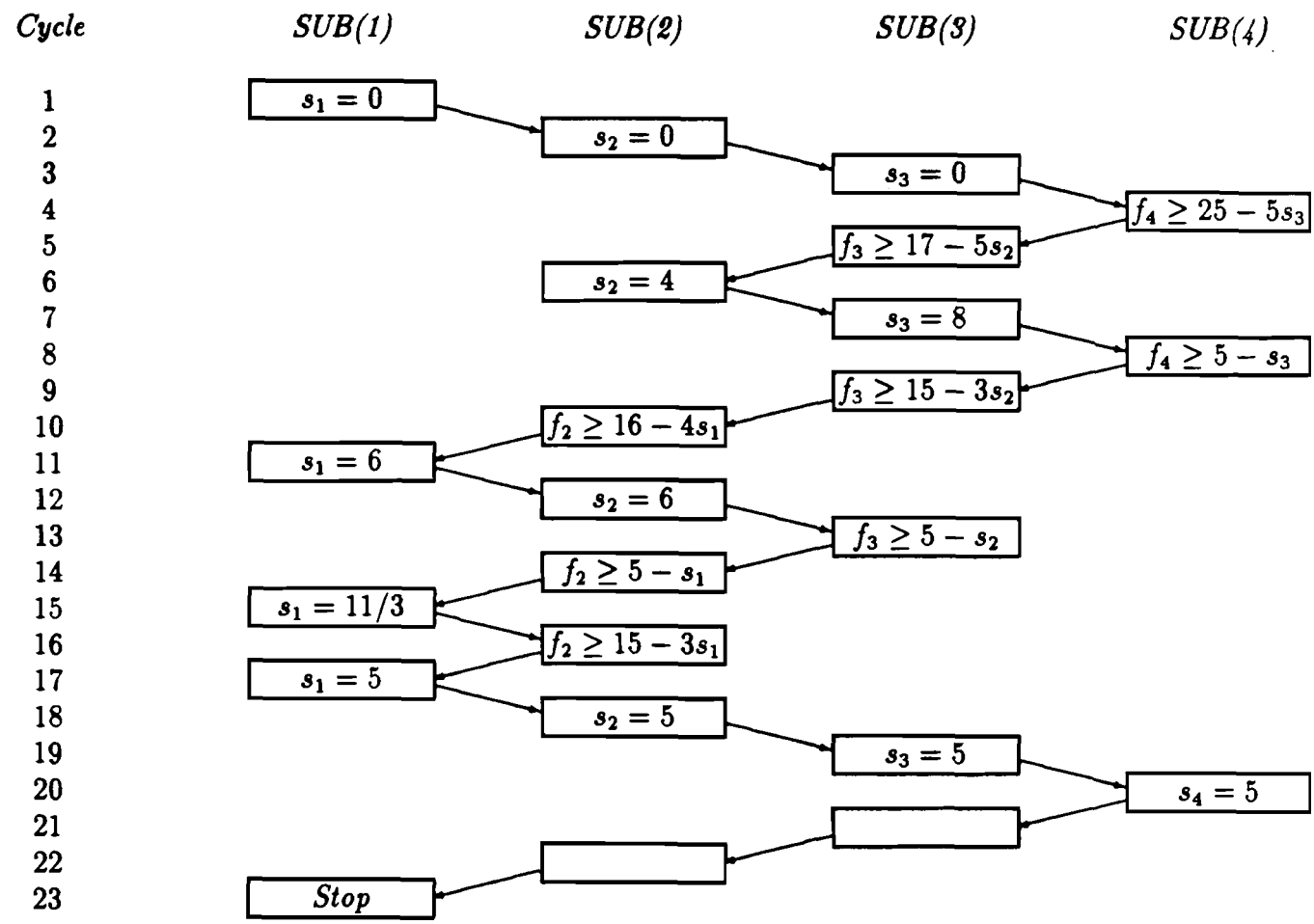


Figure 8: The optimal serial algorithm.

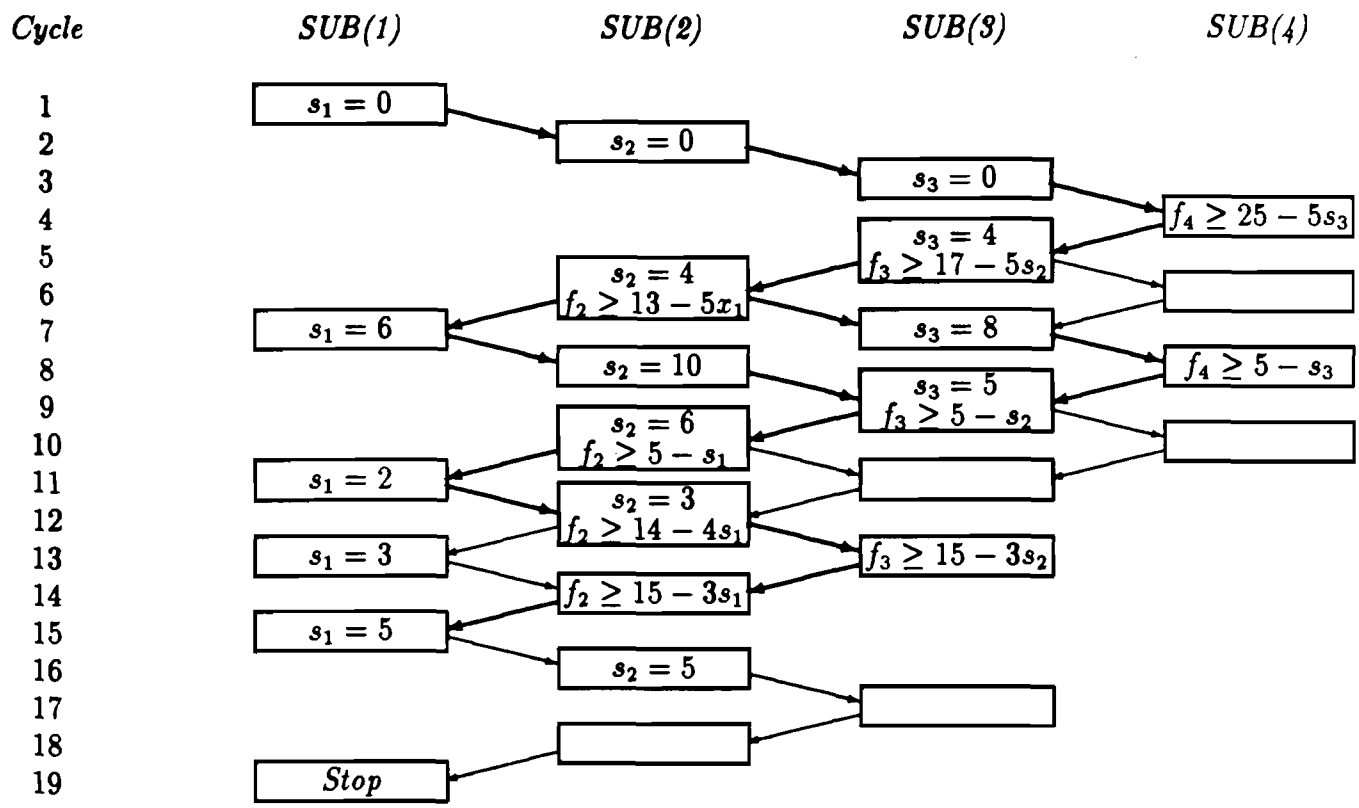


Figure 9: The parallel method.

Two–Dimensional Cutting Problem

J.Błażewicz , M.Drozdowski , B.Soniewicki , R.Walkowiak
Technical University of Poznań
Institute of Computing Science
Poznań, Poland

Abstract

This paper deals with two-dimensional cutting problems. Firstly the complexity of the problem in question is estimated. Then, several known approaches for the regular (rectangular) and irregular (not necessarily rectangular) cutting problems are described. In the second part, a decision support system for cutting a rectangular sheet of material into pieces of arbitrary shapes, is presented. The system uses two erlier described methods which prefer different types of data and the user may decide which one is more suitable for the problem in question. After brief description of system data files and its manual, some experimental results are presented.

Part I
Basic Complexity Results and Algorithms for Irregular Shapes

1 Introduction

In this work we present basic results for two-dimensional cutting problem. This problem consists in cutting a set of pieces from a sheet of material in order to minimize a waste. The problem arises in various production processes, such as the glass, steel, wooden, paper or textile industries. The problem is of combinatorial nature and, thus, can be analyzed along the lines appropriate for this class of problems. The basis of such an analysis is always computational complexity. Following it, one may design an appropriate algorithm for solving the problem in question. Unfortunately, majority of cutting problems are strongly NP-hard, thus, unlikely to admit even pseudopolynomial-time algorithms. Hence, they must be solved by approximation algorithms. One-dimensional and regular two-dimensional cutting problems allow for the application of approximation algorithms with a given accuracy (worst case behaviour). Unfortunately, no such method is known for irregular shapes, thus, heuristic approaches must be used. The above issues are presented in the following Chapters.

The organization of the paper is as follow. Chapter 2 contains problem formulation and a short introduction into the theory of computational complexity. Then, basic results for the one-dimensional version of the problem, are presented. In Chapter 3 two-dimensional regular (rectangular) problem is analyzed. A reference to several known algorithms is

made here. Chapter 4 deals with irregular (not necessarily rectangular) case and several methods solving this problem are presented. Then, some hints for the use the two algorithms described in a decision support system are given.

2 Basic Concepts, Definitions and Results for One-Dimensional Case

2.1 Problem Formulation

One-dimensional cutting problem is the easiest version of the problem. It can be stated in the following way: given rods of unit lengths cut them into the set of elements $a_i, 0 < a_i < 1, i = 1, \ldots, n$, in order to minimize the number of rods used. This problem has the same nature as memory allocation or nonpreemptive task scheduling problems for computer systems. References to this problem can be found in [7, 10, 18]. This is the same as bin-packing problem.

Two-dimensional regular problem can be formulated as follows: given a set of rectangles with dimensions y_i and x_i , $i = 1, \ldots, n$, distribute them into the minimal number of rectangular areas dimensioned Y and X. There are variants of this formulation. For example rectangular area to be filled with elements may have only one dimension limited while the other is to be minimized, rotation may be allowed or not, elements may appear once or more times. References to this problem may be found in [4, 5, 6, 8, 9, 11, 13, 16, 17, 21, 26] .

Two-dimensional irregular problem definition differs from the above formulation in the fact that any shapes of elements are admitted. The problem has been discussed in [1, 2, 3, 15].

2.2 Computational Complexity Issues

As was mentioned the complexity analysis is the basis for further studying problem. Thus, we will recall basic complexity definitions mainly with respect to decision problems, i.e. those requiring an answer of the "yes"-"no" type. Bin-packing (cutting) problem may be formulated in this way by asking a question if packing elements into the known number of bins is possible. On the other hand, plenty of optimization problems where some function is to be minimized (maximized), are known. Bin-packing in the original formulation is the optimization problem. There exists a close relation between decision and the optimization problems. If the optimization problem is easy to solve, then corresponding decision version is easy too. If decision version is difficult, then optimization problem is also difficult. We are going to use this relation further on. We consider only time complexity since space limitations are not of the great importance and may be avoided. Now we present basic definitions.

Decision problem Π is a set of parameters (sets, graphs, numbers) with values not necessarily assigned and a question with an "yes" or "no" answer. Assigning values to parameters creates **instance** I of problem Π. D_Π is a set of all instances. Data of I are encoded as a limited string $x(I)$ of symbols from known alphabet Σ according to some encoding rules. By an input size $N(I)$ we understand here the length of string $x(I)$. Only compact and precise encoding rules are allowed - redundant symbols are excluded, numbers are encoded with a base greater than 1. In practice $N(I)$ is assumed to be a number of the most important objects of the instance (tasks, polygons, nodes in a graph).

Computational complexity of algorithm A solving problem Π one defines as a function $f_A(n) = max\{t : t$ is a number of elementary computer steps needed to solve the problem for $I \in D_\Pi$ and $n = N(I)\}$.

Polynomial algorithm has computational complexity function (or complexity for short) $O(p(k))$ on deterministic Turing machine - DTM (or RAM model), where $p(k)$ is a polynomial, k is a size of the instance. Now we define classes of decision problems.

Class **P** consists of all problems solvable on DTM in polynomial time. (Hence, this class contains all problems solvable in polynomial time in practice).

Class **NP** consists of all problems solvable in polynomial time by nondeterministic Turing machine (NDTM). (In practice it is equivalent to the existence of a polynomial height branching tree in a branch and bound algorithm solving the problem) By the definition $P \subseteq NP$.

Polynomial transformation of problem Π_2 to Π_1 (we denote $\Pi_2 \propto \Pi_1$) is the function $f : D_{\Pi 2} \rightarrow D_{\Pi 1}$, satisfying:

1. for every $I_2 \in D_{\Pi 2}$ answer is "yes" if for $f(I_2)$ answer is "yes" too;
2. for every $I_2 \in D_{\Pi_2}$ time of computing f on DTM is bounded by polynomial in $N(I_2)$.

Decision problem Π_1 belongs to the class of **NP-complete** problems if $\Pi_1 \in$ **NP** and for every $\Pi_2 \in$ **NP**, $\Pi_2 \propto \Pi_1$. From the definition we conclude that if there is a polynomial algorithm for any NP-complete problem then any problem from **NP** may be solved by polynomial algorithm. This class contains such a problems as 3-dimensional matching, vertex cover, clique, Hamiltonian cycle, set partition, graph coloring. Despite many trials, no polynomial algorithm solving any NP-complete problem is known. Thus, we expect these problems to be solvable only by exponential algorithms (and then $P \neq NP$-complete class of problems).

On the other hand, certain NP-complete problems may be solved (quite efficiently, e.g. by dynamic programming) for the data appearing in the practice. Complexity of these algorithms is bounded by a polynomial of two variables - instance size $N(I)$ and maximum number value (appearing in the instance) $max(I)$. We call them **pseudo-polynomial algorithm**. Such an algorithm may only be constructed for a **number decision problem** which does not have $max(I)$ constrained by polynomial function of $N(I)$. We say that problem is **NP-complete in the strong sense** if it is in the class **NP** and there is polynomial p such that for D_Π limited to these instances only for which $max(I) \leq p(N(I))$, the problem remains NP-complete. From the above we see that no pseudo-polynomial algorithm is possible for the problem being NP-complete in the strong sense. To prove strong NP-completeness one applies strong **pseudo-polynomial transformation** (in which time bound for construction of function f is allowed to be pseudo-polynomial and some additional constraints on $N(I)$ and $max(I)$ are imposed) and some known strongly NP-complete problem.

Now, let us consider again optimization problems. If a decision version of the problem is NP-complete, then an exact optimization algorithm for the original (optimization) version must be exponential. In such a case one applies polynomial **approximation algorithms** to obtain approximate solution. It is desired to know how far from the optimum is the solution generated by such an approximation algorithm, i.e. how precise it is.

For the approximation algorithm A and instance I we define ratio $S_A = \frac{A(I)}{OPT(I)}$ (for maximization problem), where $A(I)$ is the value of the objective function obtained by A and $OPT(I)$ is the optimal value.

Absolute performance ratio S_A for the algorithm A is

$$S_A = inf\{r \geq 1 : \bigwedge_{I \in D_\Pi} S_A(I) \leq r\}.$$

Asymptotical performance ratio S_A^∞ is

$$S_A^\infty = inf\{r \geq 1 : \bigvee_{n \in Z^+} \bigwedge_{I \in D_\Pi, OPT(I) \geq n} S_A(I) \leq r\}.$$

The closer S_A, S_A^∞ are to the 1 the better algorithm is.

For some combinatorial problems it can be proved that there is no hope of finding an approximation algorithm of certain accuracy (i.e. this question is as hard as finding a polynomial-time algorithm for any NP-complete problem).

Analysis of the worst case behaviour of an approximation algorithm may be complemented by an analysis of its mean behaviour. This can be done in two ways . The first consists in assuming that the parameters of the instances of the considered problem Π are drawn from certain distribution D and then one analyzes the **mean performance** of algorithm A. One may distinguish between **absolute error** of an approximation algorithm, which is the difference between the approximate and optimal solution values and **relative error** which is the ratio of the two. Asymptotic optimality results in stronger (absolute) sense is quite rare. On the other hand asymptotic optimality in the relative sense is often easier to establish [19, 21, 24].

It is rather obvious that the mean performance can be much better than the worst case behaviour, thus justifying the use of given approximation algorithm. A main obstacle is difficulty of proofs of the mean performance for realistic distribution functions. Thus, the second way of evaluating the mean behaviour of approximation algorithms, consisting of simulation studies, is still used very often. In the later approach one compares solutions, in the sense of the values of a criterion, constructed by a given approximation algorithm and by optimization algorithm. This comparison should be made for a large representative sample of instances. There are some practical problems which follow from the above statement and they are discussed in [23].

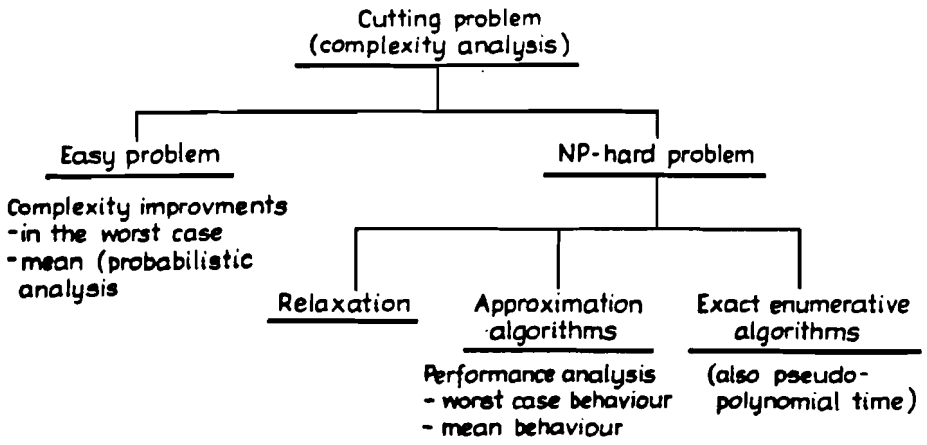


Fig.1. An analysis of cutting problem - schematic view.

The third and last way of dealing with hard cutting problems is to use exact enumerative algorithms whose worst-case complexity function is exponential in the input length. However, sometimes, when the analyzed problem is not NP-hard in the strong sense, it is possible to solve it by a pseudo-polynomial optimization algorithm whose worst-case complexity function is bounded from above by the polynomial in the input length and in the maximum number appearing in the instance of problem. For reasonably small numbers such an algorithm may behave quite well in practice and it can be used in computers applications. On the other hand "pure" exponential algorithms have probably be excluded from application, but they may be used sometimes for other cutting problems which may be solved by off-line algorithms.

The above discussion is summarized in a schematic way in Fig.1.

Definitions from this Section are base for further analysis of our problem.

2.3 One-Dimensional Problem Analysis

One-dimensional problem is the easiest version of the problem considered. From its analysis we can draw conclusions as to the general problem complexity.

One-dimensional cutting problem as stated in Section 2.1 is the same as bin-packing problem so we will refer here to the results for the latter. This problem is NP-complete in the strong sense for the decision version, this comes from pseudo-polynomial transformation of 3-partition problem [12].

3-partition problem is:

Parameters:

limit $B \in Z^+$, set $A, |A| = 3q, q \in Z^+$, value $s(a_i) \in Z^+$ for every $a_i \in A, B/4 < s(a_i) < B/2, \sum_{i=1}^{3q} s(a_i) = Bq$.

Question:

does there exist a partition of A into q disjoint subsets $s_1, s_2, \ldots, s_q$ satisfying $\sum_{a_j \in s_i} = B$ for $i = 1, \ldots, q$?

Proof is easy we see that 3-partition is a special case of bin-packing problem.

Now we know that the problem will not be solved by a polynomial algorithm (if $\mathbf{P} \neq \mathbf{NP}$), yet for the fixed number of element sizes there exists linear time solution [7].

Assume p is integer such that sizes of elements are from the set $\{1/p, 2/p, \ldots, (p-1)/p, 1\}$ and we pack them into unit size box. Elementary instance E is a set of elements satisfying $\sum_{i=1}^{n} s(a_i) \leq 1$. Data of the instance may be written as a p-dimensional vector $\overline{v} = [v_1, \ldots, v_p]$ where v_i is a number of elements of size i/p. Thus every solution is a set of elementary instances and the problem can be stated as a partition of a set of elements into the minimal number of elementary instances. We see that the number K of elementary instances is fixed. We will denote them as p-dimensional vectors $\overline{b}_1, \overline{b}_2, \ldots, \overline{b}_K$ called **elementary vectors**. Now our problem can be formulated as an integer linear programming:

find $\alpha_1, \alpha_2, \ldots, \alpha_k$ that minimize $\sum_{i=1}^{K} \alpha_i$,

subject to $\sum_{i=1}^{K} \alpha_i \overline{b}_i = \overline{v}$ and $\alpha_i \geq 0$.

A general version of integer linear programming is strongly NP-complete, but for a fixed number of variables K it can be solved in polynomial time [20]. Using the above transformation of the input data one may solve the problem in question in linear time. This is rather a theoretical result since a number of variables for practical situations may be great. Then complexity function though linear in the number of elements has a large constant before it. This constant grows exponentially with K.

There exists a number of approximation algorithms for bin-packing problem (thus for one-dimensional cutting). We are going to mention only most important.

First fit (FF) algorithm - assigns element to the box with the lowest possible number.

Best fit (BF) algorithm - assigns element to the box with the minimum remaining capacity.

Let S_{FF}, S_{BF} denote the absolute performance ratio for the FF and BF algorithms respectively and C^* - a number of boxes used by an optimal solution. Then it can be shown [25] that

$$S_{FF} = S_{BF} = \frac{17}{10} + \frac{2}{C^*}.$$

First fit decreasing (FFD) algorithm is a FF algorithm with elements assigned in nonincreasing order of their sizes.

Best fit decreasing (BFD) algorithm is a BF algorithm for elements scheduled in nonincreasing order of their sizes.

From [14, 18] the asymptotic performance ratios for FFD and BFD are known (here sizes of the elements α are drawn from the interval $(0, a]$) :

$$S^{\infty}_{FFD}(\alpha) = \begin{cases} \frac{11}{9} & \text{for } \alpha \in (\frac{1}{2}, 1] \\ \frac{71}{60} & \text{for } \alpha \in (\frac{8}{29}, \frac{1}{2}] \\ \frac{7}{6} & \text{for } \alpha \in (\frac{1}{4}, \frac{8}{29}] \\ \frac{23}{20} & \text{for } \alpha \in (\frac{1}{5}, \frac{1}{4}] \end{cases}$$

$$S^{\infty}_{BFD}(\alpha) = \begin{cases} \frac{11}{9} & \text{for } \alpha \in (\frac{1}{2}, 1] \\ \frac{71}{60} & \text{for } \alpha \in (\frac{8}{29}, \frac{1}{2}] \\ \frac{7}{6} & \text{for } \alpha \in (\frac{1}{4}, \frac{8}{29}] \\ 1 + \frac{k-2}{(k-1)k} & \text{for } \alpha \leq \frac{1}{4} \; k = \frac{1}{\alpha} \end{cases}$$

(The last line of $S^{\infty}_{BFD}\{\alpha\}$ is a proposition only).

Some other approximation algorithms are surveyed e.g. in [10].

In this Section we have shown that one-dimensional cutting problem in general case is NP-complete in the strong sense and we should not expect polynomial algorithms. Best approximation algorithms generate solutions worse about 20% than optimum in the worst case, in practice an average difference is less than 10%.

3 Two-Dimensional Regular Cutting Problem

3.1 Introduction

The problem of two-dimensional regular cutting defined in 2.1 has several variants. For all the cases the common assumptions are

- pieces can not overlap each other or the edges of material
- pieces can not be inverted (as in the mirror).

For some cases a raw material may consist of rectangular sheets of material then the objective is to minimize its number. Sometimes the ribbon of material is given then one has to minimize a length while a width is constant. On the other hand, if the area of the material is one rectangle then the aim is to pack elements that minimize a waste. Rotations of elements are rather not considered and if any then 90 degrees rotations are assumed. In some cases only guillotine cuts are allowed, i.e. from edge to edge parallel to the other pair of edges.

We know that one-dimensional version has been already NP-complete in the strong sense. Thus in such a situation one can construct exponential and optimal algorithms or polynomial approximation ones. In the following sections we describe two optimal algorithms and several approximation ones adjusted to the different versions of the problem.

3.2 Iterative Combinatorial Algorithms

Christofides and Whitelock's branch and bound algorithm [8].

This algorithm solves a single sheet problem and it is based on a tree - search procedure. It limits the number of nodes imposing necessary conditions on the optimality of patterns to be cut. This is done by means of transportation routine and dynamic programming routine.

Assume $A_0 = (L_0, W_0)$ is a sheet of material with dimensions L_0 (length) and W_0 (width). R is a set of rectangles $R = \{(a_1, b_1), \ldots, (a_m, b_m)\}$. Every rectangle has value v_i and maximal number of appearances in the resulting pattern l_i . Every number in the problem is integer, cuts are of a guillotine type, and rotations are not allowed. The problem is to

maximize $z = \sum_{i=1}^{m} \xi_i v_i$

subject to $0 \leq \xi_i \leq l_i$, $i = 1, \ldots, m$, $\xi_i \in Z^+$ and there exists a sequence of cuts of A_0 resulting in ξ_i rectangles of the i'th type.

The algorithm has two steps - generating the tree of all possible cuts and scanning it for the best solution.

Every node of the tree represents a possible cut. During the generation phase symmetrical cuts are excluded. For example cutting of rectangle (p,q) in the point e is symmetrical with the cutting in the point p-e. Such a symmetries are excluded by analyzing in the rectangle (p,q) only points with $x \leq \lfloor p/2 \rfloor$ and $y \leq \lfloor q/2 \rfloor$ (where $\lfloor a \rfloor$ is the greatest integer not greater than a). Repetitious cuts are eliminated by imposing succession of cuts - for example if we cut at point $x = \alpha$ then every succeeding cut has to be done at $x \geq \alpha$. Only **normalized cuts** are considered (cf. fig. 2) that is in points which are linear combinations of sizes of elements. This exclude cutting with waste inside a pattern.

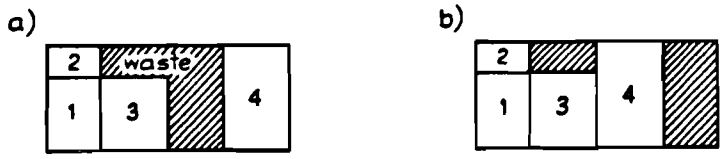


Fig.2. a) Not normalized cut. b) Normalized cut.

Possible cuts of rectangle (p,q) resulting in the elements of set R are entries of set S^q for cutting in Y direction and T^p for X direction. Now we describe how to find S^q , T^p is found in the same way. We use function $f_m(x)$ to generate S^q . This function can be computed recursively as follow (rectangles are ordered according to nonincreasing value of b_i) : for $i = 1, \ldots, m$, $x = 0, \ldots, L_0$

$$f_i(x) = \min\{f_{i-1}(x), \max_j\{b_i, f_{i-1}(x - ja_i)\}\} \; j = 1, \ldots, \min(l_i, \lfloor x/a_i \rfloor)$$
$$if \; x \geq a_i,$$
$$f_i(x) = f_{i-1}(x)$$
$$if \; x < a_i,$$
$$f_0(x) = \infty.$$

State of a node in the tree is described by the list L of rectangles cut on the path from the root. Rectangle is represented on that list by vector (p, q, x, y), (p, q) being sizes of a current rectangle and x, y are describing the following cuts if the rectangle is chosen. In order to find an optimal solution, for every node, an upper bound estimation of the objective function is computed. This estimation is computed in two ways. Suppose $H_0 \subseteq L$ is a list of rectangles that will not be cut any more. Estimation z may be computed as a total flow in a special transportation problem that assigns elements from the set R to H_0 . Upper bound estimation for nodes with still possible cuts can be computed by dynamic programming procedure for relaxed version of the problem - not considering limits l_i [13]. If computed value z^* is better than previously found $\overline{z}$, then one substitutes previous solution with the current one and the algorithm proceeds to the next node.

Wang's combinatorial algorithm [26, 21].

This algorithm is a combinatorial one that generates guillotine cutting patterns by successive adding pieces or groups of pieces to each other. These cut patterns are normalized in the sense of the previous algorithm. To avoid explosive growth of number of partial solutions the algorithm rejects solutions with a waste exceeding some percentage of stock sheet area or for the second version with a waste exceeding a percentage of the area of a partial solution.

Let us denote by S_k a partial solution generated at iteration k, by F_k - a list of all partial solutions generated during iteration k, by L_k - a list of all partial solutions generated until iteration k and by β - rejection parameter $0 \leq \beta \leq 1$. Wang's algorithm can be formulated as follows

choose β;
$L_0, F_0 := R$;
$k := 0$;
while F_k **not** empty **do**
    $k := k + 1$;
    $F_k := \{\}$;
    generate all partial solutions S_k adding elements of F_{k-1}
        to all elements of L_{k-1};
    for each S_k **do**
        if S_k fits in the stock sheet
            and the number element i appears in S_k is not greater than l_i
            and the waste in S_k is not greater than $\beta L_0 W_0$
        then $F_k := F_k \cup S_k$;
    $L_k := L_{k-1} \cup F_k$;
$M := k$;
choose the element of L_M with the least total waste.

It is shown in [26] that if the waste of the best pattern is not greater than $\beta L_0 W_0$ then this pattern is optimal (there is no pattern with a smaller waste). A modification of the above algorithm (described in [21] is done by means of dynamic programming algorithm for unconstrained number of elements [14] and it improves the way expected waste for partial solution is computed. Thus, worse solutions are rejected earlier.

3.3 Approximation Algorithms

There are many approximation algorithms. We describe only some, which, in our opinion, are the most important. If not stated otherwise the unit width of the stock sheet is assumed, a length is to be minimized, and rotations are not allowed. For a given list L of rectangles an approximation algorithm generates solution with stock sheet length A(L) while optimum is OPT(L). We use the absolute performance ratio

$$A(L) \leq \alpha OPT(L)$$

and an asymptotic one

$$A(L) \leq \alpha OPT(L) + \beta$$

Let us pass now to the algorithms.

Bottom left decreasing (BLD) algorithm. Rectangles given on the list L are ordered according to nonincreasing sizes. Put the next element from L as low and as much to the left as possible.

For every $L : BLD(L) \leq 2OPT(L)$, thus algorithm BLD generates worst case solutions 100% worse than an optimal one.

The following two algorithms [11] are so called **level oriented algorithms.** The level-oriented name comes from the fact that pieces are located in layers. The first layer bottom is a bottom of the stock sheet, the following are marked by the top of the first (that is highest) element in the preceding layer. Elements on L are ordered according to nonincreasing height.

Next fit decreasing height algorithm (NFDH) -if there is not enough room at the current (top) level to place a rectangle considered, then create a new level (fig. 3).

First fit decreasing height algorithm (FFDH) - puts rectangles at the lowest possible level and if it is not possible creates new one (fig. 4).

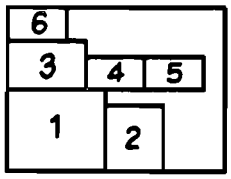


Fig.3. An example solution by NFDH algorithm.

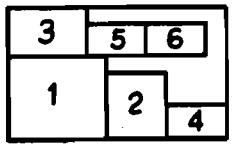


Fig.4. An example solution by FFDH algorithm.

Asymptotic performance ratio for NFDH is

$$NFDH(L) \leq 2OPT(L) + 1$$

for FFDH

$$FFDH(L) \leq 1.7OPT(L) + 7.3$$

and for sizes of elements not exceeding α

$$FFDH(L) \leq (1 + \frac{1}{m})OPT(L) + (2 + \frac{1}{m}) \text{ where } m = \lfloor 1/\alpha \rfloor$$

for squares

$$FFDH(L) \leq \frac{3}{2}OPT(L) + 2.$$

Split fit algorithm (SF) [11]. Let $m \geq 1$ be the greatest integer such that all rectangles have widths not greater than 1/m. The list of pieces is ordered according to the nonincreasing heights. Split L into two lists L_1 , L_2. L_1 consists of elements of widths greater than 1/(m+1), L_2 contains the remaining elements. First, put L_1 rectangles with FFDH algorithm then move the layers wider than $(m+1)/(m+2)$ to the bottom of the pattern down under layers thinner than $(m+1)/(m+2)$. Thus, there is a free rectangular area $1/(m+2)$ wide. Put into this area L_2 elements using FFDH algorithm. Place remaining L_2 rectangles above the pattern for L_1 (fig. 5).

Asymptotical performance ratio for SF algorithm is

$$SF(L) \leq \frac{m+2}{m+1}OPT(L) + 5.$$

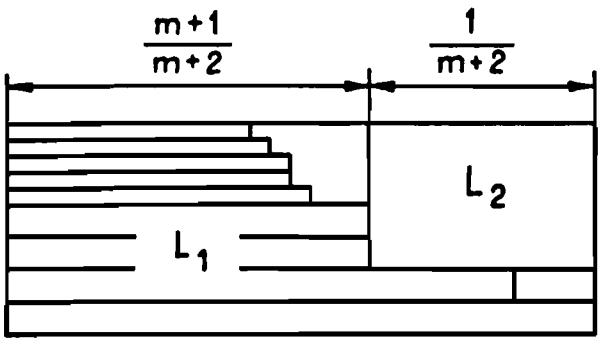


Fig.5. SF algorithm layout.

Up down algorithm (UD) [4]. This algorithm is equivalent to NFDH algorithm for rectangles thinner than 1/5; for the wider several strategies are mixed. This algorithm is a bit more sophisticated than previously mentioned and we will only give its brief outline. The algorithm splits the stock into the five regions numbered from the bottom of the stock to its upper part. In the regions $1 \leq i \leq 4$ rectangles being wide $1/(i+1)$ through $1/i$ are packed according to BL (bottom left) algorithm. Thus, there remains some free area in the right top corner. Thus more rectangles can be placed in the column from the top down. When all elements wider than 1/5 are placed in regions $1 \leq i \leq 4$, then the remaining rectangles are put into the slot between elements located by BL and column algorithm (cf. fig. 6). This is done by means of generalized next fit decreasing algorithm (GNFDH). Asymptotic worst case behaviour for rectangles not exceeding height H is

$$UD(L) \leq \frac{5}{4} OPT(L) + \frac{53}{8} H.$$

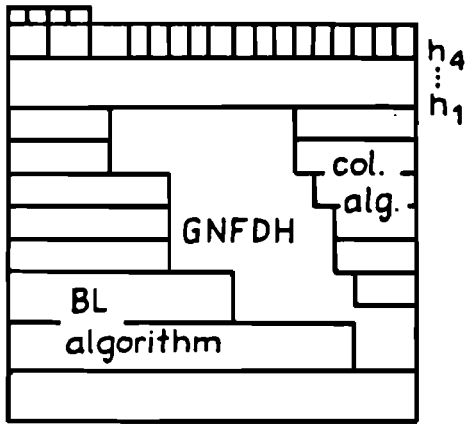


Fig.6. UD algorithm allocation layout.

Algorithms described above have to work "off line" since it is necessary to know the set of rectangles before the start, and more over these pieces have to be sorted. For certain applications however this is not possible to wait until all parts are known to sort them. For example we can not wait for the arrival of all parts to compute their allocations in the warehouse area. There are certain so-called **shelf algorithms** predestined to work "on line" without initial sorting of elements or even knowing them. These algorithms are modifications of NFDH and FFDH algorithms. Additional free space is created to handle the elements of bigger size expected to come later. The parameter r is a measure of that additional space. Every created shelf has value r^k (for some k), and an element of height h, $r^{k+1} \leq h \leq r^k$ has to be packed into the shelf of r^k height.

Next fit shelf algorithm with parameter r (NFS_r) - puts rectangle as far to the left on the highest (last) shelf as possible, and if this is not possible, a new shelf is created.

First fit shelf algorithm with parameter r (FFS_r) - puts rectangle at the lowest possible shelf as far to the left as possible, and if this is not possible, a new shelf is created.

It can be shown [5] that for $0 < r < 1$ and rectangles not higher than H

$$NFS_r = \frac{2}{r} \text{ OPT(L)} + \frac{H}{r(r-1)}$$

$$FFS_r = \frac{1.7}{r} \text{ OPT(L)} + \frac{H}{r(r-1)}$$

For the case with multiple stock sheets of the same limited sizes the objective is to minimize a number of sheets used. There exists [9] **HFF algorithm** for this purpose. HFF is a mixture of FFDH and FFD. First, according to FFDH a pattern with levels in the unlimited height stock is constructed, then levels are assigned to the stock sheets according to FFD algorithm. Asymptotic performance ratio for HFF is

$$HFF(L) \leq \frac{17}{8} \text{OPT(L)} + 5.$$

In this section a very short insight into the group of the algorithms dealing with two-dimensional regular cutting has been presented. There are two main groups of algorithms - optimal exponential combinatorial ones and those based on approximation approaches with the worst case bounds known. Due to the progress in the computer hardware speed the sizes of problems that can be solve by optimal algorithms are growing [21]. On the other hand average behaviour for realistic cases of approximation algorithms is much better than the worst case estimates suggest.

4 Irregular Two-Dimensional Cutting Problem

4.1 Introduction

This problem admits any shapes of elements. Strong NP-completeness of decision version of the problem implies the lack of the polynomial optimization algorithm. Worse still, as far as we know, there are neither algorithms with known worst case behaviour bounds nor algorithms computing optimal solution in any way. In practice, only experimental evaluation and comparison on the base of some objective function (waste, time), is possible. The only method optimal in some sense has been proposed by Adamowicz [1]. This method involves iterative solution of an integer programming problem followed by an adjusting procedure, which generate new constraints for the next iteration until an optimal solution is constructed. However, this approach is so complex that experimental program is either not completely usable or implements very simplified version of the method.

The other methods known for the problem in question are heuristics using different approaches to the problem. These algorithms though polynomial and approximate consume a lot of time involving hard numerical computations. From this fact we can draw conclusions: there is a trade-off between the solution time and quality of the solution. In this context the importance of hybrid - semiautomatic methods increases, where tentative solution is automatically generated and the interactive improvements are allowed by conversational display unit.

We outline below ideas of four methods: by Adamowicz and three heuristics [2, 3, 15]. The first and the second heuristics have been implemented in the program described in second part of this report.

4.2 Algorithm by Albano-Sapuppo

This algorithm [2] is based on the search method for optimal solution in the directed graph of all partial solutions using several heuristic techniques that increase the search power. Pieces are assumed to be irregular polygons without holes, the sheet is a rectangle. Discretized step rotations are allowed. The goal is to minimize the waste or (better) the length of produced packing.

Many problems in artificial intelligence and operations research are solved by a technique based on searching through a "space" of candidate solutions. The above approach utilizes this technique. The set of states reachable from the initial state can be seen as a directed graph with nodes - states of allocation and arcs - allocation operations. The solution is a search process for a path from the initial state to the member of the set of final nodes. Search process can be organized in the following way:

1. Put the start node on the list GENERATED.
2. If GENERATED is empty exit with failure
3. Select a node from GENERATED according to some rule **R** and put it on a list EXPANDED, call it n.
4. If n is a final node exit with a solution path.
5. Expand n, that is generate all its successors. If there are no successor go to 2, otherwise put them on GENERATED and go to 2.

Usually rule **R** selects a node with the smallest **evaluation function** which is a sum of an estimate of the cost of the path from the starting starting to the current one and estimate of the cost of the path from the current node to the final one.

Let us consider A_0 - an initial allocation containing no elements. A_i is a final allocation if there are no more elements to allocate. **Added waste** for the allocation A_i of piece p_i is defined as follows

$$added\_waste(A_i) = space(A_{i-1}) - (space(A_i) + area\_of(p))$$

where $space(A_i)$ is the area on the right side of the profile (rightmost borders of rightmost pieces), i.e. the area which may be used to allocate piece p_i (Fig. 7).

The $wasteA_i$ is recursively defined as

$$waste(A_0) = 0$$

$$waste(A_i) = \begin{cases} waste(A_{i-1}) + added\_waste(A_i) & \text{if } A_i \text{ is not final} \\ wl - \sum_i area\_of(p_i) & \text{otherwise} \end{cases}$$

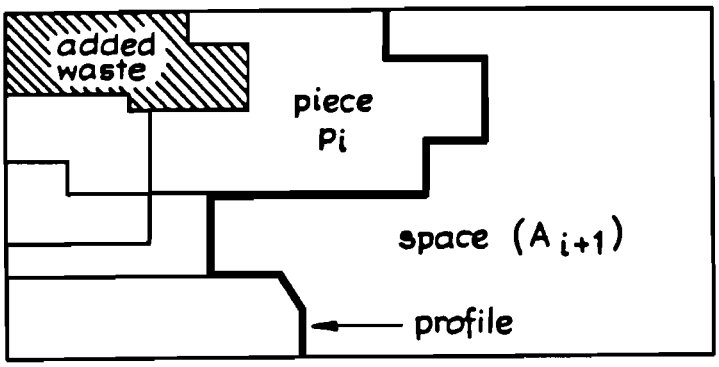


Fig.7. Partial solution pattern.

In order to transform the optimal allocation problem into the search of an optimal path in a state space, an initial state corresponds to A_0 , the cost of an arc from S_i to S_{i+1} is the added waste produced by allocation A_{i+1} . The procedure to implement the heuristic search method can be stated as follow:

```
begin
initial conditions;
input pieces and stock sheet descriptions;
let the CURRENT_NODE be the initial state;
    while CURRENT_NODE ≠ final node do
        for all pieces left to be allocated do
            for all orientations do
                apply placement policy; {waste computation}
                apply evaluation function and
                append successor to the list GENERATED
            end
        end;
    set the CURRENT_NODE EXPANDED;
    let the CURRENT_NODE be the "best" successor in GENERATED;
    end;
plot solution;
end
```

There are some techniques to increase heuristic search power because the above procedure can not be applied to any realistic applications without rejecting numerous "bad" nodes.

- Evaluation function - the problem is how to evaluate cost from the current to final node. It should always be lower than the waste that would result in the optimal solution if the piece in question were to be included. A possibility is 0, but it has been preferred to drop the admissible property because the optimality of the solution is not critical and "good" will be enough. Thus the authors suggest a constant percentage of unplaced elements as an estimation of the cost.
- Successor limitation - since the step of evaluations for all pieces and orientations is one consuming the most of the time, several limitations have been added.
    1. From unplaced pieces only the one which produces the leftmost lowest allocation will be considered in step 2.
    2. Only the fixed number of leftmost allocated pieces are preserved for step 3.
    3. For allocation from 2 only the fixed number of successors will be generated.
    4. When the list of generated nodes becomes full, the tree is pruned by erasing the node with the highest evaluation function.
- Evaluation function discretization - continuation of the search along paths with small differences is allowed only within some precision.
- Expansion band - when the search is at the k-th level, the next node to be expanded has to be at level at most k-t where t is given threshold.
- Termination condition - at the end of the routine after the first final node is found the procedure develops all the possible search trees within the expansion band and the final solution will be the best one.
- Profile simplifications - at each step the profile is simplified in order to exclude all the areas on the left side of the vertical line through the leftmost point of the last allocation pieces.

There are several notions related to this algorithm important especially in finding piece allocations.

No-fit-polygon (NFP) for a pair A,B of pieces - completely describes all those positions where the reference point of B may be placed in order to have B touching A without overlapping (fig.8).

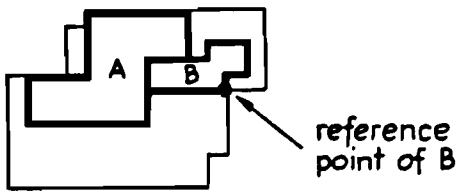


Fig.8. NFP for A and B.

Allocation region for a given resource and piece - the area in which the reference point of the piece can validly fall (fig.9).

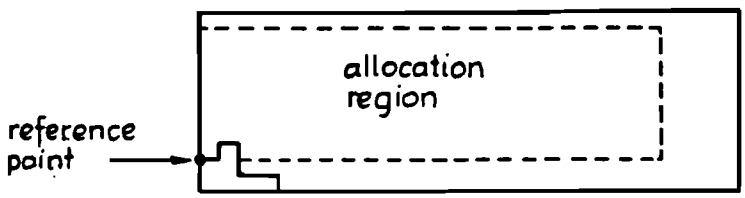


Fig.9. Allocation region.

This algorithm has been tested thoroughly and its performance and reliability indicates that it favourably compares with some others.

4.3 Algotithm by Art

This algorithm [3] is very similar to the Albano-Sapuppo algorithm in the way of handling geometrical entities. It was the base to create previous algorithm. Here, more stress is put to geometrical problems while before mainly heuristic search organization has been considered. This Section may be both description of independent algorithm and supplement for Albano-Sapuppo algorithm. There are the following assumptions made

1. There is distinguished piece direction (orientation) parallel to the distinguished direction of the material let us say X axis. Thus rotations are not allowed.
2. Flipping of pieces is not allowed (mirror symmetry).
3. The stock sheet is a rectangle with given width, and its length is to be minimized.

Every piece is defined by the sequence of line sections approximating its edge with a precision required. We exclude here any curve edges because of the growth of computational complexity. Distinguished element direction is parallel to X axis. There is a distinguished point in the sequence of vertices called reference point and its location defines uniquely the element position. Both concave and convex polygons are admitted but because of higher computational complexity concave objects are excluded and approximated to the convex equivalent.

Important notion is the **envelope** - it is a sequence of line sections defined for every piece type. This is a trace of reference points created during moving element around allocation region in contact with its border. At the start of algorithm envelope is a rectangle but successively after allocation of any element it is modified with the no-fit-polygons (NFP - Section 4.2) (fig. 10). Envelope can be understood as a border of an allocation region.

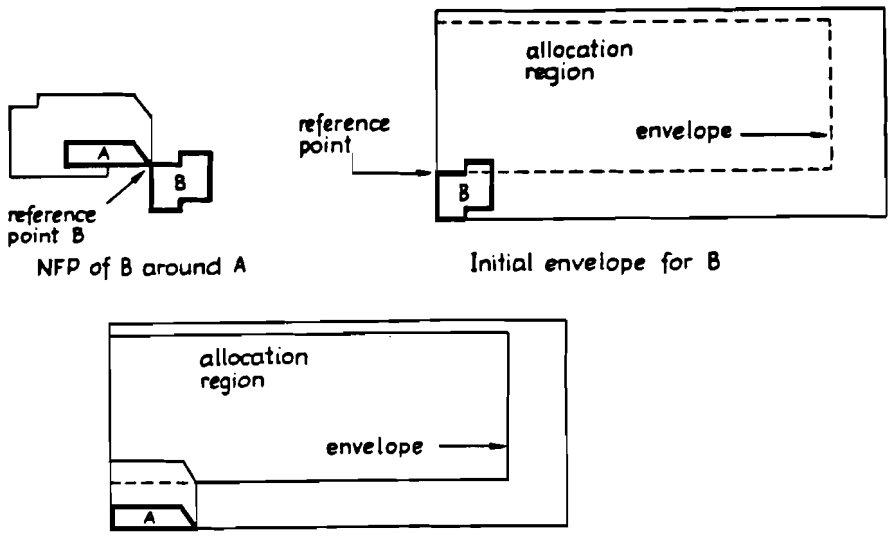


Fig.10. Modification of element B envelope after allocation of element A.

Process of allocation is deterministic and sequential. Reference point defining element location is to be placed on the envelope - this guarantees contact with other elements without overlap. In order to minimize total waste, elements are allocated from the left side to the right according to the following heuristic rules:

1. Place element with the minimal envelope X dimension value first.
2. Place element in a certain place with minimal added waste.
3. Split big groups of elements into the smaller one or into the separate elements to put in certain places separately.
4. Place element with greater area first.
5. Place element minimizing stock sheet length first.

Position on this list do not reflect importance of rules, it should be rather experimentally adjusted to typical problem instances.

4.4 Optimal Algorithm by Adamowicz

This algorithm [1] allows the most general version of the problem to be solved. Various elements and stock sheets as well as multiple linear, logical and geometrical constraints associated with them are considered. Solution is obtained by iterative application of a two-stage procedure. The first stage is a linear programming problem; its solution minimizes linear objective function subject to linear constraints. The second, geometrical, stage checks if the set of elements can be allocated feasibly satisfying geometrical constraints. If the solution does not exist, then new linear constraints resulting from the information obtained in the second step are generated for the new iteration.

This method allows elements and stock to be holed, irregular and even not continuous. There are geometrical constraints of the two types: absolute - that bound locations on the stock area and relative - defined in relation to the other elements. Logical and linear constraints are for example number of elements of certain type required, a maximal number of types of elements, a ratio of quantities in which two types of elements are cut and others. The algorithm can be stated as an iterative application of the three following procedures.

1. A solution of a linear programming problem with rejection of redundant constraints.
2. Checking if the linear programming solution satisfies geometrical constraints and if not saving an information about feasible allocations.
3. Creating new linear constraints, including information gained in step 2.

Elements and material area are defined by sequence of vertices on its polygonal border. If the element is not compact (i.e. includes holes) it has to be defined as a group of compact elements with an additional relative location constraint. A location of an element is given by three parameters: x, y and rotation angle from an initial orientation. During the processing phase of geometrical constraint an envelope (see Section 4.3) for absolute constraints and NFP (Section 4.2) for relative constraints, are computed.

Linear programming phase gets into the consideration:

- linear constraints of the number of types of elements;
- logical constraints on the presence of types of elements, this can be converted into linear constraint as well;
- geometrical constraints on the relative location of certain types of elements - since this is inherently nonlinear mainly geometrical phase of algorithm deals with this kind of constraints;
- linear objective function maximizing income or number of elements allocated, density of allocation and/or minimizing the cost, waste etc.

As a result of linear programming phase solution one gets a set of elements to be allocated.

Geometrical phase checks if allocation of elements chosen in the previous phase is possible. The elements are considered according to their nonincreasing areas. This procedure searches in the space of all possible locations of elements maximizing number of elements allocated .

As we can see this algorithm is very complex and rather difficult to handle in practice. Experimental programs are either completely inoperable or are simplified versions of the method. Let us note that the computational complexity of the geometrical phase grows

exponentially in the number of objects and orientations due to the search of candidate allocations, thus this method is rather a theoretical one.

4.5 Algorithm by Gurel

Every element is represented in this method [15] as a node in the graph. There is an arc between nodes if the corresponding elements are touching each other (or in other words are in contact). Rectangular area of the stock is represented as a disk called marker disk. Nodes corresponding to pieces in contact with the stock border are on the circle of marker disk, while nodes inside it correspond to elements without common points with edges of the raw material. In this way, a graph reflecting elements allocation inside the marker disk, has been created. There are vertical paths (strings of nodes) inside marker disk with at least one node on the border of that disk. The first string of that type corresponds to pieces in contact with the left border of material; we denote it **boundary break - BB1** for short. The second string of that type consists of nodes in contact with the right border of the stock area and will be denoted **BB2**. All other vertical strings of nodes between BB1 and BB2 we name **intermediate breaks** (IBn -for short, n being IB number). Interesting feature of the solution is that in the final layout there must be at least one horizontal string of pieces forming horizontal break. This will be called cobreak. Cobreaks as well as breaks may either lay along the boundary of marker circle or cross the marker disk by joining BB1 and BB2. Therefore there are boundary cobreaks or intermediate cobreaks. During the implementation of the method it has been observed that the biggest waste is created at the (right side) end of the layout. Therefore IBs are allocated according to minimal waste starting from both ends of the layout to the center of area. Thus, the method by Gurel can be formulated as follows

1. Initial computations.
2. Create boundary break BB1 and then BB2.
3. While not allocated elements exist, create an internal break IB and move it to the right or left group of breaks.
4. Join left and right groups of breaks.

The algorithm requires some additional parameters a, b, c, s. Coefficients a, b, c are used to create four groups of elements relative to their areas. Let us denote by P_{max} area of the largest element and by P an area of an element considered. There are the following groups of pieces (depending on the areas):

L1: $aP_{max} \leq P$
L2: $bP_{max} \leq aP_{max}$
M: $cP_{max} \leq P \leq bP_{max}$
S: $P \leq cP_{max}$

Elements from L1 are preferred in BB1, from L2 and L1 - in BB2, M - in IBs. Elements from S are not allocated by this algorithm and should be placed interactively by an operator. This follows an observation that big differences of element sizes reduce quality of the solution. In practical cases a,b,c are set to a – 40-85%, b – 25-60%, c – 0-25%. The bigger the differences in area sizes are, the greater a,b,c should be.

This method admits rotations of elements with a given step.

Now, in the short outline we describe some procedures of the algorithm.

During the initial phase, for every element and every orientation the following parameters are computed:

- area,
- reference point,
- coordinates of boundary points $BP_1, \ldots, BP_8$ (fig. 11),
- waste areas $\delta_1, \ldots, \delta_8$,
- waste $\tau_{A1}, \ldots, \tau_{C4}$ for appropriate sweeping directions

$$\tau_{A1} = \delta_7 + \delta_8 + \delta_1 + \delta_2 + \delta_3 \qquad \tau_{A2} = \delta_3 + \delta_4 + \delta_5 + \delta_6 + \delta_7$$
$$\tau_{B1} = \delta_5 + \delta_6 + \delta_7 + \delta_8 + \delta_1 \qquad \tau_{B2} = \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5$$
$$\tau_{C1} = \delta_7 + \delta_8 + \delta_1 \qquad \tau_{C2} = \delta_3 + \delta_4 + \delta_5$$
$$\tau_{C3} = \delta_1 + \delta_2 + \delta_3 \qquad \tau_{C4} = \delta_5 + \delta_6 + \delta_7$$

During the initial phase elements are assigned to groups L1 , L2 , M , S.

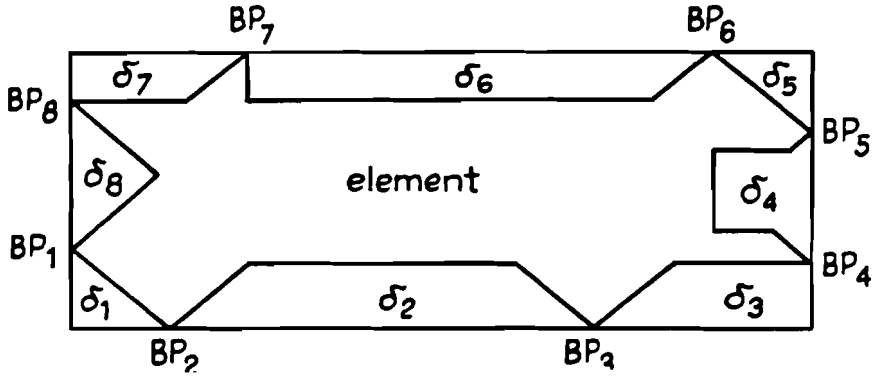


Fig.11. Object's boundary points and waste areas in Gurel's algorithm.

In order to create BB1 and BB2 we choose groups of elements minimizing waste between the borders of stock and element. Bigger pieces from BB1, BB2 are considered first. In order to check if the element fits into the BB1 or BB2 we evaluate how deeply the element in question may coincide with elements previously allocated in the string and compare the result with the width of element and width of a "slot" in the string. Internal breaks (IBs) are created a little bit easier only with comparisons of wastes τ_{C3}, τ_{C4} . The components of IB are moved to each other in the vertical direction. In order to minimize the waste inside IB, pieces are moved horizontally within some band with a given stepts.

This method seems to be faster than previously described due to its graph theoretic approach and good heuristic methods to carry on computations. This is done with a little reduction of efficiency in area usage. There are several factors influencing efficiency of algorithm, for instance number of elements, number of types of elements, parameters a, b, c, s, width of the stock sheet etc. This method seems to fit well into semiautomatic approach requirements.

5 Conclusions

In this paper, after preliminary complexity investigations, we have described four methods for irregular two-dimensional cutting. The first three of them are a bit similar in the geometrical notions used. We think that Adamowicz approach though difficult to handle in practice, shows interesting directions to create more general systems for cutting problem. The three remaining methods are applicable in practice and broadly described in references. In the next part a decision support system, using the two of the described methods, will be described.

Part II
Decision Support System for Cutting Irregular Shapes - Implementation and Experimental Comparison

6 Introduction

This part contains a description of the decision support system (DSS) for cutting irregular shapes. The basis of this system are two methods described in the previous part seriously changed and adjusted for solving the problem in question. This system has been implemented on an IBM PC working under DOS operating system.

Given a set of elements and a sheet of rectangular material with a constant width, find an allocation of elements in the sheet minimizing waste, that is a length of material required. Additionally we have assumed

1. Every element must lay entirely in the stock sheet.
2. No overlap of elements is admitted.
3. Flipping the element from "left" to the "right" side (mirror symmetry) is not allowed.
4. Edges of element are either sections of line or sections of a circle. Element can be concave or convex (Holes in elements must be treated separately).
5. Rotations with some step depending on the algorithm, are allowed.

As we already mentioned two modified approaches have been implemented in the Decision Support System presented.

The Albano-Sapuppo's approach [2] deals with hard irregular cases minimizing waste generated during allocation of every element. Elements are allocated at the left end of the stock sheet and then placed on the right side of the elements already allocated. At the current stage of computations an element minimizing waste is chosen.

Gurel's approach [15] is well defined for elements of similar size and allowing for their clustering into columns. Therefore the smallest elements are eliminated from automatic allocation. This set of small elements can be allocated interactively or with other method. It is also possible to force allocation of all elements.

Thus, the user himself is able to choose the solution or method that fits his needs best. In general we can say that Gurel's method seem to be faster while Albano-Sapuppo's generate solutions with a lower waste. Gurel's algorithm deals with a class of elements well defined to cluster, Albano-Sapuppo's algorithm does not have such a preference.

In the following Sections we describe how to use the DSS for irregular cutting, data files formats, some operational conditions, and some test results.

7 How to Use DSS ROZKROJ

On the distribution disc there are folloing files:

ROZKROJ.EXE	—	DSS program;
ROZKROJ.TXT	—	short help file for the ROZKROJ.EXE;
TEST1.IN - TEST10.IN	—	examples, input data files;
TEST1.IN1 - TEST10.IN1, TEST1.IN2 - TEST10.IN2		
	—	examples, intermediate solution files.

To start the program simply write ROZKROJ. Please be aware that any data file, solution file or auxiliary file is to be fetched from or written to the current disk and directory. Thus, there should be enough free space.

After the start of program execution the first screen is displayed. Next the main menu screen appears. We have several options here. Any of them can be chosen in the two ways

- pressing the digit key corresponding to the option number in the menu;
- moving lit up bar, up and down with cursor keys we select option and by pressing ENTER key confirm choice.

ESC key breaks program execution at this point. There are the following options in the main menu.

1. Quit the program.
2. Gurel's method.
3. Albano-Sapuppo's method.
4. Display solutions.
5. Program help.

Now, every option will be described.

Option 1. Quit the program — immediately breaks program execution and exits to DOS.

Option 2. Gurel's method — during execution of this part of the program new solutions are computed. First of all we have to give some data to the program. All file names with extension "IN" in the current directory contain different data sets. They are displayed name after name. In order to choose any file we can do two things:

1. press first letter of the name - be aware that only the first file beginning with this letter is chosen (without any additional confirmation);
2. move the lit up bar up, down, left and right to the required file and press ENTER to start computations.

ESC key breaks execution of this part of program and returns to the main menu.

Data in the file must correspond to the format described in the next section. Data from the file is preprocessed to find out some kind of errors and to approximate sections of circle with segments of lines. If any error appears then special message is issued and program breaks the execution. If there was no error new file with extension "IN1" is created, it is an intermediate data file used by the Gurel algorithm procedure. Then program starts

computations. Gurel algorithm routine displays line of twenty "*" with "—" after the initial and final phase is finished. Allocation of any element in a boundary break or an intermediate break is certified by one "*" sign. After filling a certain column of elements, previously issued "*" signs are followed by space and stars again up to twenty characters in the line. An allocation of boundary break one and boundary break two is certified with "BB1" and "BB2", respectively. An allocation of an intermediate break is certified by "IB". After all parts are allocated the new solution file is generated with the same name but with automatically added extension "IN2". This is a new intermediate format file with elements still in approximated form. Then program abandons Gurel method procedure and starts displaying solution in the graphical form. At this point program switches to the option 4 where current solution is processed.

Option 3. Albano-Sapuppo's method — during execution of this procedure new solutions according to the Albano-Sapuppo's algorithm are computed. The way this routine is handled is identical to Gurel's method described above. The only difference comes from the fact that during computations of a new solution only numbers of allocated elements are displayed.

Option 4. Display solutions — this procedure deals mainly with graphics but not only. In order to display any solution from certain file one has to choose it in the same way input data files have been chosen. Now, all files in the current directory with extension "IN2" are considered.

When solution is displayed we can save it as the final solution or print it. If we save the solution then it is converted from approximated format back to the initial format with sections of circles. Program displays new file name and waits for pressing any key. Then the new file is stored with an "OUT" extension.

Option 5. Program Help — this routine displays help text file in pages. In the current directory "ROZKROJ.TXT" file must exist otherwise program issues appropriate message and continues. We can swap pages with PGUP and PGDN keys. ESC returns program to the main menu.

8 Data File Format

The DSS uses files with extension "IN" as an input, with extension "OUT" as an output. It generates also two intermediate type files with extensions "IN1", "IN2". Figure 1 explains which module of the DSS utilizes and generates appropriate types of file.

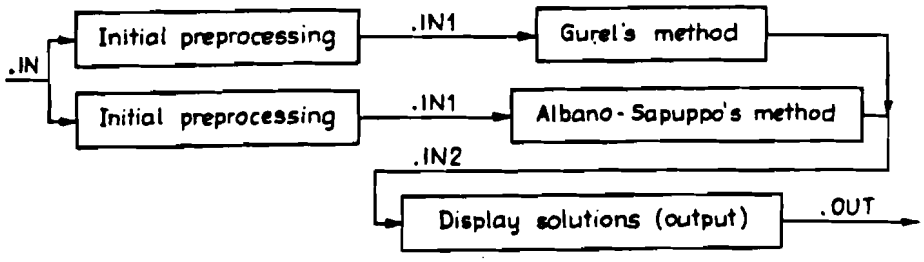


Fig.12. DSS ROZKROJ data file name extensions.

Now we describe input and output data file format. Let us denote by fl - floating point

number with at most ten characters (sign, at most five digit integer part, decimal point, at most three digits fraction), and by int - integer in the range 0 to 32767.

Input file format. Input file is a text file named with extension "IN". It must have a structure as follows

1. Width of the stock sheet (fl).
2. Number of different element shape types (int)
3. Parameters to the algorithms (all fl):
   - rotation step for the Albano-Sapuppo routine;
   - percentage parameters for the Gurel's routine;
   - step length - both routines.
4. Structure of elements. For every type of element shape one has to define:
   - number of elements of this kind (int);
   - number of vertices of the shape (int);
   - description of vertices in the clockwise order (fl)
     - x coordinate;
     - y coordinate;
     - radius of the circle if the current vertex starts a section of a circle as a part of element's edge. If the radius is negative then this section of circle causes element to be concave (line section from the beginning of circle section to the end is outside the element). Zero if it is a section of line.
     - x , y coordinates of the center of that circle, if the edge is a segment of line then components of the normal to this line vector.

Every entry of this specification is obligatory. Numbers should be separated by space or CR character.

Input file is a text file and can be prepared with any text editor. Specialized program for generation of input data files is described in other paper.

Output file format.

Output file has a name with an "OUT" extension. It has the following structure

1. Width of the stock sheet (fl).
2. Required length of the stock sheet (fl).
3. Total number of allocated elements of all types (int).
4. Description of elements allocation. Elements in this file are ordered according to the order of different shape types in the input file. For every allocated element
   - number of vertices (int);
   - description of vertices in the clockwise order (fl)
     - x coordinate;
     - y coordinate;

- radius of the circle if the current vertex starts a section of a circle as a part of element's edge. If the radius is negative then this section of a circle causes element to be concave (line section from the beginning of circle section to the end is outside the element). Zero if it is a section of line.
- x, y coordinates of the center of that circle, if edge is a segment of line then components of the normal to this line vector.

9 Operational Conditions

This DSS can operate on the IBM PC and compatible computers. Suggested minimal hardware configuration is 640 kB of RAM; CGA, HGC or VGA graphic card; floppy disc drive. Optionally hard disc drive and printer. Hard disc device simplifies data operations because all disc operations are executed in the current (where the DSS was started) directory.

10 Experimental Results

Results from several tests are presented in table 1. In that table description of a set of elements, quality of solution (percentage waste), time of computations, are given. Printouts from several example allocations are attached at the end of the paper.

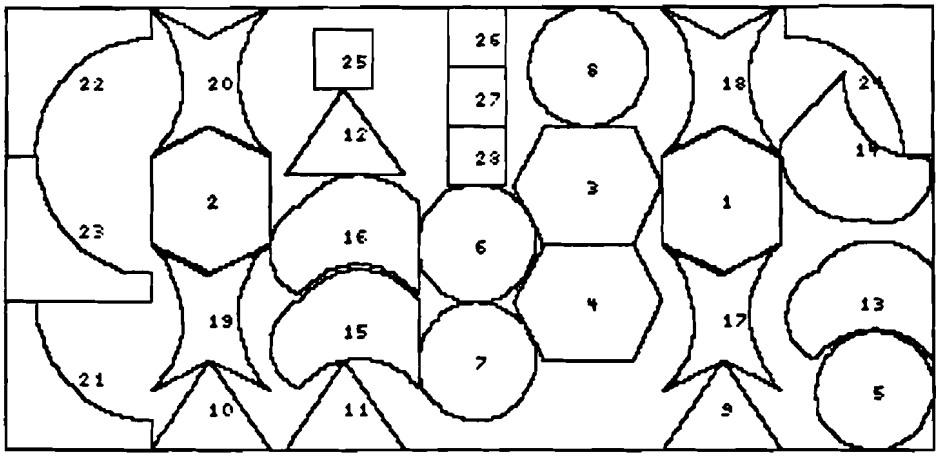
Table 1. Results of automatic element allocation with DDS ROZKROJ V.2.0.

Test file name	Description of element set	Precentage waste	Computations time	Sheet length
TEST1G [1]	8 convex heksagons	36%	< 1min	20.00
TEST1A [2]	as above	29.5 %	< 1 min	18.15
TEST2G	10 circles	36.3%	1 min	20.00
TEST2A	as above	35.7%	1 min	19.80
TEST3G	10 triangles	45.5%	< 1 min	11.00
TEST3A	as above	37.6%	3 min	9.61
TEST4G	10 concave elem. (3 arcs,line)	25.8%	1 min	19.65
TEST4A	as above	18.6%	1 min	17.90
TEST5G	12 various elem.	35.3%	1 min	22.87
TEST5A	as above	28.2%	2 min	20.60
TEST6G	10 concave elem. (6 vertices, 2 arcs, 4 lines)	59.6%	1 min	23.17
TEST6A	as above	57.0%	< 1 min	21.78

[1] G stands for Gurel's method.
[2] A stands for Albano-Sapuppo's method.

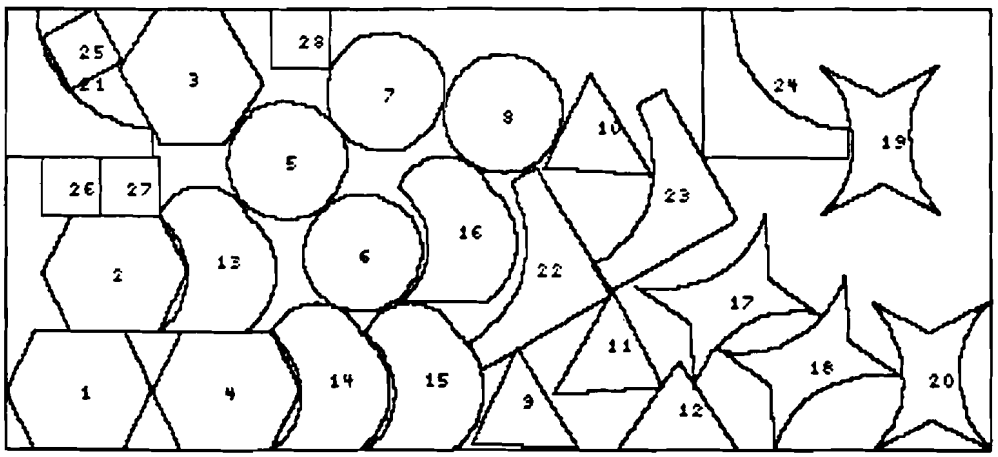
Table 1. continued

Test file name	Description of element set	Precentage waste	Computations time	Sheet length
TEST7G	13 various elem.	37.7%	2 min	20.61
TEST7A	as above	35.4%	1 min	19.86
TEST8G	12 various elem.	33.7%	< 1 min	14.00
TEST8A	as above	38.9%	2 min	15.22
TEST9G	19 various elem.	38.9%	3 min	28.75
TEST9A	as above	36.8%	5 min	27.79
TEST10G	28 various elem.	36.1%	4 min	31.470
TEST10A	as above	39.6%	9 min	33.31
TEST11G	10 convex polygons	29.5%	1 min	13.00
TEST11A	as above	37.1%	< 1 min	14.58
TEST15G	30 triangles	37.5%	2 min	60.00
TEST15A	as above	6.3%	2 min	40.00
TEST24G	4 elem. no arcs normal angles	0.7%	< 1 min	20.00
TEST24A	as above	0.7%	< 1 min	20.00
TEST25G	twice elements from TEST24A/G	3.1%	< 1 min	41.00
TEST25A	as above	11.7%	< 1 min	45.00
TEST26G	4 times elements from TEST24G/A	3.1%	3 min	82.00
TEST26A	as above	6.5%	1 min	85.00
TEST30G	TEST25G/A other sheet width	25.5%	< 1 min	40.00
TEST30A	as above	14.7%	< 1 min	35.00
TEST31G	twice TEST30A/G elements	10.5%	3 min	66.67
TEST31A	as above	8.2%	1 min	65.00
TEST35G	TEST25A/G other sheet width	24.5%	< 1 min	31.67
TEST35A	as above	20.3%	< 1 min	30.00
TEST36G	TEST26A/G other sheet width	5.0%	9 min	50.3
TEST36A	as above	13.1%	2 min	55.00



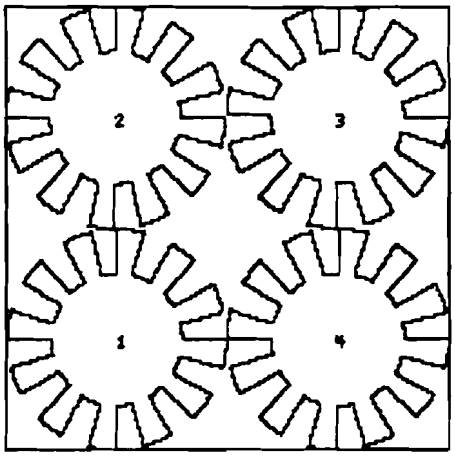
Length : 31.470 Width : 15.000 Waste : 36.1%

Fig.13. TEST10 Gurel's method.



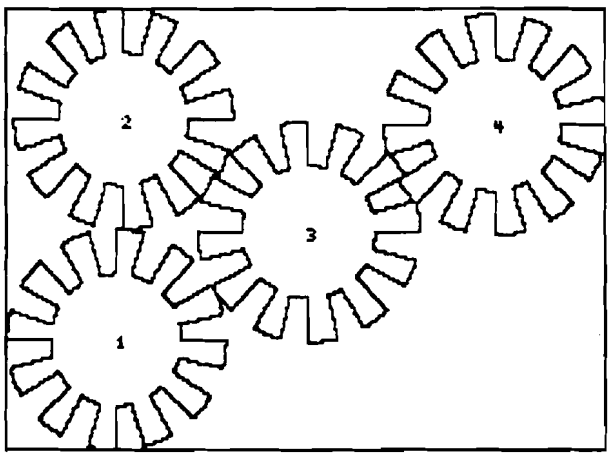
Length : 33.313 Width : 15.000 Waste : 39.6%

Fig.14. TEST10 Albano-Sapuppo's method.



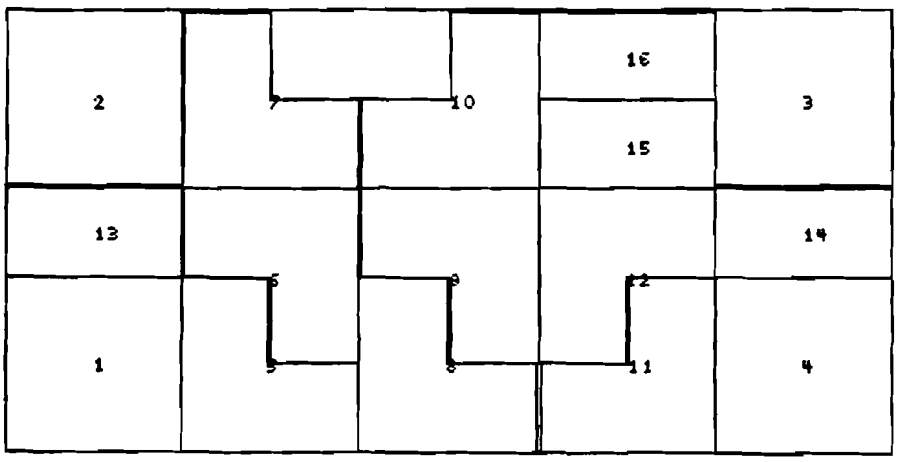
Length : 20.000 Width : 20.000 Waste : 46.5%

Fig.15. TEST21 Gurel's method.



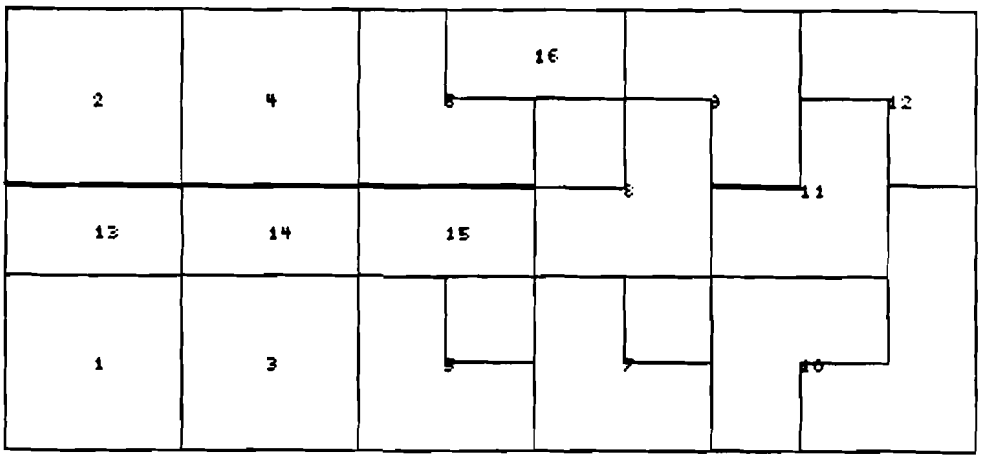
Length : 27.077 Width : 20.000 Waste : 60.5%

Fig.16. TEST21 Albano-Sapuppo's method.



Length : 50.300 Width : 25.100 Waste : 4.9%

Fig.17. TEST36 Gurel's method.



Length : 55.000 Width : 25.100 Waste : 13.1%

Fig.18. TEST36 Albano-Sapuppo's method.

References

[1] M. Adamowicz. *The Optimal Two-Dimensional Allocation of Irregular Multiply-Connected Shapes with Linear Logical and Geometric Constraints.* N.Y. Univ. Tech. Report 403-9, New York 1969.

[2] A. Albano, G. Sapuppo. *Optimal Allocation of Two-Dimensional Shapes Using Heuristic Search Methods.* IEEE Trans. on Systems, Man and Cybernetics, vol. SMC-10, No. 5, May 1980.

[3] R. C. Jr. Art. *An Approach to the Two-Dimensional Irregular Cutting Stock Problem.* IBM Report 320-2006, Cambridge 1966.

[4] B. S. Baker, D. S. Brown, H. P. Katseff. *5/4 - Algorithm for Two-Dimensional Packing.* J. of Algorithms, 2 (1981), 348-368.

[5] B. S. Baker, E. G. Jr. Coffman, R. L. Rivest. *Orthogonal Packings in Two Dimensions.* SIAM J. Comput. 9 (1980), pp 846-855.

[6] B. S. Baker, G. S. Schwartz. *Shelf Algorithms for Two-Dimensional Packing Problems.* SIAM J. Comput., 12 (1983), pp 508-525.

[7] J. Błażewicz, K. Ecker. *A Linear Time Algorithm for Restricted Bin-Packing and Scheduling Problems.* Operations Research Letters 2, No. 2, 1983, pp 80-83.

[8] N. Christofides, C. Whitelock. *An Algorithm for Two-Dimensional Cutting Problems.* Operations Research 25, 1977, pp 30-44.

[9] F. R. K. Chung, M. N. Garey, D. S. Johnson. *On Packing Two-Dimensional Bins.* SIAM J. Alg. Dis. Math. 3 (1982) No 1, pp 66 - 76.

[10] E. G. Jr. Coffman, M. R. Garey, D. S. Johnson. *Approximation Algorithms for Bin-Packing - an updated survey.* in G.Ausiello, M.Lucertin, P.Serafini (eds.) Algorithms for Computer System Design, Springer Verlag, Wien 1984, pp 49-106.

[11] E. G. Jr. Coffman, M. R. Garey, D. S. Johnson, R.E. Tarajan. *Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithms.* SIAM J. Comput. 9 (1980), pp 808-826.

[12] M. R. Garey, D. S. Johnson. *"Strong" NP-Completeness Results, Motivation, Examples and Implications.* J. ACM 25, No 4, 1978, pp 499-508.

[13] P. C. Gilmore, R. E. Gomory. *Multistage Cutting Stock Problems of Two and More Dimensions.* Operations Research 13 (1965), pp 94 - 120.

[14] P. C. Gilmore, R. E. Gomory. *The theory and Computation of Knapsack Functions.* Operation Research 15 (1967), pp 1045-1075.

[15] O. Gurel. *Circular Graph of Marker Layout.* IBM Data Processing Division, New York Scientific Center Report No 320-2965, Feb.t1969.

[16] S. S. Israni, J. L. Sanders. *Two-Dimensional Cutting Stock Problem Research: A Review and a New Rectangular Layout Algorithm.* Journal of Manufacturing Systems 1 (1982), pp 169-.

[17] S. S. Israni, J. L. Sanders. *Performance Testing of Rectangular Parts Nesting Heuristics.* Int. J. Prod. Res., 23 (1985), 437-456.

[18] D. S. Johnson. *Near-Optimal Bin-Packing Algorithms.* Ph. D. Thesis, Massachusetts Institute of Technology, Electrical Department, 1974.

[19] R. M. Karp, J. K. Lenstra, C. J. H. McDiarmid, A. H. G. Rinnooy Kan. *Probabilistic Analysis of Combinatorial Algorithms: An Annotated Bibliography.* in M.O'h Eigearthaigh, J.K. Lenstra and A.H.G Rinnooy Kan (eds.), Combinatorial Optimization: Annotated Bibliographies, J.Wiley, Chichester, 1984.

[20] H. W. Jr. Lenstra. *Integer Programming With a Fixed Number of Variables.*, Math. Oper. Res. 8, 1983, pp 538-548.

[21] J. F. Oliveira, J. S. Ferreira. *Solving Two-Dimensional Cutting Problems and Comparing Different Approaches.* unpublished paper, Instituto de Engenharia de Sistemas e Computadores, Porto, Portugal, 1989.

[22] A. H. G. Rinnooy Kan. *Probabilistic analysis of algorithms.* Annals of Discrete Mathematics 31 (1987), 1-60.

[23] E. A. Silver, R. V. Vidal, D. de Werra. *A tutorial on heuristic methods.* European Journal of Operational Research 5, 1980, 153-162.

[24] L. Slominski. *Probabilistic analysis of combinatorial algorithms: a bibliography with selected annotation.* Computing 28 (1982), 257-267.

[25] J. B. Ullman. *The Performance of a Memory Allocation Algorithm.* Tech. Rept. No 100 Princeton Univ., Electrical Engineering Department, 1971.

[26] P. Y. Wang. *Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems.* Operations Research 31 (1983), pp 573-586.

A method for minimization of piecewise quadratic functions with lower and upper bounds

Janusz S. Sosnowski
Systems Research Institute, Polish Academy of Sciences,
Newelska 6, 01-447 Warsaw, Poland.

Abstract

The paper describes methods of minimization of piecewise quadratic convex functions subject to lower and upper bounds. The presented approach may be used for solving linear and quadratic programming problems while the multiplier, proximal multiplier or penalty methods are applied to original problem.

An active set algorithm which takes into account the form of the objective function and bounds is developed. For solving a sequence of quadratic subproblems generated by the active set algorithm a numerically stable method of updating R–factors in QR factorization is adopted.

1 Introduction

The paper describes a numerically stable method of minimization of piecewise quadratic convex functions subject to lower and upper bounds. The presented approach may be used for solving linear and quadratic programming problems while the multiplier, proximal multiplier or penalty methods are applied to original problem. An active set algorithm which takes into account the form of the objective function and bounds is developed. For solving a sequence of quadratic subproblems generated by the active set algorithm a numerically stable method of updating R– factors in QR factorization is adopted.

In the paper we will deal with the following problem:

$$\min f(x) \tag{1}$$

$$f(x) = cx + (1/2)\|A_1x - b_1\|^2 + (1/2)\|(A_2x - b_2)_+\|^2 \tag{2}$$

$$l \leq x \leq u \tag{3}$$

where: $A_1 \in R^{r\times n}, A_2 \in R^{s\times n}$, $b_1 \in R^k, b_2 \in R^s, c \in R^n$, and $l, u \in R^n$ are given lower and upper bounds $x \in R^n$

The following notation will be used:

$a_i^{(k)}$ – denotes the i–th rows of matrix A_k

$b_i^{(k)}$ – denotes the i–th components of the vector b_k

x_j – denotes j–th component of x

$\|x\|$ – denotes L_2–norm of x

$(u)_+$ - denotes the vector composed of components $\max(0, u_i)$

A_k^T - denotes the transposition of matrix A_k.

The above formulation generalizes the problems of minimization faced in ordinary multiplier method for linear programming problems. It also covers subproblems in the regularized or proximal multiplier method (Rockafellar, 1976). Note that if we introduce the following notation

$$A_1 = \begin{pmatrix} A_{11} \\ \sqrt{\gamma} I \end{pmatrix} \quad b_1 = \begin{pmatrix} b_{11} \\ \sqrt{\gamma} r \end{pmatrix}$$

then the minimized function will take the form

$$f(x) = cx + (\gamma/2)\|x - r\|^2 + (1/2)\|A_1 x - b_1\|^2 + (1/2)\|(A_2 x - b_2)_+\|^2$$

The minimized function (2) is convex, piecewise quadratic, and twice differentiable beyond the set:

$$\{x \in R^n : a_i^{(2)} x - b_i^{(2)} = 0, \quad i = 1, \cdots, s\}$$

In the active set algorithm applied to the problem (1) - (3), we solve a sequence of minimization quadratic functions without constraints. After a finite number of steps, a set of indices of constraints which are active at the solution is found. That is based on the observation that at the optimal solution a certain subset of constraints is satisfied with equality. Let

$$J^* = \{j : x_j^* = l_j \quad \text{or} \quad x_j^* = u_j\}$$

where x^* - denotes solution of the problem (1) - (3).

Additionally we introduce the following set of indices

$$I^* = \{i : a_i^{(2)} x^* - b_i^{(2)} > 0\}$$

If the sets J^* and I^* were known, then the variables $x_j, j \in J^*$ could be eliminated and the problem reduced to the following unconstrained optimization problem

$$\min f_{I^*}(x) \tag{4}$$

$$f_{I^*}(x) = cx + (1/2)\|A_1 x - b_1\|^2 + (1/2) \sum_{i \in I^*} (a_i^{(2)} x - b_i^{(2)})^2 \tag{5}$$

If we solve the above problem with respect to the free variables $x_j, j \in \{1, 2, \ldots, n\} \setminus J^*$, then we will reach a solution of the problem (1)-(3). The solution meets the Kuhn-Tucker conditions, which can be formulated in the following way.

Let A_2^* be a matrix composed of rows with indices from the set I^*, and b_2^* be the corresponding vector.

The gradient of f in point x^* is defined by

$$\frac{\partial f^*}{\partial x} = c + A_1^T (A_1 x^* - b_1) + (A_2^*)^T (A_2^* x^* - b_2^*)$$

From the Kuhn-Tucker optimality condition, the following relations hold for the minimum point x^*

$$\frac{\partial f^*}{\partial x_j} \geq 0 \quad \text{if} \quad x_j^* = l_j, \qquad \frac{\partial f^*}{\partial x_j} \leq 0 \quad \text{if} \quad x_j^* = u_j,$$

and

$$\frac{\partial f^*}{\partial x_j} = 0 \quad \text{if} \quad l_j < x_j^* < u_j$$

In the active set algorithm a sequence of unconstrained quadratic problems are solved to predict the correct sets J^* and I^*.

2 An active set algorithm

Our algorithm differs from the active set algorithm described in (Fletcher,1981) and (Gill, Murray and Wright, 1981), because beside upper and lower bounds, we also take into account the piecewise quadratic form of the minimized function.

We define two types of working sets. At the k–th iteration of the active set algorithm, I_k will be a working set of the function f. That set defines a quadratic function as follows:

$$f_{I_k}(x) = cx + (1/2)\|A_1 x - b_1\|^2 + (1/2)\sum_{i \in I_k}(a_i^{(2)}x - b_i^{(2)})^2 \tag{4}$$

The second working set defines those variables which are fixed at bounds.

$$J_k = \{j : x_j = l_j \text{ or } x_j = u_j\} \tag{5}$$

For given point x, it is also useful to define the following set of indices

$$J(x) = \{j : x_j = l_j \text{ and } \partial f(x)/\partial x_j \geq 0\} \cup \{j : x_j = u_j \text{ and } \partial f(x)/\partial x_j \leq 0\} \tag{6}$$

and

$$I(x) = \{i : a_i^{(2)}x - b_i^{(2)} > 0\} \tag{7}$$

For the last set the following relation holds

$$f(x) = f_{I(x)}(x) = cx + (1/2)\|A_1 x - b_1\|^2 + (1/2)\sum_{i \in I(x)}(a_i^{(2)}x - b_i^{(2)})^2$$

Additionally, the complements of the working sets will be defined as follows

$$\bar{I}_k = \{1, 2, \ldots, s\} \setminus I_k. \tag{8}$$

$$\bar{J}_k = \{1, 2, \ldots, n\} \setminus J_k. \tag{9}$$

Using the notation defined above, for given working sets I_k and J_k, the following minimization subproblem can be formulated

$$\min f_{I_k}(x) \tag{10}$$

$$x_j = \bar{x}_j \quad j \in J_k \tag{11}$$

where

$$\bar{x}_j = \begin{cases} l_j & \text{if fixed lower bound} \\ u_j & \text{if fixed upper bound} \end{cases} \tag{12}$$

The active set algorithm, in the form described below, solves a sequence of the subproblems. For given working sets I_k and J_k, we minimize the quadratic function f_{I_k} in respect to variables x_j which indices j belong to the set $\bar{J}_k$. This variable will be free. The variables which indices belong to the set J_k are fixed on their bounds. This is an unconstrained quadratic subproblem. Its solution defines a search direction. The step length is determined to provide feasibility. The piecewise quadratic form of the function f is also taken into account while the step length is computed.

2.1 Algorithm

0. (Initialization) For the given initial feasible point x^0 determine I_0, J_0 as follows

$$I_0 = I(x^0), \quad J_0 = J(x^0).$$

Set $k := 0$.

1. (Subproblem optimality test). If

$$\partial f_{I_k}(x^k)/\partial x_j = 0, \quad i \in \bar{I}_k$$

then minimum of the subproblen is found, go to step 2.

Otherwise do to step 4.

2. (Optimality test). If

$$I_k = I(x^k) \quad i \quad J_k = J(x^k)$$

then assume x^k as an optimal solution and stop. Otherwise, continue.

3. (Working sets reduction). From the sets of indices which are defined as the workings sets delete an index for which holds

$$\max\{\max_{j \in J^d} |\partial f_{I_k}(x^k)/\partial x_j|, \max_{i \in J^d} |a_i^{(2)} x - b_i^{(2)}|\}$$

$$J^d = J_k \setminus J(x^k) \quad I^d = I_k \setminus I(x^k)$$

4. (Search direction computing). Solve the unconstrained optimization subproblem (10)-(12) and let $\bar{x}^k$ be a minimizer. Set

$$p^k = \bar{x}^k - x^k$$

as the search direction.

5. (Step length computation). Find $\bar{\alpha}$ - an upper bound for the step length

$$\bar{\alpha} = \min\{\bar{\alpha}_1, \bar{\alpha}_2\}$$

a. Where $\bar{\alpha}_1$ is chosen in such way that $x^k + \bar{\alpha}_1 p^k$ remains feasible:

$$\alpha_1 = \min_{j \in K} (l_j - x_j^k)/p_j^k, \qquad K = \{j : j \in \bar{J}, p_j^k < 0\}$$

$$\alpha_2 = \min_{j \in K} (u_j - x_j^k)/p_j^k, \qquad K = \{j : j \in \bar{J}, p_j^k > 0\}$$

$$\bar{\alpha}_1 = \min\{\alpha_1, \alpha_2\}$$

b. Where $\bar{\alpha}_2$ is maximal value which provides

$$I(x^k + \bar{\alpha}_2 p^k) = I(x^k)$$

Thus

$$\bar{\alpha}_2 = \min_{i \in I_k}\{(b_i^{(2)} - a_i^{(2)} x^k)/a_i^{(2)} p^k\}$$

For found $\bar{\alpha}$ compute:

$$\alpha^k = \operatorname{argmin}_{\alpha} f(x^k + \alpha p^k) \quad \alpha \leq \bar{\alpha}$$

$$x^{k+1} = x^k + \alpha^k p^k$$

6. (Test for working set augmentation). If $\alpha^k < \bar{\alpha}$ then:

$$I_{k+1} = I_k \quad J_{k+1} = J_k$$

Set $k := k + 1$ and go to step 1.

7. (Working sets augmentation).

   a. If $\alpha^k = \bar{\alpha}_1$ and l is the index of the variables which bounded step length, then:

$$J_{k+1} = J_k \cup \{l\}, \quad I_{k+1} = I_k$$

   b. If $\alpha^k = \bar{\alpha}_2$ and r is the index of the rows in the matrix A_2 which bounded step length, then:

$$I_{k+1} = I_k \cup \{r\}, \quad J_{k+1} = J_k$$

Set $k := k + 1$ and go to step 1.

Remark: We note that working sets reduction in step 3 based on the observation that the Lagrange multiplier λ_j for the constraints:

$$l_i \leq x_j \leq u_j$$

are

$$\lambda_j = \begin{cases} \partial f_{I_k}(x^k)/\partial x_j & \text{if } x_j = l_j \\ -\partial f_{I_k}(x^k)/\partial x_j & \text{if } x_j = u_j \end{cases}$$

If optimality conditions are fulfilled then we have found the optimal point. If not, the objective function can be decreased by deleting corresponding bound or row of the matrix which defined function (4).

For the sake of simplicity we will drop the index k in the description of the working sets. Let A_2^I and b_2^I be a submatrix and a subvector composed of rows and coordinates corresponded to indices $i \in I$.

$$A^I = \begin{pmatrix} A_1 \\ A_2^I \end{pmatrix} \quad b_1 = \begin{pmatrix} b_1 \\ b_2^I \end{pmatrix}$$

Using the above notation, the problem (10)–(12) can be rewritten as follows

$$\min f_I(x) \tag{13}$$

$$f_I(x) = cx + (1/2)\|A^I x - b^I\|^2 \tag{14}$$

$$x_j = \bar{x}_j \quad j \in J \tag{15}$$

We divide the vector x into two vectors corresponding to the working set J and its complement:

x_J – vector free variables

x_J – vector fixed variables

We have

$$x = (x_J \quad x_J)$$

Then we divide the matrix A^I into two submatrices which rows correspond to the fix and free variables respectively.

$$A^I = (A^I_J \quad A^I_J)$$

So we have:

$$f_I(x) = c^J x^J + c^J x^J + (1/2)\|A^I_J x^J + A^I_J x^J - b^I\|^2 \tag{16}$$

Let us consider the problem of finding free variables x^J as a result of minimization (16) without constraints. We assume that the matrix A^I_J has full column rank. In this case the problem of minimizing function (16) has an unique solution. Such a situation takes place when the considered subproblem is defined for the proximal or for the regularized multiplier method. The minimum of the function (16) can be obtained by solving the following system of equations:

$$(A^I_J)^T A^I_J x^J = (A^I_J)^T (b^I - A^I_J x^J) - c^J \tag{17}$$

The classical approach to solving this problem is via the system of normal equations

$$\bar{B} x^J = \bar{b} \tag{18}$$

where $\bar{B}$ is the symmetric positive definite matrix in the form:

$$\bar{B} = (A^I_J)^T A^I_J \tag{19}$$

and

$$\bar{b} = (A^I_J)^T (b^I - A^I_J x^J) - c^J \tag{20}$$

In a discussion of methods which can be useful for solving the system (18), one should take into account such features as numerical stability of algorithms, density and the dimension of matrices (Golub and Van Loan, 1983) (Heath, 1984).

Equation (18) can be solved via the conjugate gradient algorithm or by the preconditioned conjugate gradient algorithm. Those methods can be especially useful for large and sparse problems, but unfortunately the algorithms converge slowly when the problem is ill-conditioned.

Another approach for solving the normal equation based on factorization of the matrix $\bar{B}$ using Cholesky's method:

$$\bar{B} = R^T R \tag{21}$$

where R is upper triangular, and then x^J is computed by solving the two triangular systems

$$R^T y = \bar{b} \tag{22}$$

$$R x^J = y \tag{23}$$

Despite many useful features of the normal equation method, the method with direct application of Cholesky's partition to the normal equations also has several drawbacks. We mention some of them

- Necessity of explicitly forming and processing $\bar{B}$ according to (19)
- The condition number of $\bar{B}$ is the square of the condition number of A^I_J.

3 Application the QR decomposition

To simplify the discussion we write (16) in the following form:

$$f_I(x) = c^J x^J + (1/2)\|A_J^I x - h_J^I\|^2 + g_J^I \tag{24}$$

where

$$h_J^I = b^I - A_J^I x^J$$
$$g_J^I = c^J x^J$$

In the orthogonal factorization approach a matrix Q is used to reduce A_J^I to the form

$$Q^T A_J^I = \begin{pmatrix} R_J^I \\ 0 \end{pmatrix} \quad Q^T h_J^I = \begin{pmatrix} p_1 \\ p_2^I \end{pmatrix}$$

where R_J^I is upper triangular. We have

$$f_I(x) = c^J x^J + (1/2)\|R_J^I x - p_1\|^2 + (1/2)\|p_2\|^2 + g_J^I$$

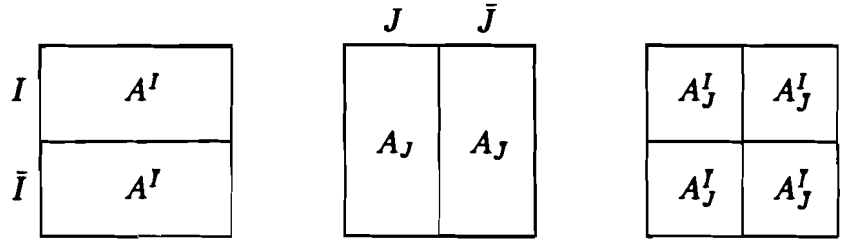
The application of the orthogonal matrix Q does not change L_2–norm and an advantage of such a transformation is that we do not need to save the matrix Q. It can be discarded after it has been applied to the vector h_J^I. Moreover, the matrix R_J^I is the same as the Cholesky's factor of $\bar{B}$ (19) apart from possible sign differences in some rows.

The above Q–R transformation can be carried out by using of Givens rotations which are very attractive for our case (see George and Heath,1980). In our implementation we do not store the orthogonal matrix Q and the obtained matrix R_J^I is used for solution (22)–(23), where the vector $\bar{b}$ is given by (20).

3.1 Update of Q–R decomposition

As we have shown in the description of the active set algorithm, the working sets were changed during sequential steps (compare steps 3, 7). Changes of working sets result in changes of the matrix A_J^I, but only one row or one column can be added or removed from that matrix at a time. This means that the matrix A_J^I which defined the Hessian of minimizing function (24) is changed. Consequently we should update Q–R factorization whenever an index is added to or deleted from the working set. Computing a new factorization ab initio would be much too expensive so we adopted numerically stable methods for updating the Q–R decomposition (e.g. see Golub and Van Loan, 1983, Lawson, Hanson,1974).

To simplify the description we split up the initial matrix A in a way which corresponds to the definition of the working sets (see the Figure below). The contents of matrix A_J^I changes along computation.



We now describe the way of updating in step 7 when an index is added to the working set.

1. If the following holds
$$I_{k+1} = I_k \cup \{r\}, \quad J_{k+1} = J_k$$
then the column r is deleted from A_J^I and it is added to the matrix A_J^I.

2. If the following holds
$$J_{k+1} = J_k \cup \{l\}, \quad I_{k+1} = I_k$$
then the row l is added to A_J^I and it is deleted from A_J^I.

Similarly, let us consider changes of the matrix A_J^I when the working sets are reduced in step 3.

1. If the following holds
$$J_{k+1} = J_k \setminus \{l\}, \quad I_{k+1} = I_k$$
then the column l is deleted from A_J^I and it is added to A_J^I.

2. If the following holds
$$I_{k+1} = I_k \setminus \{r\}, \quad J_{k+1} = J_k$$
then the row r is removed from A_J^I and it is added to A_J^I.

We have just seen that to modify Q–R decomposition of the matrix A_J^I the following cases should be considered:

1. Adding a column
2. Deleting a column
3. Adding a row
4. Deleting a row

In sequel we shortly describe the above four modifications of the Q–R factorization. Assume that we have the upper triangular matrix R_J^I which has been obtained after application of the Q–R decomposition to the matrix A_J^I.

3.2 Adding a column

Assume that the column a_l^l is to be added to the matrix A_J^I.

$$(A_J^I, a_l^l)$$

We want to obtain a new decomposition with the upper triangular matrix in the form:

$$\begin{pmatrix} R_J^I & u \\ 0 & \gamma \end{pmatrix}.$$

Where the column vector u is obtained by solving the triangular system of equations

$$(R_J^I)^T u = (A_J^I)^T a_l^l$$

and the scalar γ is calculated in the form

$$\gamma = (\|a_l^l\|^2 - \|u\|^2)^{1/2}$$

3.3 Deleting a column

Deleting the column l from the matrix A_J^I corresponds to deleting the column l from the matrix R_J^I. Note that the matrix H obtained from R_J^I after deleting l is an upper Hessenberg matrix. This matrix contains some of subdiagonals elements not equal zero. Clearly, the nonzero subdiagonal elements can be zeroed by sequence Givens rotations (Golub and Van Loan, 1983).

3.4 Adding a row

Suppose that we have the upper triangular matrix R_J^I and we wish to obtain an upper triangular of

$$\tilde{A} = \begin{pmatrix} a_r^J \\ A_J^I \end{pmatrix}$$

It corresponds to the following Hessenberg matrix

$$H = \begin{pmatrix} a_r^J \\ R_J^I \end{pmatrix}$$

After application a sequence of Gives rotation to the matrix H the nonzero subdiagonal elements can be zeroed.

3.5 Deleting a row

This type of modification of Q–R decomposition is possible in the case when the matrix after removing a row is positive define. Suppose that for an orthogonal matrix Q we have

$$QA_J^I = \begin{pmatrix} R_J^I \\ 0 \end{pmatrix}$$

Note that the matrix Q is not stored. For the deleting row a_r^J we wish to find an upper triangular matrix $\tilde{R}_J^I$, for which we have

$$(\tilde{R}_J^I)^T \tilde{R}_J^I = (R_J^I)^T R_J^I - a_r^J (a_r^J)^T \tag{25}$$

We should determine an orthogonal matrix U as the product of Givens rotations, that the following holds

$$U \begin{pmatrix} R_J^I \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{R}_{J_r}^I \\ a_r^J \end{pmatrix} \tag{26}$$

Note that due to $U^T U = I$ the equation (25) holds.

The matrix U is chosen in such a way, that

$$U \begin{pmatrix} u \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

where u and α are determined as the solution of the system

$$(R_J^I)^T u = a_r^J$$

and

$$\alpha = (1 - \|u\|^2)^{1/2}$$

If the Givens rotations which defines U are then used as in (26), the desired matrix $\tilde{R}_J^I$ will be found.

4 Final remarks

We have developed a stable numerical method for minimization of the piecewise quadratic function with lower and upper bounds. Such problems arise, for example, in application of the multiplier method to linear programming problems. The presented approach can be also useful for problems in which the matrices A_1 and A_2 are large and sparse. In those cases, the methods for symbolic generation of sparse structure for storing the factors R_J^I can be adopted in the similar way as in (Björck, 1988).

References

Björck Å, (1988). A direct method for sparse least squares problems with lower and upper bounds. Numerische Mathematik,Vol.54,19–32.

Flecher, R. (1981). Practical methods of optimization, vol II, Constrained optimization, Wiley, New York.

Gill P.E, W. Murray, M.H. Wright, (1981) Practical Optimization. Academic Press. London, New York.

George A. and M.T. Heath, (1980) Solution of Sparse Linear Least Squares Problem Using Givens Rotations. Linear Algebra and its Application Vol.34,69–83.

Golub, G.H. and C.F. Van Loan (1983). Matrix Computations, Johns Hopkins University Press, Baltimore, Maryland.

Heath, M.T. (1984). Numerical Methods for Large Sparse Linear Least Squares Problems. SIAM J. Sci. Stat. Comput., Vol. 5, No. 3.

Lawson, C.L and R.J. Hadson, (1974). Solving Least Squares Problems. Prentice–Hall. New Jersey.

Rckafellar,R.T, (1976). Augmented lagrangians and Applications of the Proximal Point Algorithm in Convex Programming. Mathematics of Operations Research, Vol. 1, No 3,97–116.

F L I P
Multiobjective Fuzzy Linear Programming Package

P. Czyżak , R. Słowiński
Technical University of Poznań
Institute of Computing Science
Poznań 1990, Poland

Abstract

FLIP (Fuzzy LInear Programming) is a package designed to help in analysis of multiobjective linear programming (MOLP) problems in an uncertain environment. The uncertainty of data is modeled by L-R type fuzzy numbers. They can appear in the objective functions as well as on the both sides of the constraints.

The input data to the FLIP package include the characteristics of the analyzed fuzzy MOLP problem, i.e., the number of criteria, constraints and decision variables, fuzzy cost coefficients for every objective and fuzzy coefficients of LHS and RHS for all constraints. The data loading is supported by a graphical presentation of fuzzy coefficients. The calculation is preceeded by a transformation of the fuzzy MOLP problem into a multiobjective linear fractional program. It is then solved with an interactive method using a linear programming procedure as the only optimiser. In every iteration, one gets a series of solutions that are presented very clearly in a graphical and numerical form.

In FLIP, interaction with the user takes place at two levels: first, when safety parameters have to be defined in the transformation phase, and second, when the associate deterministic problem is solved.

The package is written in TURBO-Pascal and can be used on microcomputers compatible with IBM-PC XT/AT with hard disc and a graphic card.

Part I
Methodological guide

1 Formulation of the problem

FLIP solves the following MOLP problem with fuzzy coefficients:

$$(P1) \qquad minimize \begin{bmatrix} \tilde{z}_1 = \tilde{\underline{c}}_1 \underline{x} \\ \vdots \\ \tilde{z}_k = \tilde{\underline{c}}_k \underline{x} \end{bmatrix}$$

$$
\begin{array}{rlll}
 & \underline{\tilde{a}}_i\underline{x} \leq \tilde{b}_i & i = 1,...,m_1 & (A) \\
s.t. & \underline{\tilde{a}}_i\underline{x} \geq \tilde{b}_i & i = m_1+1,...,m & (B) \\
 & \underline{x} \geq \underline{0} & &
\end{array}
$$

where $\underline{x}$ is a column vector of n decision variables, $\underline{\tilde{c}}_1$,...,$\underline{\tilde{c}}_k$ are row vectors of fuzzy cost coefficients corresponding to the objective functions $\tilde{z}_1$,...,$\tilde{z}_k$, $\underline{\tilde{a}}_i$ is the i-th row of the matrix of fuzzy coefficients, and $\tilde{b}_i$ is its corresponding fuzzy right-hand-side. The equality constraints are excluded from the above formulation since they can be obviously represented by pairs of inequality constraints. It is assumed, moreover, that for the particular objectives, the decision maker (DM) is in a position to define fuzzy aspiration levels, thought of as goals, denoted by $\tilde{g}_1,...,\tilde{g}_k$. All fuzzy coefficients are given as fuzzy numbers, i.e. normal and convex fuzzy subsets of the real line.

FLIP has been proposed to solve a water supply planning problem under uncertainty. Its presentation in [11] has been preceded by a brief survey of approaches to fuzzy mathematical programming proposed before 1984. The survey has shown that there was no method which would deal with a multiobjective linear programming problem with fuzzy coefficients in the objective functions and on the both sides of the constraints. Since then, a lot of new work has been done; a short updated review has been made in [12], together with some refinements of FLIP. More comprehensive surveys have been done recently by Dubois [6], Luhandjula [8], and Inuiguchi, Ichihashi and Tanaka [7]. Others, like Delgado, Verdegay and Vila [5], Rommelfanger [9], and Sakawa and Yano [10], presented their new methods on the background of existing ones.

The idea of FLIP relies on an observation that for a given $\underline{x}$, the main question to be answered consists in the comparison of the left- and right-hand-sides in objectives and constraints which are fuzzy numbers. Assuming an L-R representation of fuzzy coefficients, a comparison principle has been proposed ([11]), which allows a transformation of the fuzzy MOLP problem into a multiobjective linear fractional programming problem. The best compromise solution of the latter problem ensures the "best consistency" between the goals and the objective functions, and satisfies the constraints with a given "safety".

In the next section, we present the comparison principle of fuzzy numbers which is used in the transformation of the fuzzy MOLP problem into a deterministic mathematical program. The transformation is described in chapter 3, and in chapter 4, the way of solving the associate deterministic problem is outlined. In chapter 5, the steps of FLIP are summarized with a special emphasis on interactive steps.

2 Comparison of L-R fuzzy numbers

A convenient representation of a fuzzy number $\tilde{a}$ is a triple of parameters (a,α,β) of its membership function

$$
\mu_a(x) = \begin{cases} L((a-x)/\alpha) & \text{if } x < a \\ R((x-a)/\beta) & \text{if } x > a \end{cases}
$$

where a is an interval of the "most possible" values, α and β are nonnegative left and right "spreads" of $\tilde{a}$, respectively, and L, R are symmetric bell-shaped reference functions that are decreasing in $(-\infty,\infty)$ and L(0)=R(0)=1, L(1)=R(1)=0; $\tilde{a}$ is said to be an L-R fuzzy number. When the spreads are zero, then $\tilde{a}$ is a nonfuzzy (crisp) number equal to a.

Let $\tilde{a} = (a, \alpha, \beta)_{LR}$ and $\tilde{b} = (b, \gamma, \delta)_{L'R'}$ be fuzzy numbers. In order to evaluate the possibility for $\tilde{b}$ to be greater than $\tilde{a}$, we use two different indices: σ and π , called optimistic and pessimistic, respectively.

(a)**Optimistic index σ**

Let us suppose a situation presented in Fig.1. Optimistic index $\sigma\left(\tilde{b} > \tilde{a}\right)$ is defined as an ordinate of the intersection point of R' and L, i.e. the right slope of $\tilde{b}$ with the left slope of $\tilde{a}$:

$$\sigma(\tilde{b} > \tilde{a}) = R'((d-b)/\delta) = L((a-d)/\alpha) = \omega \tag{1}$$

where d is an abscissa of the intersection point. The smaller $\sigma(\tilde{b} > \tilde{a})$ is, the less true is the assertion that $\tilde{b}$ is greater than $\tilde{a}$.

Let us observe that (2.1) is equivalent to

$$b + \delta R'^{-1}(\omega) = a - \alpha L^{-1}(\omega) \tag{2}$$

and

$$\delta R'^{-1}(\omega) + \alpha L^{-1}(\omega) = a - b \tag{3}$$

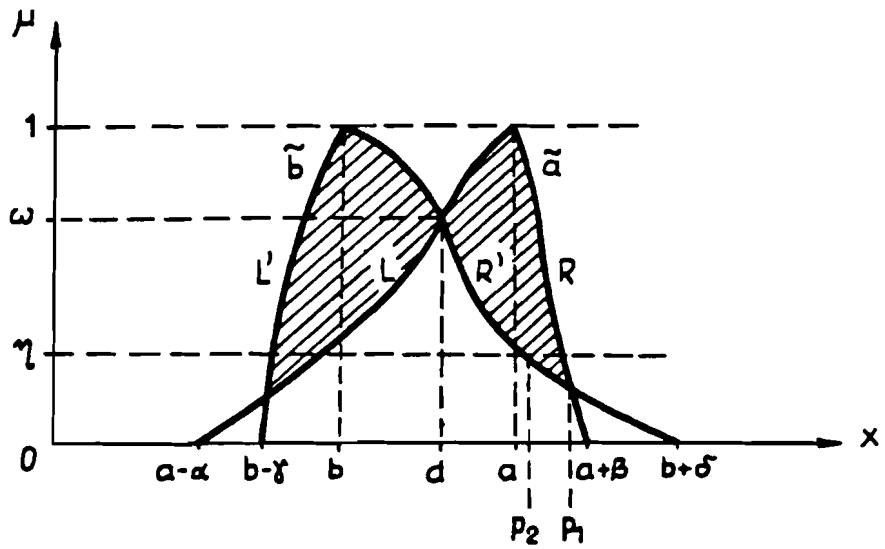


Fig.1. Comparison of fuzzy numbers $\tilde{a}$ and $\tilde{b}$

For $F(\omega) = (\delta R'^{-1}(\omega) + \alpha L^{-1}(\omega))^{-1}$, we have

$$F^{-1}(\omega) = a - b \tag{4}$$

which implies

$$\sigma(\tilde{b} > \tilde{a}) = \omega = F(a-b) \tag{5}$$

Similarly, $\sigma(\tilde{a} > \tilde{b})$ is defined as an ordinate of the intersection point of R and L', i.e. the right slope of $\tilde{a}$ with the left slope of $\tilde{b}$:

$$\sigma(\tilde{a} > \tilde{b}) = \psi = G(b-a) \tag{6}$$

where $G(\psi) = (\gamma L'^{-1}(\psi) + \beta R^{-1}(\psi))^{-1}$.

If L=R' and L'=R then (2.5) and (2.6) take the form

$$\sigma(\tilde{b} > \tilde{a}) = L((a-b)/(\alpha+\delta)) \tag{7}$$

$$\sigma(\tilde{a} > \tilde{b}) = R((b-a)/(\beta+\gamma)) \tag{8}$$

In order to say that $\tilde{b} > \tilde{a}$, we need both $\sigma(\tilde{a} > \tilde{b})$ and $\sigma(\tilde{b} > \tilde{a})$. If for instance $\sigma(\tilde{b} > \tilde{a}) \geq 1$, we know that either $\tilde{b}$ is greater than $\tilde{a}$, or both fuzzy numbers are too close to be separated. Then we may choose a threshold $0 \leq \tau \leq 1$ and admit that $\tilde{b}$ is greater than $\tilde{a}$ at level τ as soon as $\sigma(\tilde{a} > \tilde{b}) \leq \tau$. If $\min\{\sigma(\tilde{a} > \tilde{b}), \sigma(\tilde{b} > \tilde{a})\} \geq \tau$, we say that $\tilde{a}$ and $\tilde{b}$ are approximately equal.

In the case of weak inequality $\tilde{b} \geq \tilde{a}$, we only need $\sigma(\tilde{b} > \tilde{a})$. Indeed, for $\sigma(\tilde{b} > \tilde{a}) \geq 1$ the weak inequality is satisfied for any value of $\sigma(\tilde{a} > \tilde{b})$. Then we may choose a credibility constant $0 \leq \tau \leq 1$ and admit that $\tilde{b} \geq \tilde{a}$ at credibility level τ as soon as $\sigma(\tilde{b} > \tilde{a}) \geq \tau$.

The comparison index σ can be seen as optimistic because even for a high value of $\sigma(\tilde{b} > \tilde{a})$, the possibility for $x \in \mathcal{R}$, fuzzily restricted to belong to $\tilde{a}$, to be greater than $y \in \mathcal{R}$, fuzzily restricted to belong to $\tilde{b}$, may be quite big. As a measure of this possibility one can consider the hatched area marked in Fig.1. Thus, in order to make the comparison more credible, we should use σ conjointly with the pessimistic index π.

(b) **Pessimistic index π**

The pessimistic index follows from the comparison of the right slopes of $\tilde{a}$ and $\tilde{b}$ at some level $0 \leq \eta \leq 1$:

$$\pi(\tilde{b} >_\eta \tilde{a}) = b + \delta R'^{-1}(\eta) - a - \beta R^{-1}(\eta) = p_2 - p_1 \tag{9}$$

$$\pi(\tilde{a} >_\eta \tilde{b}) = a + \beta R^{-1}(\eta) - b - \delta R'^{-1}(\eta) = p_1 - p_2 \tag{10}$$

For $\eta = 0$ we have

$$\pi(\tilde{b} >_0 \tilde{a}) = b + \delta - a - \beta \tag{11}$$

$$\pi(\tilde{a} >_0 \tilde{b}) = a + \beta - b - \delta \tag{12}$$

Thus, a joint use of $\sigma(\tilde{b} > \tilde{a})$ and $\pi(\tilde{b} >_\eta \tilde{a})$ consists in the comparison of both slopes of a which is supposed to be smaller with the right slope of $\tilde{b}$ which is supposed to be greater.

Now, we can admit that $\tilde{b} \geq \tilde{a}$ at credibility levels τ and η if and only if

$$\sigma(\tilde{b} > \tilde{a}) \geq \tau \tag{13}$$

and

$$\pi(\tilde{b} >_\eta \tilde{a}) \geq \theta \tag{14}$$

where $\tau, \eta \in$ [0,1] and $\theta \in (-\infty, \infty)$. $\theta \geq 0$ means that for any pair (x,y) such that $x \geq a$, $y \geq b$ and $0 \leq \mu_a(x) = \mu_b(y) \leq \eta$, inequality y≥x is true. A negative value of θ makes possible that inequality y≥x is not true.

The constants τ,η and θ are called "safety parameters" because they are responsible for the safety of the assertion that $\tilde{b}$ is greater than $\tilde{a}$. Instead of safety, we could speak of course about risk which is a complementary term. Let us remark that using τ,η and θ one can control the surface of the hatched area marked in Fig.1 which corresponds to the risk of the above assertion.

3 The associate deterministic problem

Conditions (2.13) and (2.14) can be used directly to transform the fuzzy MOLP problem into an associate deterministic one. Let us separately analyze the constraints and the objective functions of the fuzzy MOLP problem.

Let us suppose that fuzzy coefficients of the constraints are given as L-R fuzzy numbers:

$$\tilde{a} = (a, \alpha, \beta)_{LR}, \tilde{b} = (b, \gamma, \delta)_{RL}, \qquad \text{i=1,...m.}$$

For the sake of clarity, we assume that the reference functions of all fuzzy coefficients are of two kinds only: L and R. It can be seen from the preceding considerations that this assumption is not necessary for the calculation of σ and π (cf. [5], [6] and [9], [10]).

It should be specified that all arithmetic operations on fuzzy numbers taking place in (P1) are extended operations in the sense of Zadeh's principle. For any $\underline{x} \geq \underline{0}$, the left side of the i-th constraint can be summarized to the fuzzy number:

$$\tilde{\underline{a}}_i \underline{x} = (\underline{a}_i \underline{x}, \underline{\alpha}_i \underline{x}, \underline{\beta}_i \underline{x})_{LR}, \qquad \text{i=1,...,m.}$$

It is easy to verify that for i=1,...,m_1

$$\sigma(\tilde{b}_i > \tilde{\underline{a}}_i \underline{x}) = L((\underline{a}_i \underline{x} - b_i)/(\underline{\alpha}_i \underline{x} + \delta_i))$$

$$\pi(\tilde{b}_i >_{\eta_i} \tilde{\underline{a}}_i \underline{x}) = b_i + \delta_i L^{-1}(\eta_i) - \underline{a}_i \underline{x} - \underline{\beta}_i \underline{x} R^{-1}(\eta_i)$$

and for $i = m_1 + 1, ..., m$

$$\sigma(\tilde{\underline{a}}_i \underline{x} > \tilde{b}_i) = R((b_i - \underline{a}_i \underline{x})/(\underline{\beta}_i \underline{x} + \gamma_i))$$

$$\pi(\tilde{\underline{a}}_i \underline{x} >_{\eta_i} \tilde{b}_i) = \underline{a}_i \underline{x} + \underline{\beta}_i \underline{x} R^{-1}(\eta_i) - b_i - \delta_i L^{-1}(\eta_i)$$

For given values of safety parameters τ_i, η_i and θ_i , i=1,...,m, the constraints (A) and (B) of (P1) may be transformed to:

$$L((\underline{a}_i \underline{x} - b_i)/(\underline{\alpha}_i \underline{x} + \delta_i)) \geq \tau_i \qquad \text{i=1,...,}m_1 \qquad (15)$$

$$R((b_i - \underline{a}_i \underline{x})/(\underline{\beta}_i \underline{x} + \gamma_i)) \geq \tau_i \qquad \text{i=}m_1\text{+1,...,m} \qquad (16)$$

$$b_i + \delta_i L^{-1}(\eta_i) - \underline{a}_i \underline{x} - \underline{\beta}_i \underline{x} R^{-1}(\eta_i) \geq \theta_i \qquad \text{i=1,...,}m_1 \qquad (17)$$

$$\underline{a}_i \underline{x} + \underline{\beta}_i \underline{x} R^{-1}(\eta_i) - b_i - \delta_i L^{-1}(\eta_i) \geq \theta_i \qquad \text{i=}m_1\text{+1,...,m} \qquad (18)$$

Basing on the property of strict monotonicity of reference functions, one can transform (3.1) and (3.2) into the following equivalent conditions:

$$\underline{a}_i \underline{x} - b_i \leq L^{-1}(\tau_i)(\underline{\alpha}_i \underline{x} + \delta_i) \qquad \text{i=1,...,}m_1 \qquad (19)$$

$$b_i - \underline{a}_i \underline{x} \leq R^{-1}(\tau_i)(\underline{\beta}_i \underline{x} + \gamma_i) \qquad \text{i=}m_1\text{+1,...,m} \qquad (20)$$

In the case of fuzzy objective functions, σ can be used to evaluate the degree of consistency between fuzzy objectives and fuzzy goals. Let the fuzzy cost coefficients and fuzzy goals be:

$$\tilde{\underline{c}}_l = (\underline{c}_l, \underline{\varepsilon}_l, \underline{\kappa}_l)_{LR}, \tilde{g}_l = (g_l, 0, \nu_l)_{LL}, \qquad \text{l=1,...,k}$$

where the left spread of $\tilde{g}_l$, l=1,...,k, is equal to zero because it is immaterial for the evaluation of consistency between goals and objectives. Here again, the equality of the reference functions is not a necessary assumption. For any $\underline{x}$, the components of the l-th objective function can be summarized with the extension principle to the flat fuzzy number:

$$\tilde{\underline{c}}_l\underline{x} = (\underline{c}_l\underline{x}, \underline{\epsilon}_l\underline{x}, \underline{\kappa}_l\underline{x})_{LR}, \qquad l=1,...k.$$

On the basis of (2.7), we can write for l=1,...,k

$$\sigma(\tilde{g}_l > \tilde{\underline{c}}_l\underline{x}) = L((\underline{c}_l\underline{x} - g_l)/(\underline{\epsilon}_l\underline{x} + \nu_l)) \tag{21}$$

In order to ensure the best consistency between goals and objectives, $\sigma(\tilde{g}_l > \tilde{\underline{c}}_l\underline{x})$, l=1,...,k should be maximized. In consequence, we arrive to the following deterministic mathematical programming problem equivalent to problem (P1):

$$(P2) \qquad maximize \begin{bmatrix} f_1(\underline{x}) = L((\underline{c}_1\underline{x} - g_1)/(\underline{\epsilon}_1\underline{x} + \nu_1)) \\ \vdots \\ f_k(\underline{x}) = L((\underline{c}_k\underline{x} - g_k)/(\underline{\epsilon}_k\underline{x} + \nu_k)) \end{bmatrix}$$

$$\begin{array}{lll} & \underline{a}_i\underline{x} - b_i \le L^{-1}(\tau_i)(\underline{\alpha}_i\underline{x} + \delta_i) & i=1,...,m_1 \quad (A1) \\ & b_i - \underline{a}_i\underline{x} \le R^{-1}(\tau_i)(\underline{\beta}_i\underline{x} + \gamma_i) & i=m_1+1,...,m \quad (B1) \\ s.t. & b_i + \delta_i L^{-1}(\eta_i) - \underline{a}_i\underline{x} - \underline{\beta}_i\underline{x}R^{-1}(\eta_i) \ge \theta_i & i=1,...,m_1 \quad (A2) \\ & \underline{a}_i\underline{x} + \underline{\beta}_i\underline{x}R^{-1}(\eta_i) - b_i - \delta_i L^{-1}(\eta_i) \ge \theta_i & i=m_1+1,...,m \quad (B2) \\ & \underline{x} \ge \underline{0} & (C) \end{array}$$

If reference function L of fuzzy cost coefficients is linear, i.e. L(y)=1-y, than problem (P2) becomes a multiobjective linear fractional programming (MLFP) problem:

$$(P3) \qquad maximize \begin{bmatrix} f_1(\underline{x}) = 1 - (\underline{c}_1\underline{x} - g_1)/(\underline{\epsilon}_1\underline{x} + \nu_1) \\ \vdots \\ f_k(\underline{x}) = 1 - (\underline{c}_k\underline{x} - g_k)/(\underline{\epsilon}_k\underline{x} + \nu_k) \end{bmatrix}$$

s.t. (A1), (B1), (A2), (B2), (C).

In order to be sure that the denominators of $f_1(\underline{x}), ..., f_k(\underline{x})$ are greater than zero for any feasible $\underline{x}$, we may admit that $\nu_l > 0$, l=1,...,k, which is quite natural.

4 Interactive steps

FLIP can be summarized in the following steps:

1. Formulation of problem (P1) and definition of fuzzy coefficients.
2. Definition of fuzzy aspiration levels $\tilde{g}_l$ (l=1,...,k) on objectives.
3. Definition of safety parameters τ_i, η_i and θ_i (i=1,...,m) for fuzzy constraints.
4. Formulation of the associate multiobjective deterministic problem (P2) or (P3).
5. Application of an interactive method for solving the associate deterministic problem formulated in step 4.

6. If a best compromise solution has been found then stop, otherwise return to step 3 for revision of safety parameters. Retraction to steps 1 and 2 for redefinition of fuzzy coefficients and/or aspiration levels is also possible.

Our experience indicates that an interaction with the DM in searching of the best compromise solution is very beneficial for the final decision. So, in step 5, in order to solve the associate multiobjective deterministic problem, we propose to use an interactive method (cf.[16]). If the associate deterministic problem has the form of the MLFP problem (P3), a very convenient interactive procedure is that of Choo and Atkins (1980).

Let us recall its general idea informally.

In order to find a starting efficient point of the MLFP problem, the objective functions $f_1, ..., f_k$ are aggregated by the Chebyshev norm which is the maximum weighted deviation from the ideal point $\underline{u}^*$:

$$(P4) \qquad minimize \quad \max_l\{\phi_l(u_l^* - f_l(\underline{x}))\}$$

s.t. (A1), (B1), (A2), (B2), (C)

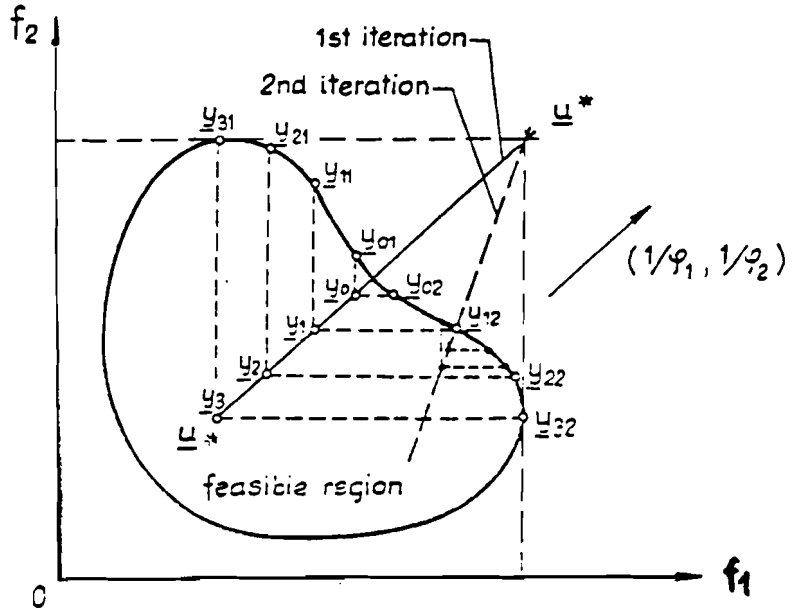


Fig.2. An idea of the interactive procedure by Choo and Atkins (1980)

where $\underline{\Phi} = (\Phi_1, ..., \Phi_k)$ is a weighting vector defining a "direction" of the Chebyshev norm, i.e. a line passing through an ideal and a nadir point, $\underline{u}^*$ and $\underline{u}_*$, respectively (cf. Fig.2).

The Chebyshev norm minimization chooses the "corner" closest to $\underline{u}^*$ and still in contact with the feasible region. This final point of contact ensures weak efficiency. In the case of the MLFP, (P4) can be transformed to a linear program with a single parameter. One can thus use any convenient univariate search method over this parameter to find point $\underline{y}_0$ as close to efficient as we like. The "closeness" here is not critical and even a rough approximation will be quite sufficient. This point is taken as starting point for the interactive part of the algorithm. This part crucially involves the DM. The search direction from $\underline{u}^*$ to $\underline{y}_0$ is then extended to include several more points $\underline{y}_1, \underline{y}_2, ..., \underline{y}_p$. Next,

taking each criterion in turn, say f_1 first, f_1 is maximized subject to all other criteria being at least equal to their values at $\underline{y}_0$, then $\underline{y}_1$, all the way to $\underline{y}_p$. This gives a sequence of (weakly) efficient points $\underline{y}_{01}, \underline{y}_{11}, ..., \underline{y}_{p1}$ for criterion 1 and $\underline{y}_{20}, \underline{y}_{21}$, etc. for criterion 2, and so on for all the criteria. The minimization of $f_l(\underline{x})$ for l=1,...,k is of a linear fractional subject to linear constraints, so the Charnes and Cooper transformation into a linear program can be used. Thus, at each step a single-objective linear programming problem has to be solved. The calculated sample of the efficient border is then presented to the DM who is asked to select the one that best fits his needs. The selected point becomes the new starting point and the procedure is repeated, but now the search space is focused on a part of the efficient border of most interest. The interactive process ceases when the most satisfactory efficient point is reached.

An important advantage of the above algorithm is that the only optimization procedure to be used is a linear programming one. Moreover, it has a simple scheme and allows retractions to the points which have been found uninteresting in previous iterations.

In a microcomputer implementation of FLIP, the efficient points proposed to the DM are presented both numerically, in terms of $\underline{x}$ and middle values of $\tilde{z}_l(\underline{x})$, l=1,...,k, and graphically, in terms of mutual positions of fuzzy numbers corresponding to objectives and aspiration levels on the one hand, and to left- and right-hand-sides of constraints, on the other hand (cf. [3], [4],[14]). In this way, the DM gets quite a complete idea about the quality of each proposed solution. The quality is evaluated taking into account:

- scores of fuzzy objectives in relation to the goals,
- dispersion of values of the fuzzy objectives due to uncertainty,
- safety of the solution or, using a complementary term, the risk of violation of the constraints (cf. the hatched areas in Fig.1).

So, the definition of the best compromise involves an analysis of the compromise among criteria and an evaluation of the safety of the corresponding solution.

It steems from the above considerations that the DM intervenes in two steps of FLIP. First, when fixing the safety parameters (step 3), and then in the course of the guided generation and evaluation of the efficient points of the associate deterministic problem (step 5). Thus, the interaction with the DM takes place at two levels. As to the first one (step 3), it is worth noting that there are two practical ways of controlling the safety of solutions using parameters τ_i , η_i and θ_i :

(a) fix $\eta_i = 0$, i=1,...,m, and control the optimistic safety with τ_i , and the pessimistic safety with θ_i , i=1,...,m, or

(b) fix $\theta_i = 0$, i=1,...,m, and control the optimistic safety with τ_i , and the pessimistic safety with η_i , i=1,...,m.

The safety parameters are defined taking into account their interval of variation and the knowledge acquired in preceding iterations about the dependency between safety and the quality of the compromise among criteria.

If way (a) is chosen, an (approximate) interval of variation of θ_i at level $\eta_i = 0$, i=1,...,m, can be calculated in the following way. Let the interval be denoted by $[\theta_i^L, \theta_i^U]$, and the set of feasible solutions by D.

It is clear that

$$\theta_i^U = \gamma_i + \delta_i \qquad \text{for i=1,...m}_1 \tag{22}$$

and

$$\theta_i^U = \max_{x \in D} \{\underline{\alpha}_i x + \underline{\beta}_i x\} \qquad \text{for i=m}_1\text{+1,...,m} \tag{23}$$

which corresponds to a perfect safety of solutions. The lower bound of the interval is defined as:

$$\theta_i^L = \min_{x \in D} \{b_i + \delta_i - \underline{\alpha}_i x - \underline{\beta}_i x\} \qquad \text{for i=1,...m}_1 \tag{24}$$

and

$$\theta_i^L = \min_{x \in D} \{\underline{\alpha}_i x + \underline{\beta}_i x - b_i - \delta_i\} \qquad \text{for i=m}_1\text{+1,...,m} \tag{25}$$

In practice, an exact calculation of $[\theta_i^L, \theta_i^U]$, i=1,...,m, may lead to too large intervals with very low utility. The following procedure permits "killing two birds with one stone". One substitutes all fuzzy coefficients of problem (P1) by their middle values and calculate the pay-off table of the multiobjective deterministic problem obtained from (P1). It needs solving k single-objective linear programming problems. In this way, a set S of k solutions is obtained. The result can be used in two ways. First, one can substitute D by S in formulas (4.2),(4.3),(4.4). Then $[\theta_i^L, \theta_i^U]$, i=1,...,m, become approximate intervals having often a greater practical utility than exact ones. Second, the individual optima from the diagonal of the pay-off table may serve as suggestions for middle values of fuzzy goals. The columns of the pay-off table may also give an idea about the right spreads of the goals. This option is useful when the DM has no his own definition of the aspiration levels.

To conclude, let us remark that since its appearance, FLIP has been successfully applied to several real-world problems from the field of water supply systems ([15], [12]) and agriculture ([2], [4], [3]).

Part II
User's manual

5 Executive summary

The FLIP package supports the following general functions:

- the definition and edition of a source model in the form of a multiobjective linear programming problem with fuzzy coefficients; in this phase, particular attention is paid to modeling of fuzzy coefficients and their graphical representation.
- interactive analysis of the problem, with user-friendly graphical and numerical representation of generated solutions, and various facilities for comparison of those solutions in the process of searching for the best compromise.

The FLIP package is recorded on one diskette in a compiled code. After installing it in the user's directory, it can be activated by the command FLIP $< CR >$.

6 User's reference manual

6.1 Installing and activating the program

There are two main versions of FLIP package. One version requires a math coprocessor (8087,80287,80387) while another one does not. Each version is recorded on one diskette that should be installed on an IBM-PC XT/AT or a compatible computer preferably with a hard disc and with Hercules or color graphic card (CGA,EGA,VGA). The diskette contains the compiled code of FLIP with a test example.

While it is possible to use the program from floppy disc it is preferable to install it on a hard disc.

If we run the program from a floppy disc we must remember that on the distribution disc there is no room for big problems and for saving partial or final results. If you need more free space you can use another disc for data (change disc before the problem is created). System starts with command FLIP $< CR >$.

To install FLIP package on the hard disc make the following steps:

a. create a new user directory (e.g. md FLIP);

b. copy the contents of a distribution disc to this directory;

c. run FLIP program by the command: FLIP $< CR >$.

A moment after giving a starting command for FLIP package, the invitation screen is displayed (see fig.3). Pressing any key we pass to main menu of the FLIP program.

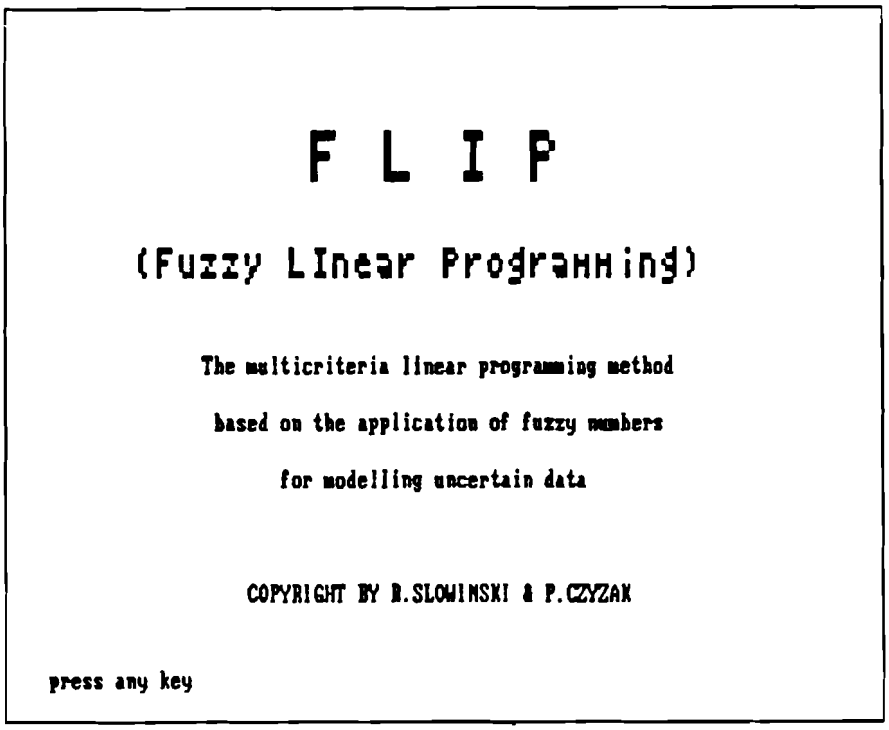


Fig.3 Invitation screen.

6.2 Main menu

Main menu (fig.4) provides the following functions:

Exit FLIP to leave the program and return to the operating system;

Data Input to create a new problem and save it into a disc file;

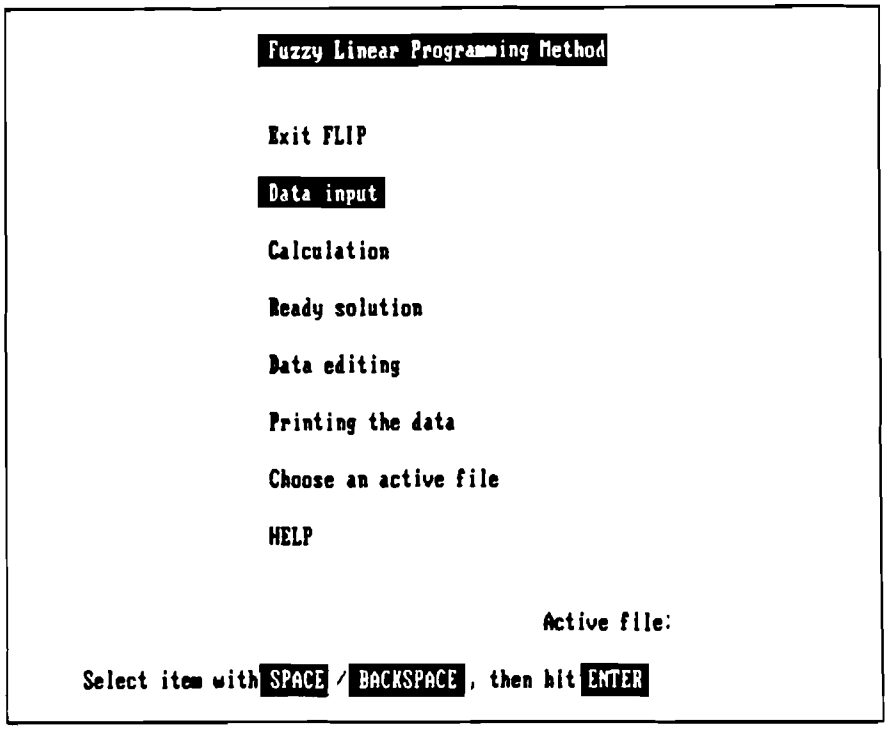


Fig.4 Main menu of the FLIP program.

Calculations after activating this function, the fuzzy problem is transformed into a deterministic one and an interactive procedure starts to search for the best compromise solution;

Retrieving Previous Solutions one can retrieve from disc files some series of previously got solutions;

Data Edit with this function we can review and/or change both source model and definition of fuzzy aspiration levels;

Printing the Data this function allows to get a copy of data file on a printer;

Choose an Active File we can choose from directory a data file for further consideration;

Help offers the information about FLIP method and how to make use of the FLIP program.

Selection of any of these functions is made by moving a highlighted window to the required function and acceptance with $< CR >$ key.

Moving of the highlighted window can be performed in one of three ways:

- using arrow keys;
- pressing space bar or backspace;
- pressing appropriate number keys e.g. 0 for "Exit FLIP", 1 - for "Data Input", 2 for "Calculation" and so on.

6.3 Data Input

This function allows to create a new disc file containing a new problem formulation. At the beginning, we enter the name of data file we create. Then, we define sizes of a new problem, that are: number of criteria, number of decision variables and number of constraints. We accept this values pressing function key - F10.

Now we pass to the definition of criterion number one. First, we choose a type of cost coefficients we want to use in this criterion. There are three alternatives:

real numbers if we want to have nonfuzzy coefficients;

linear fuzzy numbers if we want to define simple fuzzy numbers, giving only the middle value and left and right ranges of fuzzy number (see fig.5a);

piecewise linear fuzzy numbers similarily to linear fuzzy numbers we have to define the middle value, left and right ranges and additionally to construct left and right reference functions consisting of maximum three linear pieces each (fig.5b).

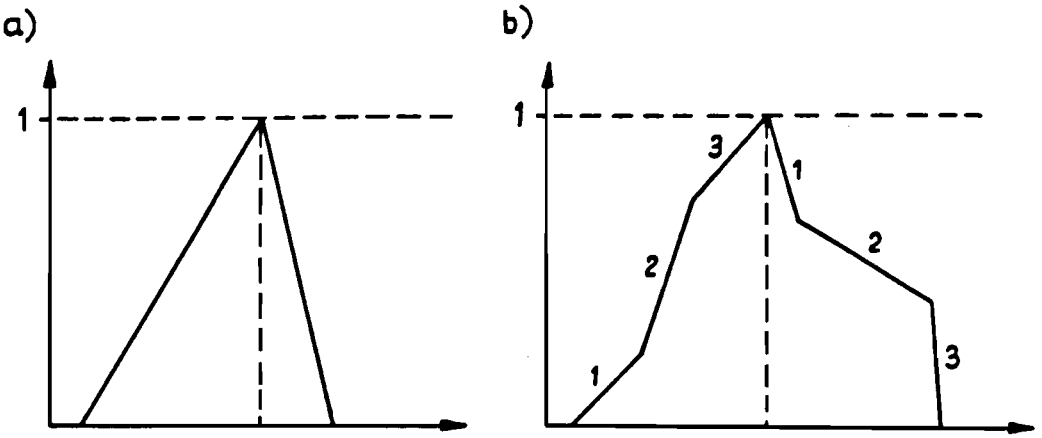


Fig.5 Definition of a fuzzy number.

In the next step, we declare if we want to maximize or to minimize the considered criterion.

When choosing the type of coefficient and maximization or minimization, we move a highlighted window using Space or Backspace keys and then we accept our choice with function key F10.

Before explaining in detail how to built two types of fuzzy numbers, we must make some important remarks.

In the definition of the fuzzy number (chapter 2), we spoke about left and right "spreads" of the fuzzy number, the left "spread" means a distance between the middle value and the left range of the fuzzy number and the right "spread" means a distance

between the middle value and the right range of the fuzzy number. In our implementation of FLIP, we don't use the values that express the "distance", but we give directly the real values of the left and right ranges of the fuzzy number. It means that we give values of abscissae of the appropriate points.

Next, we start defining of successive coefficients. At this moment, on the monitor screen we have:

- general form of the considered criterion with a highlighted coefficient which can be introduced or changed now;
- two windows containing an information about the highlighted coefficient: in the left window, numerical representation of the coefficient is displayed and in the right window, the graphical representation, i.e. the shape of the membership function.

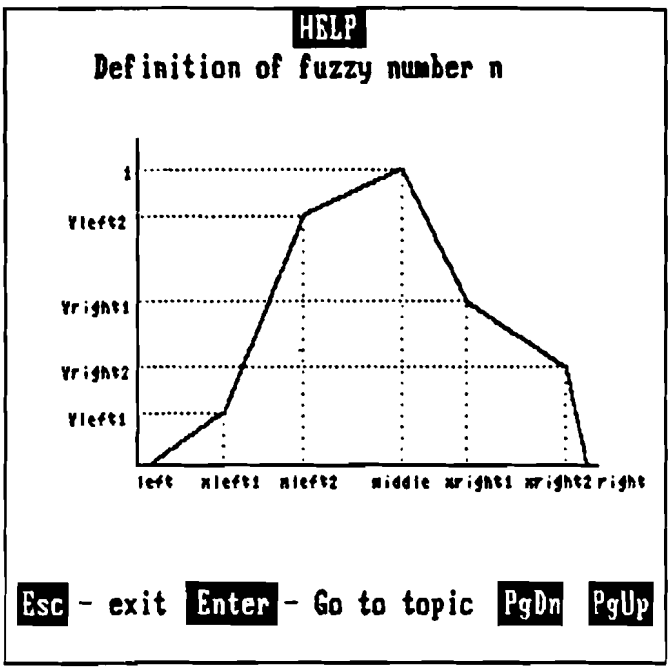


Fig.6 Definition of piecwise linear fuzzy number (Help).

The menu is displayed in the right bottom corner of the screen, offering the following functions:

number $< ENTER >$ data input;
if we define coefficient of type:

- real number, we type a value of the coefficient and press ENTER;
- linear fuzzy number, we type first the middle value of the coefficient, i.e. the "most possible" and press ENTER; then, we type values of left an right ranges of the fuzzy number; We must remember that:
  left range $\leq$ middle value $\leq$ right range.
- piecewise linear fuzzy number (Fig.6), we first type the middle value, left and right ranges such that again:
  left range $\leq$ middle value $\leq$ right range.
  If the left range $<$ the middle value, we define the left reference function in the following steps:

a. We are asked about the value of Xleft1. This is the abscissa of the point ending the first linear piece and beginning the second. If we press ENTER without any number, we resign from construction of the left reference function, otherwise, we type value for XLeft1 such that:
left range $\leq$ XLeft1 $\leq$ middle value
and then, we define YLeft1 *in* [0,1], that is value of the ordinate of the first breakpoint.

b. In the same way, we define the second breakpoint [XLeft2,Yleft2] ending the second linear piece and beginning the third such that:
XLeft1 $\leq$ XLeft2 $\leq$ middle value and YLeft1 $\leq$ YLeft2 $\leq$ 1.
If we press "ENTER" without any number, the left function will consist of two linear pieces only.

Next, if the middle value < right range, we start definition of the right reference function. We have to proceed it in the following steps:

a. We are asked about the value of XRight1. If we answer by "ENTER", we end the definition of the fuzzy number, otherwise, we define the first breakpoint $[XRight1, YRight1]$ such that:
middle value $\leq$ XRight1 $\leq$ right range
and YRight1 *in* [0,1].

b. We define the second breakpoint [XRight2,YRight2] of the right reference function. If we don't type any value for XRight2 and press "ENTER", we have the second linear piece between points [XRight1,YRight1] and [right range,0].

next coefficient pressing right arrow key we can move to the next coefficient (right from presently highlighted coefficient), if the last coefficient is presently pointed we moved back to the first coefficient of the criterion or to the RHS coefficient of the constraints;

previous coefficient pressing left arrow key we can move to the previous coefficient (left from presently highlighted coefficient); if we are at the first coefficient, then we go back to the last coefficient of the criterion or to the RHS coefficient of the constraint;

Choose coefficient this function allows to jump to a selected coefficient; after pressing "C" we are asked about the number of the coefficient we want to consider; this function can be very useful for problems with a great number of decision variables and a low density of the matrix coefficients.

F1 (help) in particular, we can see a general definition of a fuzzy number; it can be helpful to understand how to define fuzzy numbers;

F10 next criterion or

F10 next constraint with this function we end definition of the presently displayed criterion or constraint, and we go to the next criterion or constraint; if it was already the last constraint, we pass to the definition of fuzzy aspiration levels.

The procedure described above is just the same for criteria and constraints, with an obvious exception that for constraints we don't have maximization or minimization.

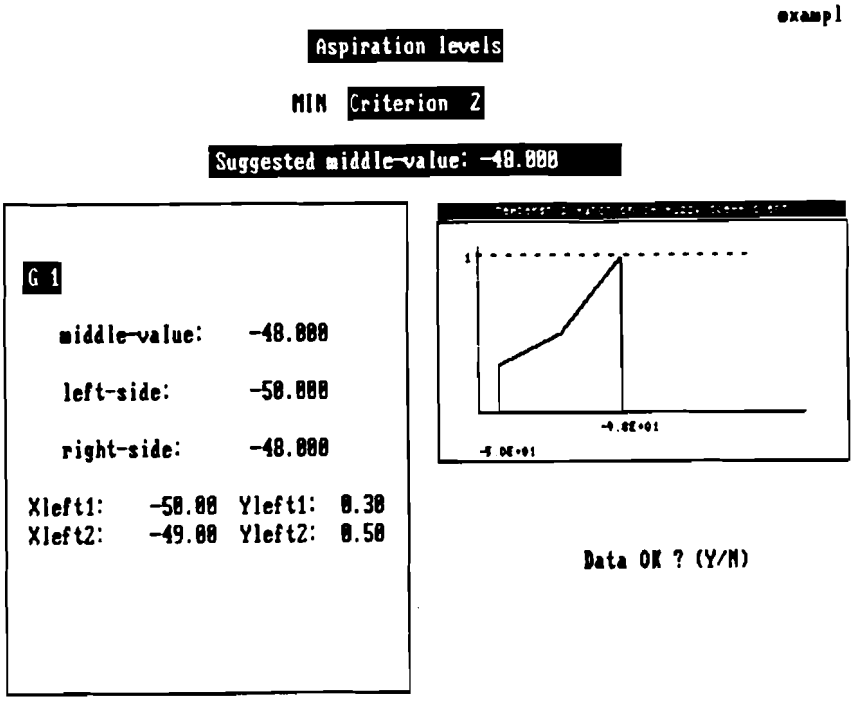


Fig.7 Definition of an aspiration level.

"Input Data" procedure ends with definition of a fuzzy aspiration level for every criterion. To support the decision maker the individual optima are calculated for every criterion under assumption, that fuzzy coefficients becomes crisp ones, equal to middle values of the corresponding fuzzy numbers. The calculated optimum is displayed on the screen as a suggestion. The decision maker chooses the type of an aspiration level (real number or fuzzy number - with linear or piecewise linear membership function) and then defines the value of the aspiration level (fig.7).

6.4 Data edit

This function gives a possibility of reviewing and changing a problem created using function "Data Input".

If an active file has been already choosen (with function "Choose an Active File" or other functions, e.g. "Data Input", "Calculations"), then, the active file will be edited, otherwise, the directory with data files will be displayed and we have to choose a data file we want to edit.

Then, an edit menu is displayed on the screen, which consists of ten functions (fig.8). At the bottom of the screen the sizes of the analyzed problem are given.

Now we have at our disposal the following menu:

Return to main menu With this function we can go back to the main menu of the FLIP program. If we have performed any edit function on an active file we are asked if we want to save the problem. If we don't save the changes they are lost and the source active file remains with no change.

Write the current problem to disc storage This function allows, at any moment, to save the currently edited problem with all changes we have made up to now. It is of course possible to change the name of the data file storing the changed problem and; if we do so, the source data file remains unchanged.

Add more criteria to the current problem Using this function, we can add new criterion to the current problem. The procedure of defining a new criterion is just the same as in "Data Input" function. So we define consequently: type of coefficients, maximization or minimization and nonzero coefficients. After accepting definition of the new criterion by pressing F10 we are asked if we want to define next criterion; if not, we return to edit menu.

Add more constraints to the current problem With this function we can add new constraints to the current problem. The way of entering new constraints is the same as for criteria.

Add more variables to the current problem It allows to increase the number of decision variables in the current problem. First, we are asked "How many variables do you want to add ?". If we answer "0", we return to edit menu, otherwise, the new variables are added to the problem and we can define coefficients corresponding to

the new variables. New variables are added as the last variables.

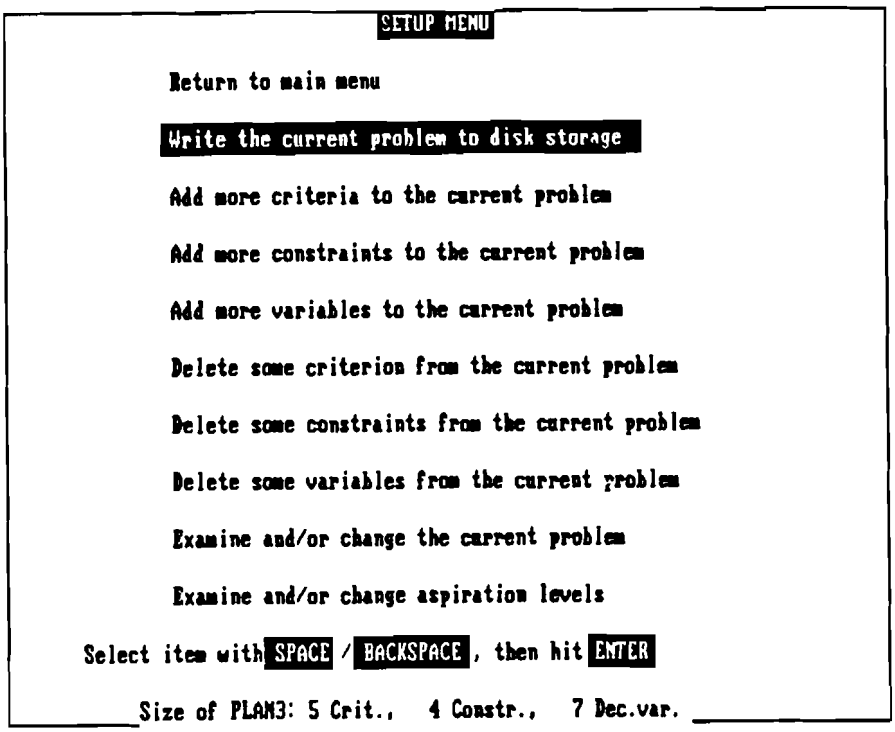


Fig.8 Edit menu.

Delete some criterion from the current problem With this function we can remove some criterion from the current problem. If we choose this function, we examine all criterion and point the criteria which we want to delete.

Delete some constraints from the current problem It allows to delete any constraint in a similar way to the previous function;

Delete some variables from current problem We are asked about the number of variable we want to delete. If we answer "0" then no variable is deleted.

Examine and/or change the current problem It is very useful function that allows reviewing and changing the current problem. We have at our disposal the same menu as when entering the data (see chapt.2). Our reviewing/changing procedure starts from the first criterion. We can move along the coefficients using left and right arrow keys or with function "Choose coefficient". We can change any coefficient. Pressing F10 we skip to the next criterion or constraint.

Examine and/or change aspiration levels This function gives the possibility of changing or reviewing aspiration levels. Numerical and graphical presentation of the current state of aspiration levels starts from the first criterion. The DM is asked, if he wants to change the value of the aspiration level. If answer is "No" then procedure goes to the next aspiration level of the next criterion, otherwise, the following procedure is realized:

a. individual optimization problem is solved for the considered criterion with all fuzzy coefficients fixed on their middle value and a suggestion about possible value of the aspiration level is displayed,

b. the DM choose the type of the aspiration level: real number or fuzzy number (linear or piecewise linear membership function),

c. the DM types a new value of the aspiration level and program goes to the next criterion.

6.5 Calculation

Calculation procedure we can divide into three parts:

- transformation of the fuzzy problem into an associate deterministic fractional linear programming problem;
- searching for a starting point of an interactive procedure;
- interactive search for the best compromise solution.

If active file isn't defined till now, the calculation procedure starts from choosing a data file from a directory.

6.5.1 Transformation of the fuzzy problem

Transformation of the fuzzy problem into an associate deterministic fractional linear programming problem is based on conception of comparison of fuzzy numbers proposed by Słowiński [11].

At the beginning, we have to define safety parameters: τ_i, η_i and θ_i (i=1,...,m) for fuzzy constraints. In our implementation of FLIP, in order to support the DM, standard values are proposed for safety parameters: τ_i =0.6, η_i =0, and θ_i equal to the middle value of an approximated range of variation for i=1,...,m.

The DM is asked if he accept this proposal or not. If not these values can be changed in two ways:

- simultaneously for all fuzzy constraints;
- independently for each fuzzy constraint.

After fixing the final value for safety parameters the fuzzy problem is transformed as it

was shown in chapt.3.

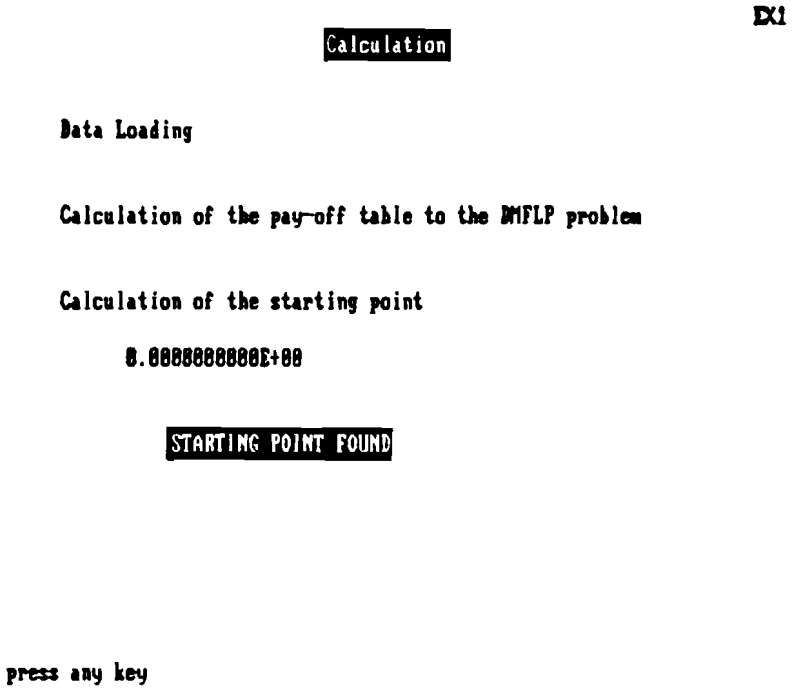


Fig.9 Search for a starting point.

6.5.2 Searching for the starting point

The second step of calculation procedure consist in searching for a starting point of the interactive method by Choo and Atkin's. First, utopia and nadir points are calculated, (the following information is displayed on the screen: "Pay-Off table is calculated") and then the procedure looks for a point lying in the proximity of an efficient border on the line connecting utopia and nadir points. This search consist in solving a series of LP problems and it can take some time, particularly, for big problems. During this calculation information about the distance between current solution and the efficient border is displayed.

It may happen that certain LP problems are contradictory (ERROR 2). In this situation, we should try to change safety parameters or, eventually, reformulate the analyzed problem. When the starting point will be found, an appropriate information will be displayed on the screen, together with the order "press any key" (fig.9). After pressing a key we pass to the most important, interactive phase of searching for the best compromise solution.

6.5.3 Interactive phase

This phase starts with a question to the DM how many compromise solution he wants to get. At this moment possible numbers of solutions appear on the screen; that are, obviously, multiples of the number of criteria. The DM choose the number of compromise solution by moving a highlighted window (using arrow keys) and accepting the choice by ENTER. Then, we can see, that single objective optimizations are performed and, finally, we get required number of efficient solutions (points) to be analyzed.

So, we can pass to the stage of comparison of compromise solutions and searching for the best one. The obtained solutions are presented both graphically and numerically. We have on the screen four windows with graphical representation of fuzzy criterion, constraints and values of decision variables (Fig.10).

In this moment, the DM has at his disposal the following menu:

arrow keys change active window (highlighted) - functions F1 to F4 can be performed in the active window;

F1 next display;

F2 previous display;

F3 setting display in the active window; allows to see selected criterion or constraint or variables for a given solution (point);

F4 enlarge/reduce the active window;

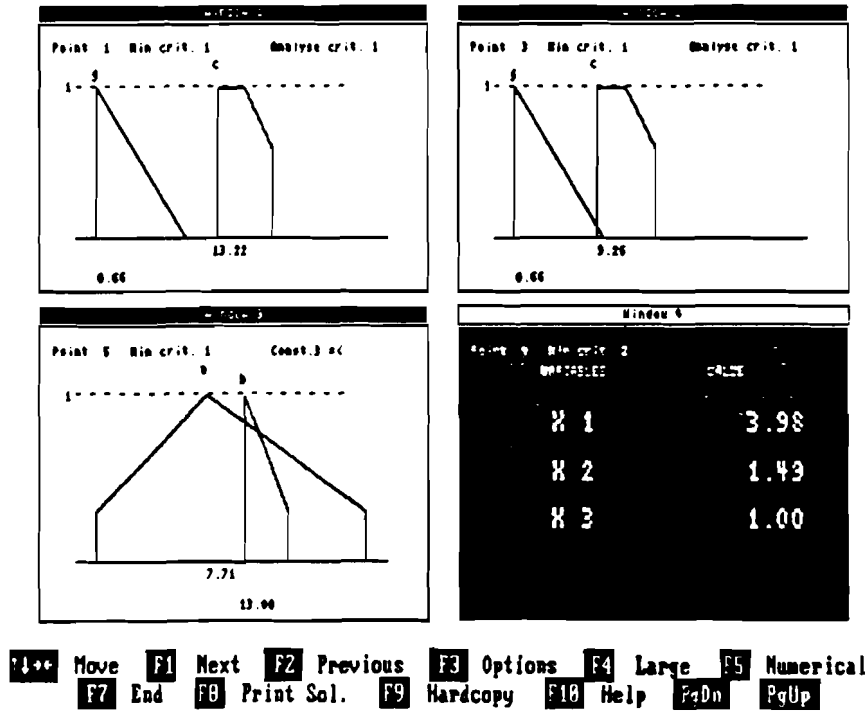


Fig.10 Comparison of compromise solutions.

F5 numerical display of middle values of criteria; it gives to the DM the first impression about the generated compromise solutions (fig.11);

F7 gives the options:

C ontinue to restart review of the generated set of compromise solutions.

N ew starting point to point the most interesting solution at the current stage which becomes a new starting point for the next iteration. In the next iteration, a new set of compromise solutions is generated from the neghbourhood of the starting point.

P revious solution to come back to review of solutions generated in the previous iteration.

S ave solution to save the currently generated set of compromise solution in a disc file. Using function "Retrieving previous solutions" from Main Menu, we can restore these solutions and continue their review.

E nd of survey to return to Main Menu.

Middle values of criteria

Points	1	2	3	4	5	6	7	8	9
Crit.1	13.2	13.0	10.8	15.4	8.5	17.8	6.1	[illegible]	3.7
Crit.2	5.1	5.2	5.8	4.6	6.4	3.9	7.0	3.3	7.7

Points	10	11	12
Crit.1	22.6	1.3	24.9
Crit.2	2.7	8.3	2.0

Fig.11 Middle values of compromise solutions.

F8 print selected solution we can get on a printer numerical information - values of criteria and decision variables - for selected solution.

F9 screen copy on the printer this function gives a hardcopy with a present contents of the monitor screen.

6.6 Retrieving previous solutions

This function displays previously saved solutions for a selected problem. We can examine a chosen set of solutions and, possibly, start generation of a new set of solutions. So, we pass to the interactive phase (described above) where a number of options is offered.

6.7 Printing a data

If no file is active we have to choose an active file. Then the DM is asked about the state of a printer. If the printer is ready then, after pressing any key, we get a copy of source data file on a printer.

7 User's training manual

7.1 General form of solving problem

FLIP package allows solving the MOLP problem with fuzzy coefficients which can be written in the following general form:

$$minimize \begin{bmatrix} \tilde{z}_1 = \underline{\tilde{c}}_1 \underline{x} \\ \vdots \\ \tilde{z}_k = \underline{\tilde{c}}_k \underline{x} \end{bmatrix}$$

$$s.t. \quad \begin{array}{rcll} \underline{\tilde{a}}_i \underline{x} & \leq & \tilde{b}_i & i = 1, ..., m_1 \\ \underline{\tilde{a}}_i \underline{x} & \geq & \tilde{b}_i & i = m_1 + 1, ..., m \\ \underline{x} & \geq & \underline{0} & \end{array}$$

where $\underline{\tilde{c}}_l$ (l=1,...,k) is a row vector of fuzzy cost coefficients of objective l, $\underline{\tilde{a}}_i$ is the i-th row of the matrix of fuzzy left-hand-side coefficients and $\tilde{b}_i$ is a right-hand-side coefficient of the i-th constraint (i=1,...,m).

To complete the above problem formulation, the decision maker must be able to define fuzzy aspiration levels thought of as goals, denoted by $\tilde{g}_1$ to $\tilde{g}_k$.

All coefficients, denoted with ~, and aspiration levels can be defined in one of three following ways:

- as real numbers;
- as linear fuzzy numbers - defined by three real numbers: middle value, left and right range;
- as piecewise linear fuzzy numbers - defined by maximum 11 real numbers:middle value, left and right range and, eventually, $[Xleft1, Yleft1]$, $[Xleft2, Yleft2]$, $[Xright1, Yright1]$, $[Xright2, Yright2]$.

7.2 Illustrative example

Let us consider the following simple MOLP problem with fuzzy coefficients:

$$minimize \begin{bmatrix} \tilde{z}_1 = \tilde{c}_{11} x_1 + \tilde{c}_{12} x_2 \\ \tilde{z}_2 = \tilde{c}_{21} x_1 + \tilde{c}_{22} x_2 \end{bmatrix}$$

$$s.t. \quad \begin{array}{rcrcl} -2x_1 & + & 2x_2 & \leq & 12 \\ 3x_1 & + & x_2 & \leq & 25 \\ x_1 & & & \leq & 7 \\ \tilde{a}_{41} x_1 & + & \tilde{a}_{42} x_2 & \leq & \tilde{b}_4 \end{array}$$

$$x_1, x_2 \geq 0$$

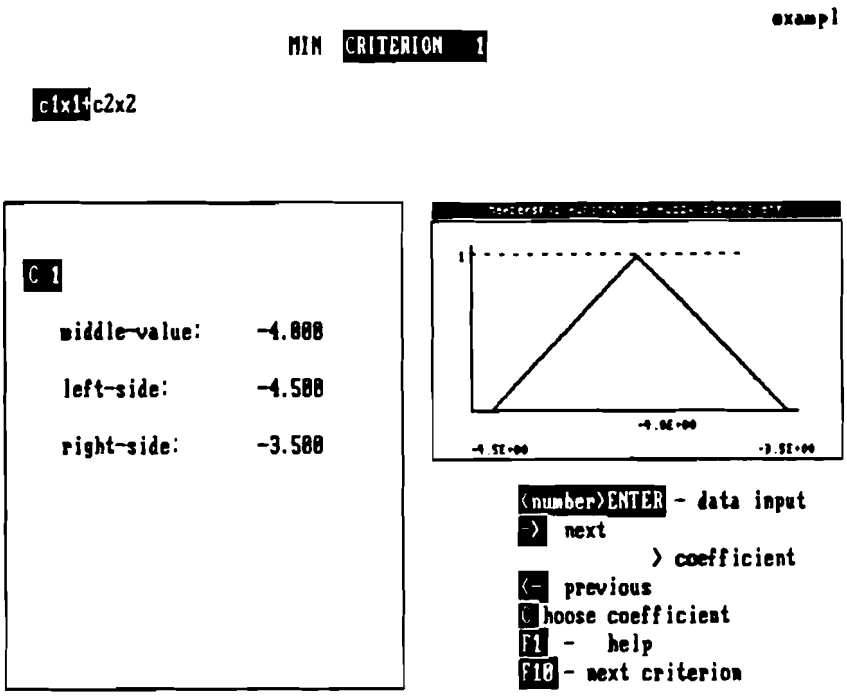


Fig.12 Definition of coefficient $\tilde{c}_{11}$.

where $\tilde{c}_{11}$=(-4,-4.5,-3.5)$_{LR}$, $\tilde{c}_{12}$=(1,1,1.5)$_{LR}$, $\tilde{c}_{21}$=(2,2,2.5)$_{LR}$, $\tilde{c}_{22}$=(-6,-6.5,-5.5)$_{LR}$, $\tilde{a}_{41}$=(2,1.5,2)$_{LR}$, $\tilde{a}_{42}$=(3,2.8,3.5)$_{LR}$, $\tilde{b}_4$=(33,32,36)$_{LR}$ and function L, R are linear.

The above problem is introduced to the program using the "Data Input" function. In criteria no.1 and 2 and in constraint no.4, we use fuzzy numbers with their linear form (Fig.12).

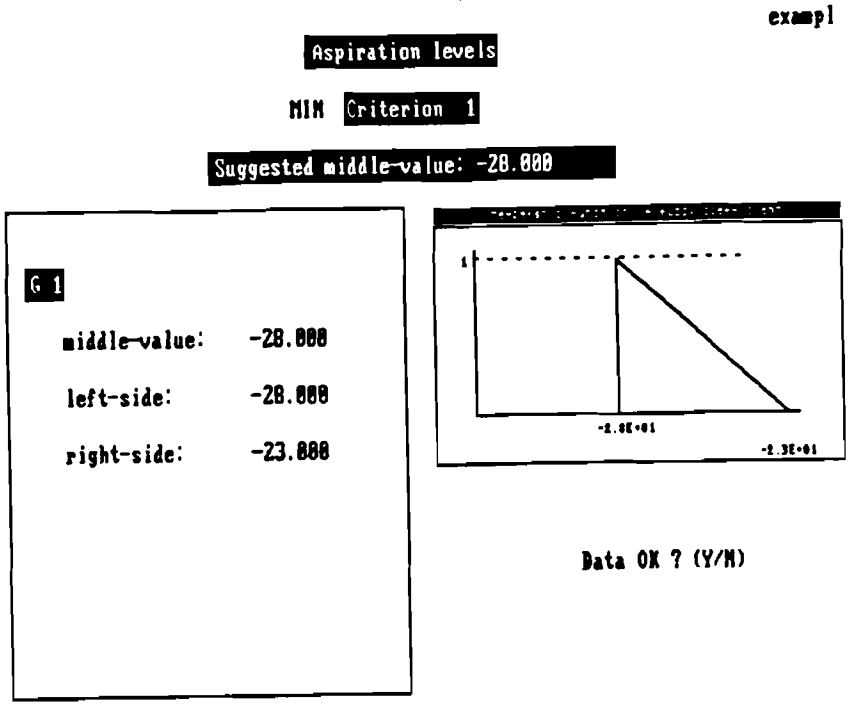


Fig.13 Definition of aspiration level $\tilde{g}_1$.

Then aspiration levels are defined. On the monitor screen the suggestion is displayed: '-28' for criterion no.1 and '-48' for criterion no.2. The decision maker takes as aspiration levels the following fuzzy numbers (fig.13):

$$\text{for } z_1 \text{ - } \tilde{g}_1 = (\text{-}28,\text{-}28,\text{-}23)_{LR}$$
$$\text{for } z_2 \text{ - } \tilde{g}_2 = (\text{-}48,\text{-}48,\text{-}18)_{LR}$$

At this moment the data input is completed and the new problem is saved into a disc file. After returning to the Main Menu of FLIP we choose the function "Calculation".

At the beginning we define safety coefficients that allow to transform fuzzy problem into deterministic one. The program makes a proposal: τ=0.6, η=0 and θ=0.85. The value proposed for θ is a middle value of approximated range of variability for θ_4 that is [-1.5,3]. The decision maker accepts this values without any changes.

Next, the fuzzy problem is transformed into a deterministic problem according to the rules explained in part I. Then an appropriate pay-off table is calculated and the search for a starting point begins. When it is completed, we can pass to the interactive part of looking for the best compromise solution. Let l_j denote the l-th solution obtained in the j-th iteration and S, the number of compromise solution required.

Let's take S=8. In the first iteration, we have got a series of 8 compromise solution for which middle values of criteria are presented in Tab.1.

TABLE 1.

	1_1	2_1	3_1	4_1	5_1	6_1	7_1	8_1
z_1	-18.2	-18	-23.2	-11.7	-25.9	-6.3	-28	-1.8
z_2	-26.5	-27.1	-12	-34.6	1.5	-41	14.3	-46.4

When comparing them, we come to conclusion that the most interesting solutions are solutions 1_1 , 2_1 and 4_1, because they are not far from aspiration levels.

Fig.14 shows a mutual position of particular criteria and their aspiration levels for solution 4_1 - upper windows - and for solution 2_1 -lower windows. Let's suppose that we want to improve the value of criterion 1 (left windows). We choose solution 2_1 as a new starting point for the second iteration.

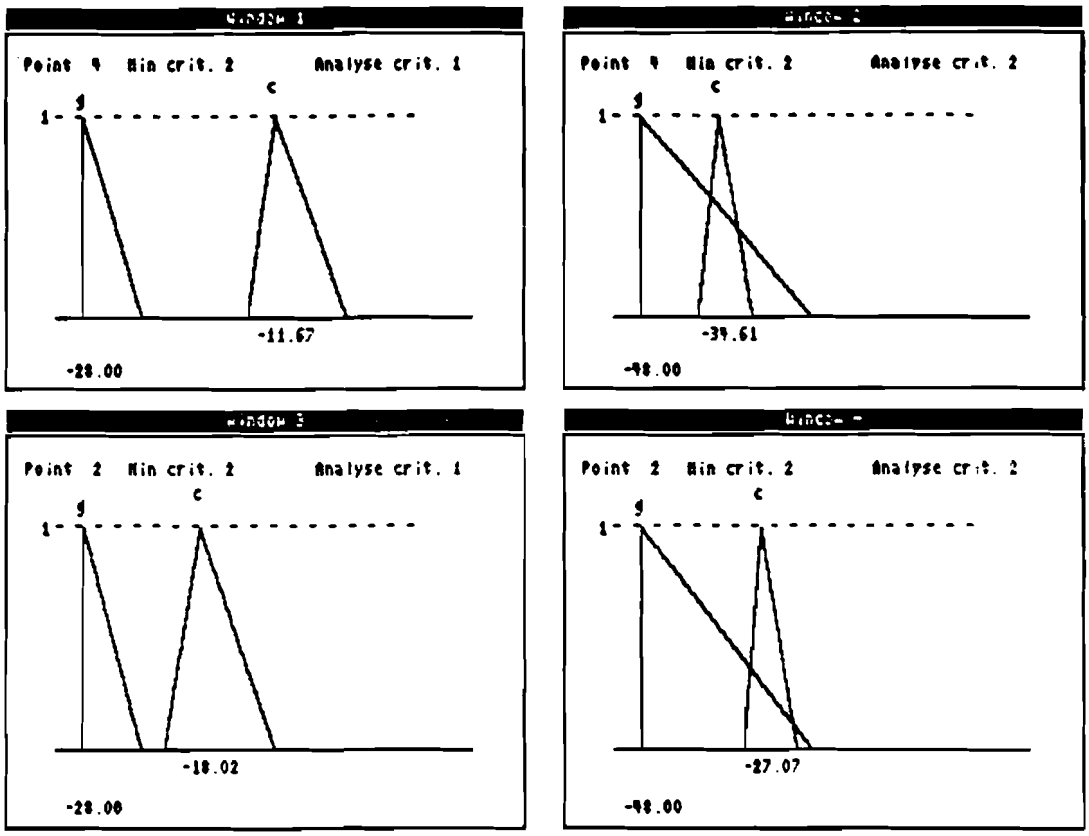


Fig.14 Values of criteria for solutions 4_1 and 2_1 .

For S=10 we obtain solutions listed in Tab. 2.

TABLE 2.

	1_2	2_2	3_2	4_2	5_2	6_2	7_2	8_2
z_1	-18	-18	-19.9	-15.5	-21.8	-13.1	-23.6	-10.9
z_2	-27.1	-27.1	-21.6	-30.1	-16.3	-32.9	-11.2	-35.5

	9_2	10_2
z_1	-24.6	-8.8
z_2	-6.2	-38

As we see, the most interesting solutions from the viewpoint of criterion 1 are solutions 3_2, 5_2, 7_2 and 9_2. Among those four solutions only solution 3_2 has a satisfactory value of

criterion 2 (see Fig.15, upper windows).

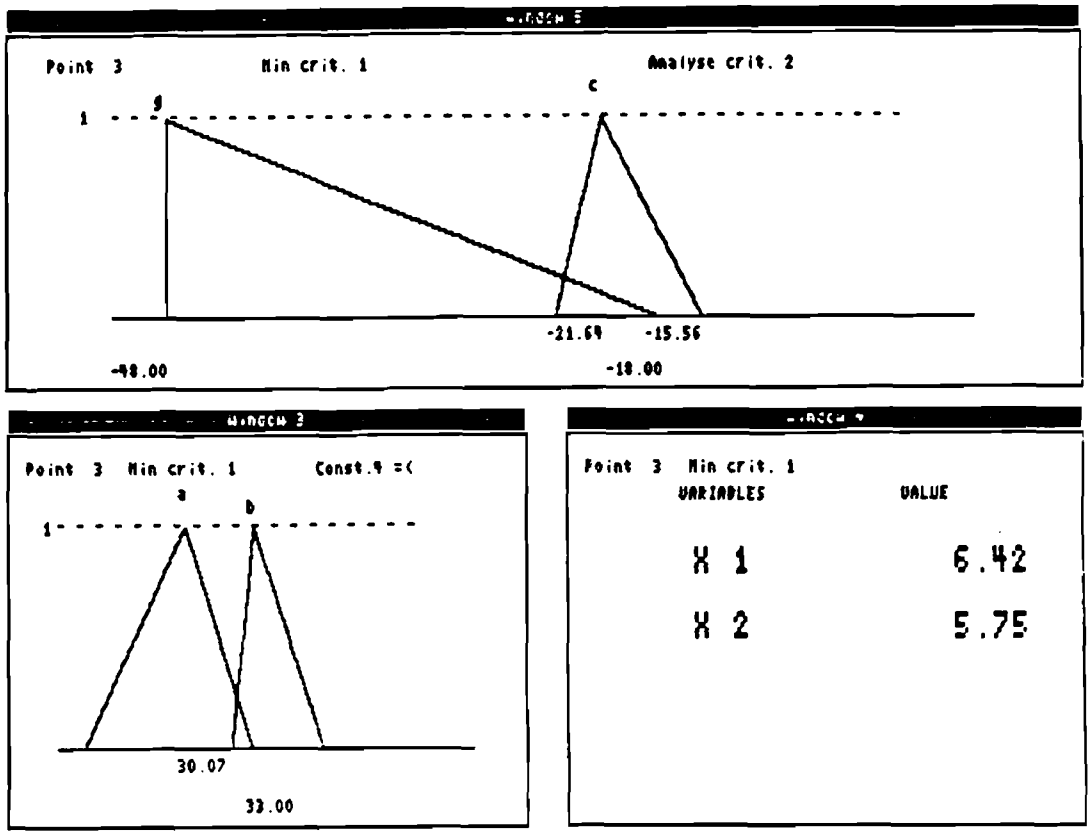


Fig.15 Solution 3_2

To accept this solution as the best compromise solution, we must analyze the state of fuzzy constraint for this solution. By the state we mean information about a risk of violation of the constraint.

In the lower, left window in fig.15, we present the state of constraint no. 4. As we can seen, there is no risk of violation of this constraint.

So finally, we choose solution 3_2 as the best compromise solution with following values of decision variables (Fig.15):

$$x = 6.42 \text{ and } x = 5.75.$$

Part III
Solution of a real farm structure optimization problem

Let us consider a real-life problem of searching for the best structure of a typical Polish private farm. The considered farm has 20 hectares (ha) of arable land and 4 ha of permanent grassland. The farmer possesses 6 sows and 4 cows. We take into account 26 activities which can be divided into 5 groups:

- plant production for sale: winter wheat, winter barley, triticale, spring wheat, spring barley, rape, pea, potato, sugar beet;
- plant production for fodders consumed in the farm: winter barley, spring barley, triticale, spring wheat, potatoes, fodder beet, lucerne, clover, corn;
- permanent grassland cultivation;
- purchase of fertilizers: phosphorus, nitrogen, potassium;
- purchase of a nutritive fodder;
- manpower hire: in the spring, summer and autumnal period.

All these activities correspond to decision variables $x_1, x_2, ..., x_{26}$ defining their dimension, e.g. number of hectares of winter barley for sale, number of kilograms of phosphorus to be bought, number of hours of manpower hire in the spring period.

We take into consideration the following constraints:

- balance of arable land;
- 2 balances of crop succession: for spring crops and for rape;
- 3 constraints on the area of group of plants: for crops, for winter crops and for root crops;
- 3 balances of manpower: for spring, summer and autumnal periods of extend manpower demand;
- 3 balances of artificial fertilizers: for phosphorus, nitrogen and potassium;
- 4 balances of fodders: for green fodder, for potatoes, for fodder beet and for nutritive fodder;
- balance of permanent grassland.

The matrix of constraints is presented in Tab.1. Definition of fuzzy coefficients is given in Tab.2.

TABLE 1. Set of constraints.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
1	1	1	1	1	1	1	1	1	1	1	1	1	1
			-1	-1				1		-1		-1	
				1	-1					1			
1	1	1	1	1					1	1	1	1	1
								1	1				
1	1	1							1		1		
$\tilde{a}_1^7$	$\tilde{a}_2^7$	$\tilde{a}_3^7$	$\tilde{a}_4^7$	$\tilde{a}_5^7$	$\tilde{a}_6^7$	$\tilde{a}_7^7$	$\tilde{a}_8^7$	$\tilde{a}_9^7$	$\tilde{a}_{10}^7$	$\tilde{a}_{11}^7$	$\tilde{a}_{12}^7$	$\tilde{a}_{13}^7$	$\tilde{a}_{14}^7$
$\tilde{a}_1^8$	$\tilde{a}_2^8$	$\tilde{a}_3^8$	$\tilde{a}_4^8$	$\tilde{a}_5^8$	$\tilde{a}_6^8$	$\tilde{a}_7^8$	$\tilde{a}_8^8$	$\tilde{a}_9^8$	$\tilde{a}_{10}^8$	$\tilde{a}_{11}^8$	$\tilde{a}_{12}^8$	$\tilde{a}_{13}^8$	$\tilde{a}_{14}^8$
$\tilde{a}_1^9$	$\tilde{a}_2^9$	$\tilde{a}_3^9$	$\tilde{a}_4^9$	$\tilde{a}_5^9$	$\tilde{a}_6^9$	$\tilde{a}_7^9$	$\tilde{a}_8^9$	$\tilde{a}_9^9$	$\tilde{a}_{10}^9$	$\tilde{a}_{11}^9$	$\tilde{a}_{12}^9$	$\tilde{a}_{13}^9$	$\tilde{a}_{14}^9$
160	140	120	120	120	180	20	120	160	10	15	35	15	160
120	120	120	100	100	120	100	120	120	120	100	120	100	120
100	100	90	80	80	180	80	120	120	90	80	90	80	120
								300					
													250
									45	40	43	40	

x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	sg.	RHS
1	1	1	1									=	20
1			1									¿=	0
												¿=	0
												¡=	14
1												¡=	5
												¡=	10
$\tilde{a}_{15}^7$	$\tilde{a}_{16}^7$	$\tilde{a}_{17}^7$	$\tilde{a}_{18}^7$	$\tilde{a}_{19}^7$					-1			¡=	$\tilde{b}_7$
$\tilde{a}_{15}^8$	$\tilde{a}_{16}^8$	$\tilde{a}_{17}^8$	$\tilde{a}_{18}^8$	$\tilde{a}_{19}^8$						-1		¡=	$\tilde{b}_8$
$\tilde{a}_{15}^9$	$\tilde{a}_{16}^9$	$\tilde{a}_{17}^9$	$\tilde{a}_{18}^9$	$\tilde{a}_{19}^9$							-1	¡=	$\tilde{b}_9$
160	30	30	200	200	-1							¡=	0
120	120	120	140	120		-1						¡=	0
120	100	100	120	100			-1					¡=	0
	400	350	550	350								¿=	3000
												¿=	480
900												¿=	450
								1				¿=	360
				1								=	4

TABLE 2. Fuzzy coefficients of constraints.

$\tilde{a}_j^i = (a_j^i, \alpha_j^i, \beta_j^i)$									
j	a_j^7	α_j^7	β_j^7	a_j^8	α_j^8	β_j^8	a_j^9	α_j^9	β_j^9
1	15	1	1	40	2	2	35	2	3
2	15	1	1	40	2	2	10	1	1
3	15	1	1	40	2	2	35	2	3
4	25	2	3	40	2	2	15	1	1
5	25	2	3	40	2	2	15	1	1
6	30	3	3	90	4	5	10	1	1
7	30	2	3	75	3	3	20	2	3
8	110	3	5	15	1	1	120	4	5
9	80	4	4	20	2	2	160	5	6
10	30	2	3	40	2	3	10	1	1
11	25	2	2	40	2	3	15	2	2
12	15	1	1	40	2	3	35	2	3
13	25	2	3	40	2	3	15	1	1
14	110	3	5	15	1	1	120	4	4
15	80	2	3	20	2	2	160	5	6
16	30	2	2	34	2	3	10	1	1
17	30	2	2	34	2	3	10	1	1
18	25	2	2	10	1	1	60	3	3
19	10	1	1	40	2	3	40	2	2
$\tilde{b}_i = (b_i, \gamma_i, \delta_i)$									
b_7		γ_7	δ_7	b_8	γ_8	δ_8	b_9	γ_9	δ_9
600		0	50	580	30	30	930	0	90

TABLE 3. Fuzzy cost coefficients.

$\tilde{c}_i^s = (c_i^s, \lambda_i^s, \rho_i^s)$, s=1 for i$_i$10, s=2 for i $\geq$ 10							
i	c_i^s	λ_i^s	ρ_i^s	i	c_i^s	λ_i^s	ρ_i^s
1	500	0	20	14	780	30	0
2	400	0	20	15	300	20	0
3	380	0	16	16	140	5	0
4	350	0	20	17	150	10	0
5	405	0	20	18	250	10	0
6	565	0	25	19	100	5	0
7	500	0	30	20	0.1	0.01	0
8	1400	0	50	21	0.75	0.25	0
9	600	0	30	22	0.04	0.02	0
10	180	20	0	23	13	2	0
11	170	10	0	24	0.8	0.1	0
12	180	10	0	25	0.9	0.2	0
13	175	10	0	26	1	0.2	0

Three criteria are used to evaluate solutions:

- **gross profit**

$$f_1 = \tilde{c}_1^1 x_1 + ... + \tilde{c}_9^1 x_9 - \tilde{c}_{10}^2 x_{10} - ... - \tilde{c}_{26}^2 x_{26} + \sum_{s=1}^{S} \tilde{T}_s h_s$$

where fuzzy cost coefficients are presented in Tab.3 and constant $\sum_{s=1}^{S} \tilde{T}_s h_s$ is equal to (22000,0,0).

- **structure-forming plants area**

$$f_2 = c_6^3 x - 6 + c_7^3 x_7 + c_{16}^3 x_{16} + c_{17}^3 x_{17}$$

where $c_6^3 = 0.5$, $c_7^3 = c_{16}^3 = c_{17}^3 = 1$.

- **manpower hire**

$$f_3 = c_{24}^4 x_{24} + c_{25}^4 x_{25} + c_{26}^4 x_{26}$$

where $c_{24}^4 = c_{25}^4 = c_{26}^4 = 1$. After entering the data we get individual optima for the criteria that are calculated with fuzzy coefficients of the problem fixed on their middle values:

$$\text{Max } \tilde{f}_1 = 20716,$$

$$\text{Max } \tilde{f}_2 = 17.580,$$

$$\text{Min } \tilde{f}_3 = 180.52.$$

According to the farmer suggestion, we take the following aspiration levels[1]:

$$\text{for } f_1 \text{ - } \tilde{g}_1 = (20716,1716,0),$$

$$\text{for } f_2 \text{ - } \tilde{g}_2 = (4, 0.5, 0.5),$$

$$\text{for } f_3 \text{ - } \tilde{g}_3 = (500, 50, 50),$$

Now we must define safety coefficients. Let's take them in conformity with program suggestion $\tau_i = 0.6, \eta_i = 0$ and $\theta_i = 16.7$ for i=1,....,m. Approximate ranges of variation of θ_i are: $\theta_7 \in$ [-20.1,50],$\theta_8 \in$ [-47.7,20] and $\theta_9 \in$ [13.4,90].

Then, the fuzzy problem is transformed into the deterministic one. For this new problem the pay-off table is calculated and a starting point of the Choo-Atkin's method is searched for. The first starting point is y_0=(-2.64,13.65,-0.72).

Now we start with examination of the efficient border of the transformed problem looking for the best compromise solution. In each iteration, the number of compromise solution S to be compared must be a multiple of the number of criteria. Let's l_j denote the l-th solution obtained in the j-th iteration.

Iteration 1.

S=15. We obtain 15 solutions listed in Tab.4. In this table, only middle values of criteria for particular efficient points are shown. We have got a wide range of criteria values.

[1]Although we use here a representation off fuzzy numbers consistent with their definition, in FLIP we introduce directly the values of left and right ranges insted of left and right spreads

After first short analysis we can point out solutions that are not attractive because of too high deviation of values of particular objectives from their aspiration levels, e.g for gross profit - solutions $1_1, 2_1, 3_1, 5_1, 6_1, 8_1, 9_1, 11_1, 12_1, 14_1, 15_1$.

From among solutions $4_1, 7_1, 10_1, 13_1$, solution 10_1 is chosen as a new starting point for the next iteration. Now we are looking for a solution with a lower value of criterion z_3.

TABLE 4.

	1_1	2_1	3_1	4_1	5_1	6_1	7_1	8_1
z_1	15618	15645	15644	18891	14022	14126	20088	12506
z_2	10.8	10.8	10.8	8.5	14.9	8.5	6.1	17.6
z_3	536	536	538	648	648	373	760	690

	9_1	10_1	11_1	12_1	13_1	14_1	15_1
z_1	12539	20568	11178	13339	20763	9940	14605
z_2	6.1	3.7	17.6	3.8	1.3	17.6	1.4
z_3	278	872	600	241	983	516	215

Iteration 2.

Let's take S=12. We have got 8 solutions (Tab.5) and 4 of generated problems are contradictory. When we are searching through the feasible region, we make successive reductions of it and it can lead, in some cases, to contradictory problems.

In this iteration we have got three very interesting solutions: $1_2, 7_2$ and 10_2. For our DM, solution 7_2 is the best (Fig.16).

TABLE 5.

	1_2	3_2	4_2	6_2	7_2	9_2	10_2	12_2
z_1	19045	20567	19647	20557	19878	20547	19952	20537
z_2	3.7	3.7	3.1	1.9	4.0	0.2	4.0	0.1
z_3	503	937	539	905	575	897	611	891

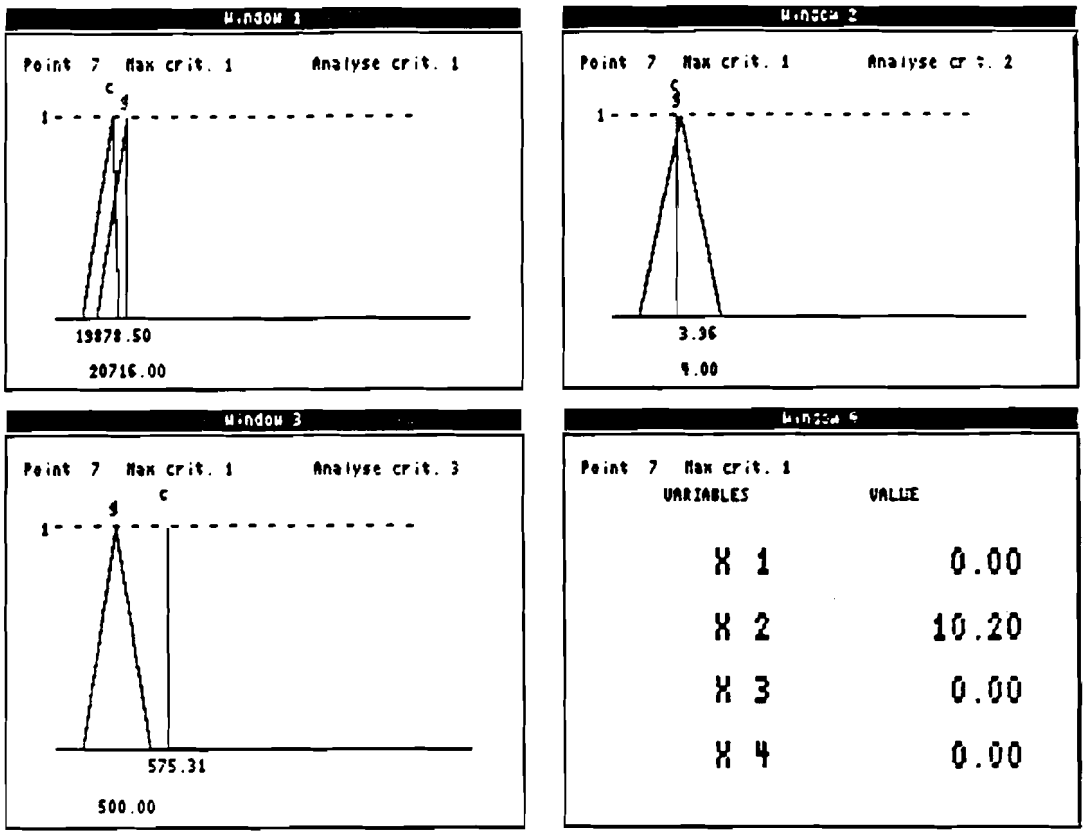


Figure 16. Values of criteria for solution 7_2.

As we can see, for solution 7_2, the value of the gross profit and the structure-forming plants area are within the range of the fuzzy aspiration levels. To get full evaluation of solution 7_2, let us also analyze the state of fuzzy constraints. In the upper windows of Fig.17, we have constraints corresponding to the demand of manpower in the spring and summer periods. As we can see, this demand has been satisfied with some surplus and no risk of violation of these constraints. For a constraint in the lower window that corresponds to the demand of manpower in the autumnal period, the risk of violation is very small. So, we can try to relax constraints 7, 8, 9 by changing safety coefficients.

Let's assume $\tau_i = 0.6, \eta_i = 0$ and $\theta_i = 0$ for all i.

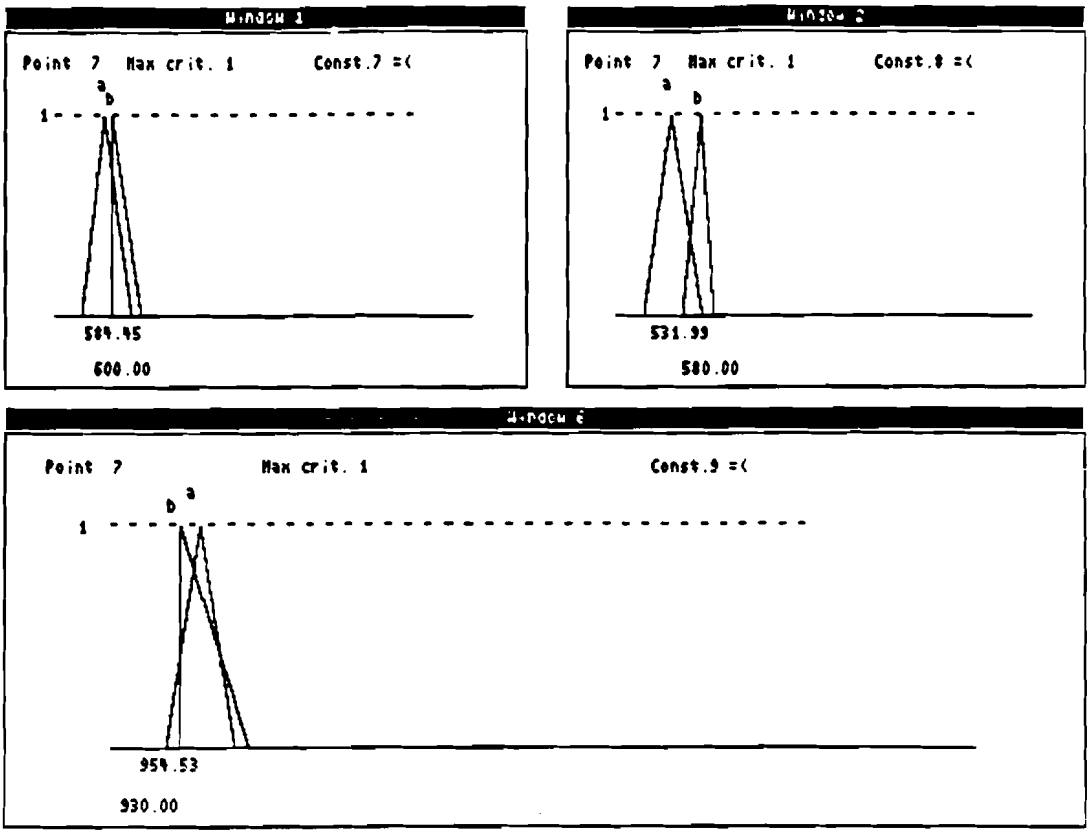


Figure 17. Fuzzy constraints for solution 7_2.

Iteration 1'.

For S=15 we obtain solutions listed in Tab.6.

TABLE 6.

	$1_{1'}$	$2_{1'}$	$3_{1'}$	$4_{1'}$	$5_{1'}$	$6_{1'}$	$7_{1'}$	$8_{1'}$
z_1	15642	15652	15651	18969	14039	14133	20167	12524
z_2	10.6	10.6	10.6	8.3	14.7	8.3	5.9	17.6
z_3	492	492	493	602	602	333	713	656

	$9_{1'}$	$10_{1'}$	$11_{1'}$	$12_{1'}$	$13_{1'}$	$14_{1'}$	$15_{1'}$
z_1	12554	20595	11202	13319	20692	9969	14596
z_2	5.9	3.4	17.6	3.5	2.1	17.6	1.1
z_3	236	823	566	198	933	483	171

Iteration 2'.

Taking solution $10_{1'}$ as a starting point and S=12, we get 8 new solutions (Tab.7).

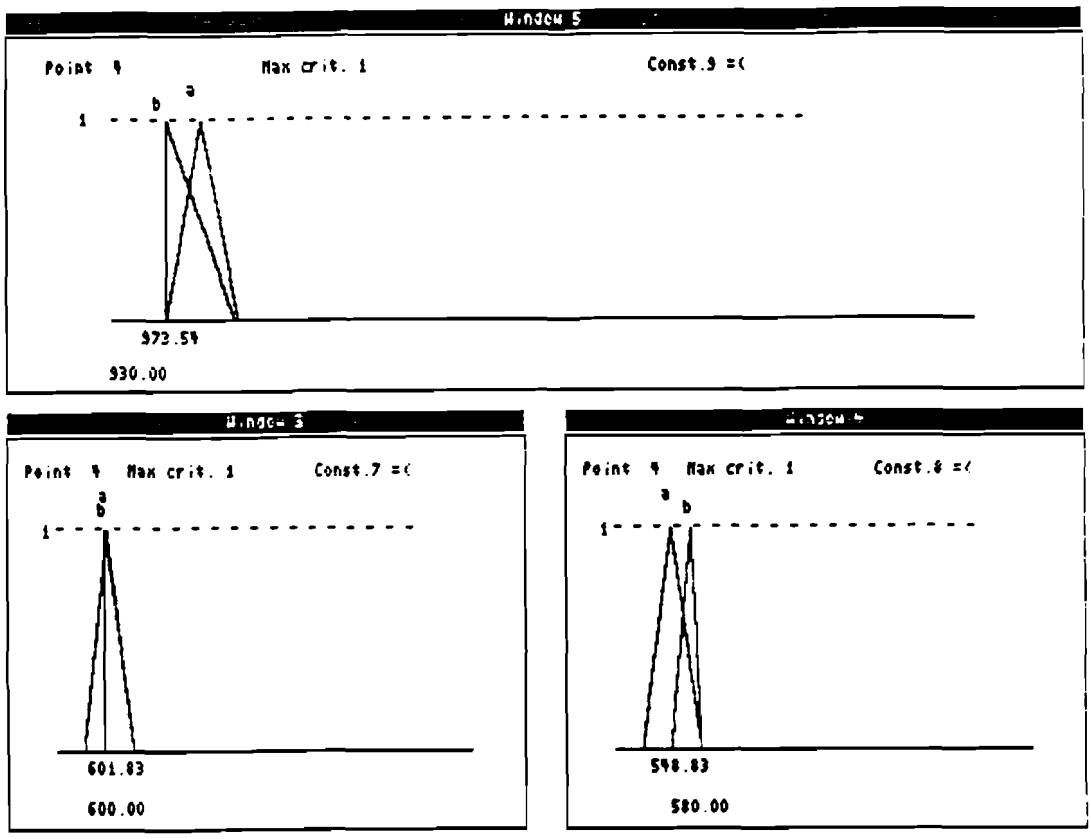


Figure 18. Fuzzy constraints for solution $4_{2'}$.

TABLE 7.

	$1_{2'}$	$3_{2'}$	$4_{2'}$	$6_{2'}$	$7_{2'}$	$9_{2'}$	$10_{2'}$	$12_{2'}$
z_1	19707	20594	19954	20582	20058	20570	20164	20558
z_2	3.2	3.4	4.0	1.7	4.0	0.0	4.0	0.3
z_3	503	862	541	841	584	833	624	826

For solution $4_{2'}$ the middle values of all objectives are within the range of their fuzzy aspiration levels. Fig.18 shows the state of fuzzy constraints for this solution. As we can see, there is no risk of violation of constraints 7 and 8 (lower windows) and a little bigger

risk for constraint 9, then in solution 7_2.

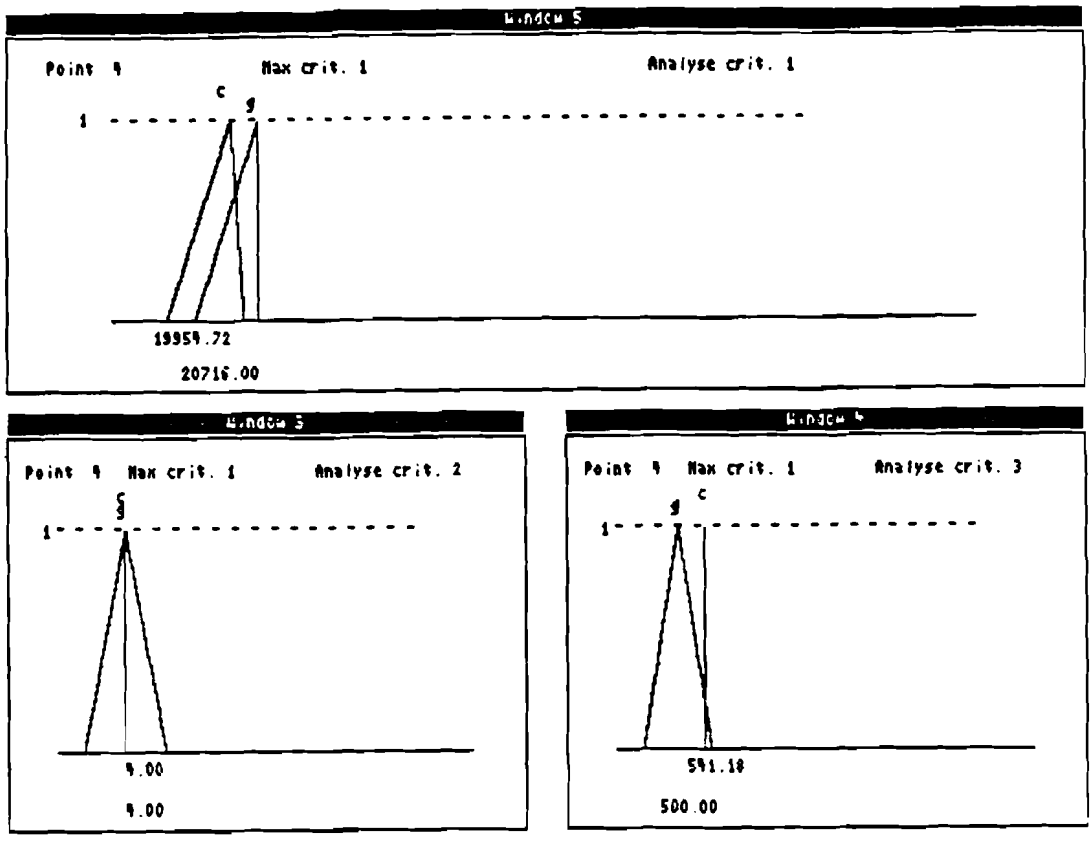


Figure 19. Values of criteria for solution $4_{2'}$.

Finally, the DM has accepted solution $4_{2'}$ as the best compromise. The values of corresponding criteria are presented in Fig.19.

So, we have got the following definition of the best-compromise farm structure:

a) plant production for sale:

- 0.80 ha of winter wheat,
- 9.20 ha of winter barley,
- 0.67 ha of spring barley,
- 2.75 ha of potatoes;

b) plant production for fodder consumed in the farm:

- 1.92 ha of potatoes,
- 0.5 ha of fooder beat,
- 4 ha of lucerne;

c) 4 ha of permanent grassland under cultivation;

d) purchase of fertilizers:

- 3267 kg of phosphorus,
- 2866 kg of nitrogen,

- 2397 kg of potassium;

e) manpower hire:

- 281 hour's in the spring period,
- 260 hour's in the summer period.

References

[1] Choo,E.U.,Atkins,D.R.(1980) *An interactive algorithm for multicriteria programming.* Computers and Operations Research 7, 81-87.

[2] Czyżak,P.(1989) *Multicriteria agricultural problem solving under uncertainty* Foundations of Control Engineering 14, no.3.

[3] Czyżak,P.(1990) *Application of the FLIP method to farm structure optimization problems* in R.Słowiński, J.Teghem (eds.), Stochastic vs. Fuzzy Approaches to Multiobjective Mathematical Programming, D.Reidel Pub. Comp.

[4] Czyżak,P. Słowiński,R.(1989) *Multiobjective diet optimization problem under fuzziness* in J.L.Verdegay, M.Delgado (eds.), The Interface between Artificial Intelligence and Operations Research in Fuzzy Environment, Verlag TUV Rheinland, Koln, pp.85-103.

[5] Delgado, M., Verdegay, J.L., Vila, M.A.(1989) *A general model for fuzzy linear programming* Fuzzy Sets and Systems 29, 21-30.

[6] Dubois, D. (1987) *Linear programming with fuzzy data* in J.C.Bezek (ed.), Analysis of Fuzzy Information, Vol.3:Applications in Engineering and Science, CRC Press, Boca Raton, pp. 241-263.

[7] Inuiguchi,M., Ichihashi,H., Tanaka, H. (1990) *Fuzzy programming: a survey of recent developments* in R.Słowiński, J.Teghem (eds.), Stochastic vs. Fuzzy Approaches to Multiobjective Mathematical Programming, D.Reidel Pub. Comp.

[8] Luhandjula, M.K. (1989) *Fuzzy optimization: an appraisal* Fuzzy Sets and Systems 30, 257-282.

[9] Rommelfanger, H. (1989) *Interactive decision making in fuzzy linear optimization problems* European J. Operational Res. 41, 210-217.

[10] Sakawa, M., Yano, H. (1989) *Interactive fuzzy decision making for generalized multiobjective linear programming problems with fuzzy parameters* Fuzzy Sets and Systems 29, 315-326.

[11] Słowiński, R. (1986) *A multicriteria fuzzy linear programming method for water supply system development planning* Fuzzy Sets and Systems 19, 217-237.

[12] Słowiński, R. (1987) em An interactive method for multiobjective linear programming with fuzzy parameters and its application to water supply planning in J.Kacprzyk, S.A.Orlovski (eds.), Optimization Models using Fuzzy Sets and Possibility Theory, D.Reidel, Dordrecht, pp.396-414.

[13] Słowiński, R., Teghem, J. (1990) *Comparison study of STRANGE and 'FLIP'* in R.Słowiński, J.Teghem (eds.), Stochastic vs. Fuzzy Approaches to Multiobjective Mathematical Programming, D.Reidel Pub. Comp.

[14] Słowiński, R., Teghem, J., eds. (1990) *Stochastic vs. Fuzzy Approaches to Multiobjective Mathematical Programming* D.Reidel, Theory and Decision Library, Dodrecht.

[15] Słowiński, R., Urbaniak, A., Węglarz, J. (1986) *Probabilistic and fuzzy approaches to capacity expansion planning of a water supply system* in L.Valadares Tavares, J.Evaristo da Silva (eds.), Systems Analysis Applied to Water and Related Land Resources, Pergamon Press, Oxford, pp.93-98.

[16] Vanderpooten, D.(1990) *Multiobjective programming: basic concepts and approaches* in R.Słowiński, J.Teghem (eds.), Stochastic vs. Fuzzy Approaches to Multiobjective Mathematical Programming, D.Reidel Pub. Comp.

Application of min/max Estimates for Decision Making under informational uncertainty

Edward Nawarecki
Institute for Computer Science
Academy of Mining and Metallurgy

Grzegorz Dobrowolski
Joint System Research Department
of the Institute for Control and Systems Engineering,
Academy of Mining and Metallurgy, Cracow,
and the Industrial Chemistry Research Institute, Warsaw.

Abstract

The paper deals with *decision situations* for which the *optimization model* can be formulated but with a relatively great out-of-model sphere. The optimization model is computerized in the shape of *decision support system* that assures easy simulation experiments and supports all necessary auxiliary actions.

For the problem with uncertain data and a wide out-of-model sphere the optimality looses its sense when a solution is tried to be realized. It may be even a limiting factor when attractive optimal states of the model draw the decision maker's attention to unrealistic states of reality.

A natural tactics of the decision maker is to consider states of the model that are pre-defined (variants) on the basis of some states of reality and ought to be close (in some sense) to the optimal state in the sense of the criterion used.

In the paper the concept of *min/max graphs* is proposed to devise management of the variants. Basic definitions and theorems important for the application are reproduced. A central position is occupied by theorems about estimation of the graphs that open possibility to avoid investigation of the majority of variants. A simple procedure is sketched out.

It is expected that the min/max graphs can create a new skeleton for DSS building.

1 Introduction

The paper deals with decision situations that may be characterized as follows:

- Input data are uncertain and it is hard or even impossible to evaluate deviation introduced by them.
- As a basic mathematical tool the optimization problem is used.

- A bunch of phenomena staying beyond the optimization model is numerous.
- The influence of unmodeled elements is taken into consideration by simulation scenarios produced by a human.
- The optimization model is computerized in the shape of a Decision Support System **DSS** that assures easy simulation experiments and supports all necessary auxiliary actions.

The decision situations can be classified according to (Lewandowski and Wierzbicki, 1988) as the centralized single-actor (decision maker) situations. They involve: a user and a DSS. The user personifies a whole staff with the decision maker itself, it means: experts, analysts. The decision maker has the authority and experience to reach the decision; the experts and the analysts are responsible for the analysis of the decision situation.

A role of the optimization model in the DSS, characterized above, is different than the one in technical problems where the optimal solution is almost directly applicable. The solution enriched with results of the postoptimal analysis or other validation technics ends the analysis as long as consensus of adequacy of the model is gained.

For the problem with uncertain data and wide out-of-model sphere optimality looses its sense when the solution is tried to be realized. Application of the optimization algorithm meets another goal than searching for the best alternative. Generally, it plays a role of a data processing algorithm featured by drastic reduction of input versus output data.

Optimality in DSS of the type may be even a limiting factor when attractive optimal states of the model draw the decision maker's attention to unrealistic states of reality. A natural tactics of the decision maker is to consider the states of model that are pre-defined (variants) on the basis of some states of reality and ought to be close (in some sense) to the optimal states in the sense of the criterion used.

Let us assume that the way in which the variants arise has no direct links to the supported part of the decision process (uncomputerized human creative factor). A schema of dealing with optimality in the DSS discussed can be proposed:

1. To find the optimal solution.
2. To list the distinctive variables of the model (definition of the variants by the model variables).
3. To evaluate a consecutive variant — the optimal state of model under assumed behavior of the distinctive variables. The state is usually far from the optimal solution.
4. To verify and validate the variant also by comparison with the optimal solution.

It is assumed that the shape of the model is stable with respect to its structure and parameters what create a platform for the final choice from among few variants.

The schema is a *what-if* tactics elastically controlled by the optimal solution.

In the paper the concept of min/max graphs is proposed to devise management of the variants that can arise in the process described below.

Basic definitions and theorems important for the application are reproduced. A central position is occupied by theorems about estimation of the graphs that allow to avoid investigation of most variants. A simple procedure is sketched out.

2 A Model of Decision Process under Informational Uncertainty

Let us build the model on a basis of optimization problem with an objective function:

$$F(x) = F(x_1, \ldots, x_N) \tag{1}$$

defined on an admissible domain of realizations of an argument vector:

$$x \in X = \prod_{i=1}^{n} X_i$$

where: X_i a set of realizations of a component $x_i, i = 1, \ldots, N$.

The admissible set X_i can be a set of real numbers (values of a physical parameter) or freely defined elements representing a class of phenomena, or even strategies (chains of decisions and incidents). For the sake of simplicity, it is assumed that X_i is a set of numbers representing realizations $x_i \in X_i$. The decision process searches for such $x^* \in X$, that the minimum (or maximum) of the objective function is gained:

$$F(x^*) = \min F(x_1, \ldots, x_N) \tag{2}$$

To model the informational uncertainty, it is assumed that only some components of $x = [x_1, \ldots, x_N]$ are controlled by a decision maker (their realizations can be freely chosen). The remaining components stay independent from decisions to be taken. Their realization can be observed by the decision system and only their domains X_i are predictable.

Moreover, it is assumed that composition of the set X in the sense of chosen X_i as well as a dichotomy into these under control and those subordinated under the active partner activity or independent part of reality can vary in the consecutive stages of the decision process. The above assumptions reflect features of a wide class of real decision problems, such as economic decision making, technical and medical diagnosis.

In the paper some representation of knowledge in the decision problem as above is proposed together with an inference engine that allows effective decision procedures. The dichotomy can be formulated as follows:

$$U \times V = X \tag{3}$$

where: U a set of parameters under the decision maker's control;
V a set of parameters unknown and independent from the decision system.

After reindexing the sets U, V is done:

$$[u_1, \ldots, u_n] \in U \quad ; \quad [v_1, \ldots, v_m] \in V$$

and $n + m = N$ holds. The objective function changes its denotation: $F(x) \equiv F(u, v)$.

For given U and V the objective function $F(u, v)$ gains its saddle point (Kluska-Nawarecka and Nawarecki, 1982) (Kluska-Nawarecka and Nawarecki, 1987)what can be written as:

$$\min_{u \in U} \max_{v \in V} F(u, v) = \max_{v \in V} \min_{u \in U} F(u, v) = F(u^*, v^*) \tag{4}$$

where: u^*, v^* min/max strategies of, respectively, the decision maker and an opponent.

In practice, the assumption of a saddle point existence does not create a limiting factor because there is a margin of elasticity in X construction and, in consequence, in uv-dichotomy. A triple $\langle F, U, V\rangle$ is called a decision situation.

The loss (decreasing of the objective function F) that comes from the opponent's activity or independence of realization of parameters belonging to the set V can be expressed as a difference between solutions of the problems (2) and (4).

$$\Lambda(U,V) = \min_{u\in U}\max_{v\in V} F(u,v) - \min_{x\in X} F(x) = F(u^*,v^*) - F(x^*) \tag{5}$$

Obviously, the biggest loss is observed when the set of controlled parameters is empty $U = \emptyset$.

For a given objective function F the value of $\Lambda(U,V)$ depends only on the uv-dichotomy and can characterize the decision situation $\langle F, U, V\rangle$.

The loss, defined as above, in real-life decision problems for the defined decision situation $\langle F, U, V\rangle$ is lesser than theoretical $\Lambda(U,V)$ given by (5). In average $v \neq v^*$ holds and

$$F(x^*) < F(u^*,v) < F(u^*,v^*) = F(x^*) + \Lambda(U,V) \tag{6}$$

Having two decision situations $\langle F, U, V\rangle$ and $\langle F, U', V'\rangle$ so that:

$$U' = U \times X_k \quad ; \quad V' = \prod_{i=1;i\neq k}^{n} V_i \tag{7}$$

(it means that the parameter X_k is excluded from V and joins U), the following inequality holds:

$$\Lambda(U',V') \leq \Lambda(U,V) \tag{8}$$

and the estimation can be done:

$$\lambda(X_k,U,V) = \min_{u\in U}\max_{v\in V} F(u,v) - \min_{u\in U'}\max_{v\in V'} F(u,v) \tag{9}$$

Defined that way $\lambda(X_k,U,V)$ is interpreted as information value of the set X_k. It means that if the parameter represented by X_k was measured and controlled the value of the objective function in that case could be approximated using $\lambda(X_k,U,V)$. It is worth while to underline that $\lambda(X_k,U,V)$ depends not only on X_k but also on the decision situation (the uv-dichotomy) as well.

3 A graph of min/max solutions

3.1 Definitions

A graph uniquely described by the objective function $F(x)$ and the admissible set of the parameter realizations $x \in X$ is called a graph of min/max solutions or shortly a min/max graph and denoted by $\Gamma(F,X)$. For each solution to the problem (4) corresponding to a given decision situation (the uv-dichotomy of X) a node $A(U,V)$ of the min/max graph exists.

A number of parameters grouped in the sets U and V introduces partition of the min/max graph $\Gamma(F,X)$ into levels, as follows:

- A starting node A^0 of the min/max graph $\Gamma(F,X)$ corresponds to the solution of problem (2), i.e., to the dichotomy $U = X; V = \emptyset$. The node A^0 is the only on the level 0.

- A final node A^N corresponds to the case of maximization with respect to all parameters; uv-dichotomy is $U = \emptyset; V = X$. It is the single node on the level N.

- The rest of nodes of the min/max graph $\Gamma(F, X)$ corresponds to the solutions of (4) for all uv-dichotomy. A node A^m belonging to the level m represents a solution of (4) when $V = \prod_{i=1}^{m} X_i$ and $U = \prod_{i=m+1}^{N} X_i$.

A min/max graph has $N+1$ levels. Levels 0 and N have a single node. A number of nodes on remaining levels is equal to a number of possible uv-dichotomies; on the level 1 set V contains a single parameter each time and a number of nodes is equal to N, on the level 2 V contains a pair of parameters, etc. Generally, on the level m there are $\Omega_m = \begin{bmatrix} N \\ m \end{bmatrix}$ nodes. Total number of nodes of the min/max graph is given by the formula:

$$\Omega = \sum_{m=0}^{N} \frac{N!}{m!(N-m)!} \tag{10}$$

To point at the particular node (except A^0 and A^N) the level index is insufficient. It is necessary to enumerate all the parameters in the U or V set and, in fact, m or $N - m$ indexes are needed. Below, whenever it does not cause misunderstanding, simplified indexing will be used: a superscript will point at the level of the node and a subscript will differentiate the nodes of the same level.

Two nodes A^m, A^n of the min/max graph $\Gamma(F, X)$ are joined by an arc $a\,(A^m, A^n)$ if and only if they belong to the neighboring levels (i.e., $n = m + 1$ or $n = m - 1$) and, simultaneously, the uv-dichotomies corresponding to the nodes differ only in a parameter X_i that is excluded from the set U and bound to V when one steps down the level. It may be written as an implication:

$$a\,(A^m, A^n) \in \Gamma(F, X) \longrightarrow U^m = U^n \times X_i \vee U^n = U^m \times X_i \tag{11}$$

It is comfortable, especially, when one constructs a graphical form of the min/max graph $\Gamma(F, X)$ to introduce a normalization according to the following assumptions:

- the value F^* obtained as $\min_{x \in X} F(x)$ is taken as the reference level;

- values $\underline{F}(x)$ for nodes are calculated according to:

$$\underline{F}(U_i^m, V_j^m) = \underline{F}(A_i^m) = \frac{\max_{v \in V_j^m} \min_{u \in U_i^m} F(u, v) - \min_{x \in X} F(x)}{\max_{x \in X} F(x) - \min_{x \in X} F(x)} \tag{12}$$

- following (12) $\underline{F}$ is equal to 0 for the starting node and to 1 for the final one.

Because the normalization means change of values associated with the nodes (a scale of the graph) and does not affect features of the min/max graph the normalization symbol will be neglected next.
Some ideas and definitions will be introduced about the min/max graphs.

Two nodes A^m, A^n are called neighboring if and only if there is an arc joining them $a\,(A^m, A^n)$. The definition can be formulated:

$$A^m \sim A^n \longleftrightarrow a\,(A^m, A^n) \in \Gamma(F, X) \tag{13}$$

where: $\sim$ a symbol denoting the neighborhood relation.

Arcs joining a sequence of neighboring nodes that belong to consecutive levels create a path.

A set of nodes belonging to the same level $\{A_i^m, A_j^m, \ldots, A_r^m\}$ is ordered if the relation is known:

$$F(A_i^m) \leq F(A_j^m) \leq \ldots \leq F(A_r^m) \tag{14}$$

If the ordered set $\{A^m\}$ contains all nodes of the level the first and the last elements with respect to the relation are called the lower A_0^m and the upper A_M^m node of the graph $\Gamma(F, X)$ on the m level.
The value of:

$$\sigma^m = F(A_M^m) - F(A_0^m) \tag{15}$$

will be called a width of the min/max graph.

A set groups all upper nodes of the graph for the consecutive levels $\{A_M^m\}$, $m = 0, \ldots, N$ described an upper envelope of the graph $\Gamma(F, X)$. In the analogous way a lower envelope is defined.

A conjugate node $\hat{A}_i^m = A_j^{N-m}$ with the given node A_i^m is a node for which sets U_j,V_j are complements of U_i, V_i to the whole X.

$$U_i^m = V_j^{N-m} \quad ; \quad V_i^m = U_j^{N-m} \tag{16}$$

A conjugate node corresponds to the replaced uv-dichotomy.

If a path $D_{i,k,\ldots,p}$ that joins nodes $A_i^m, A_k^{m+1}, \ldots, A_p^{m+r}$ exists then a sequence of the conjugate nodes $A_j^{N-m} = \hat{A}_i^m$, $A_l^{N-m} = \hat{A}_k^{m+1}, \ldots, A_q^{N-m} = \hat{A}_p^{m+r}$ creates the unique path $D_{j,l,\ldots,q}$. The path is called conjugate:

$$D_{i,k,\ldots,p} = \hat{D}_{j,l,\ldots,q} \tag{17}$$

3.2 Relation and Estimation

For two neighboring nodes A_i^m, A_j^n of the min/max graph $\Gamma(F, X)$ the following inequality holds:

$$F(A_i^m) \leq F(A_j^n) \quad ; \quad m < n \tag{18}$$

Applying the inequality to consecutive pairs of the neighboring nodes it is easy to observe that for a triple of nodes A_i^m, A_k^s, A_j^n joining with the path $D_{i,\ldots,k,\ldots,j}$ the following estimation can be obtained:

$$F(A_i^m) \leq F(A_k^s) \leq F(A_j^n) \quad ; \quad m < s < n \tag{19}$$

For a triple of nodes belonging to different levels no matter the joining path exists or not a weaker estimation is true:

$$F(A_0^m) \leq F(A_k^s) \leq F(A_M^n) \quad ; \quad m < s < n \tag{20}$$

where: A_0^m the lower node of level m,
A_N^n the upper node of level n.

The estimation (20) can be improved in the way:

$$F(A_{i-1}^m) \leq F(A_k^s) \leq F(A_{j+1}^n) \quad ; \quad m < s < n \tag{21}$$

where: A^m_{i-1} the node of level m with $F(x)$ lesser than for the node of level m joined with A^s_k,
A^n_{j+1} the node of level n with $F(x)$ greater than for the node of level n joined with A^s_k.

Let $F(x), G(x)$ be two objective functions defined on X. Let $\Gamma(F,X) = \Gamma_F$ and $\Gamma(G,X) = \Gamma_G$ be min/max graphs for the objective functions, respectively. If Γ_F and Γ_G are normalized their starting and final nodes overlap each other $A^0 = B^0$, $A^N = B^N$.

Depending on mutual location of nodes A^m_i and B^m_j on remaining levels $m = 1, 2, \ldots, N-1$ the following relations for the min/max graphs Γ_F and Γ_G can be defined:

- inclusion denoted by $\Gamma_G \subseteq \Gamma_F$ when:

$$G(B^m_M) \leq F(A^m_M) \text{ and } G(B^m_0) \geq F(A^m_0) \quad ; \quad m = 0, 1, \ldots, N \tag{22}$$

where: A^m_0, A^m_M the lower and upper nodes of level m of the graph Γ_F;
B^m_0, B^m_M the lower and upper nodes of level m of the graph Γ_G;

- dominance denoted by $\Gamma_G \prec \Gamma_F$, when:

$$G(B^m_M) \geq F(A^m_M) \text{ and } G(B^m_0) \geq F(A^m_0) \quad ; \quad m = 0, 1, \ldots, N \tag{23}$$

- conformity of order denoted by $\Gamma_G \approx \Gamma_F$. The conformity effects when on each level for the ordered set of nodes $\{A^m_i\}, \{B^m_i\}$ holds:

$$F(B^m_i) \geq F(A^m_j) \longleftrightarrow G(B^m_i) \geq G(A^m_j) \quad ; \quad m = 0, 1, \ldots, N \tag{24}$$

where: $(\{A^m_i\}, \{B^m_i\}); (\{A^m_j\}, \{B^m_j\})$ pairs of nodes of the graphs Γ_F and Γ_G accord each other with respect to the uv-dichotomy.

- pseudoconjugate nodes denoted by: $A^m_i; \hat{A}^m_i = B^{N-m}_j$ and defined analogously to conjugate nodes with respect to the uv-dichotomy (according to (16)) but evaluated by the objective functions $F(x)$ and $G(x)$, respectively;

- pseudoconjugate paths defined analogously to conjugate paths but for pseudoconjugate nodes.

4 A framework for the DSS

On the basis of theoretical preliminaries an impression can arise that all the specified variables of the model are taken into account, are uv-dichotomized, and the min/max graph acquires enormous number of nodes and finally the question appears: Is the concept really efficient? Fortunately, in the technical problems the dimensionality is not so large and moreover many variables stay beyond the influence of the decision process. The evaluation methods and algorithms are often at hand that allow for excluding some of the variables from the analysis by their approximation with a given accuracy. For the decision problems with uncertain data and a wide out-of-model sphere the limiting factor in the subject is ability of the decision maker to deal with a number of alternatives (variants) in the comparison regimè.

In practice, there is no need to generate and analyze the *complete* min/max graph. Instead the *partial* graph (built with respect to the chosen subset of the variables) is pondered and, of course, theoretical results hold for that case. The trick well corresponds to the idea of the optimization problem with distinctive variables (Gass and Dror, 1983) or with the assumption of existence of the decision maker's preference in the variable space (Dobrowolski, 1990).

As it will be shown, generating all nodes even of the partial graph is not necessary. The analysis can be successfully completed with an approximated form of the graph, some nodes of which are estimated using the rules presented in the previous section.

4.1 Remarks on architecture

The concept of the min/max graph is introduced as a consequence of the optimization problem. The graph represents a set of solutions (variants) of the problem obtained in the procedural way. Such representation has the following valuable features:

- creates a common platform for a number of variants, especially for comparison;
- opens way for graphical representation of the set of variants;
- can influence the decision maker to carry out the analysis in more precise and systematic way;
- utilizing the estimation features of the min/max graph, allows for avoidance of detailed investigation of some variants.

The above suggests that the concept can be used in the DSS in the model management and dialog management subsystem (Sprague, 1980).

No sooner than the variant generation procedure is established the only link between the problem and the min/max graph is a set of values of the objective function. Then the concept can be encapsulated in the separate module of the DSS. Let us call the module the UV module. Because no assumptions were made with respect to the shape of the objective function, the UV module may be built independently of the DSS means.

If an interactive mode will be assumed, communication between the UV module and the remaining part of the DSS is as follows: the module asks for the solution of the optimization problem corresponding to the node approximated so far, as an answer it obtains the value of the objective function and the new approximation of the min/max graph may be calculated. Each time the approximation is analyzed by the decision maker and the process may be interrupted when the decision field (the current state of the min/max graph) becomes clear to him.

4.2 Types of analysis

The strategy of the variants generation is a crucial point of the concept. It decides about the semantic of the min/max graph and finally about the usefulness of the concept for the particular decision problem. From the formal point of view, the following strategies are possible:

Sensitivity analysis The analysis can be carried out both with respect to variables and parameters of the model.

The first subcase directly arises from the formulation of the min/max graph concept. A variable is under control when its value is fixed otherwise it is free. For the linear optimization model a slight different schema can be interesting: a variable is fixed on its infimum versus supremum in an admissible set.

In the second subcase it can be assumed that influence of deviation of parameters in an interval is investigated. An appropriate schema is: a parameter is fixed on the lower bound of the interval next on the upper one.

The greatest value of (9) calculated for the all uv-dichotomies can serve here as a sensitivity measure. Graphical representation of the measure is the greatest jump for a variable or a parameter between neighboring levels of the min/max graph.

Improving the model The procedure is oriented towards minimalization of uncertainty introduced by elements of the model (admissible intervals for the variables, assumed deviations of the parameters). Under assumption that enhancement means additional cost, the results from the sensitivity analysis can indicate what elements of the model ought to be refined.

In its simpliest form (controllable variable is fixed) the procedure corresponds to looking over the min/max graph to find a subgraph for which the sensitivity measure (appropriate value of (9)) is the smallest or acceptable.

Unlimited what–if analysis Because there are no limitations imposed on the variants generation strategies completely free schema can be used as long as the decision maker can manage it.

5 Summary

The concept of the min/max graph and its utilization in the DSS application calls for the practical verification.
The UV module is completed now and several trials are planned to appraise its usefulness.
The concept needs also extensions for the multiobjective case.

6 References

Dobrowolski, G. (1990) Interactive Multiobjective Optimization Method in the Space of Model Variables. In print.

Gass, S. I. and Dror, M. (1983) An interactive approach to multipleobjective linear programming involving key decision variables. *Large Scale Systems* **5**, pp. 95–103.

Kluska-Nawarecka, S. and Nawarecki, E. (1982) Problème d'evaluation et de choix d'information dans les systèmes de commande. In Symposium Components and Instruments for Distributed Control Systems, Paris.

Kluska-Nawarecka, S. and Nawarecki, E. (1987) Decision Making in a System with Insuffcient Input Data. In ACIAM'87, Paris.

Lewandowski, A. and Wierzbicki, A. (1988) Aspiration Based Decision Analysis and Support. Theoretical and Methodological Backgrounds. *IIASA Working Paper* **WP-88-03**, International Institute for Applied System Analysis, Laxenburg, Austria.

Sprague, R. H. (1980) A framework for the development of decision support systems.. *MIS Quarterly* **4** (4), pp. 1–26.

Indexed Variables in Model Generators for Decision Support Systems

Jerzy Paczynski
Institute of Automatic Control
Warsaw University of Technology
00665 Warsaw, Poland

1 Motivation

One of the important classes of decision support systems are the systems based on analytical model, multi objective optimization and choice [3]. The main work of institutions in Poland, cooperating with the System and Decision Sciences Program of the International Institute for Applied System Analysis (IIASA), was concentrated on this class of systems and this paper follows the tradition although the results can be used in other classes of DSS too. The work, whose preliminary stage is reported in this paper, is motivated by experience gained in the using of the systems of the IAC-DIDAS family. Here are some impressions.

— The formulation of a model at the "mathematical level" differs from the formulation at the "computer input level". Especially, when the underlying problem is formulated by expressions which contain indexed variables, the latter is of much lower level. Besides, the formulation of a model (at the computer level) differs for each DSS.

E.g. the natural area of application of the system IAC-DIDAS-N are static problems. Their mathematical formulation corresponds to the formulation at the computer input level. Each variable has a name and optionally a descriptor of physical units. Input variables and parameters have values while output (outcome) variables have formulae defining their values. Additionally input and output variables have lower and upper bounds. When the user wants to apply this system to a problem not belonging to this class, its formulation has to be manually 'translated' to the former level.

The inconvenience of this process is (perhaps unintentionally) illustrated by the tutorial example on the distribution diskette, described in detail in [2]. It is a very rough approximation of the model of acid deposition in forest soil. Two regions are considered, denoted by the index k, and — as the problem is a dynamic one — there are three periods of time (each of five years duration) denoted by t.

One of the output variables is the sulphur dioxide emission in each region and time period determined by the formula

$$S_{k,t} = S^{p}_{k,t}(1 - p_{k,t}) \qquad k = 1,2;\ t = 1,2,3. \tag{1}$$

where $S^{p}_{k,t}$ is the potential emission and $p_{k,t}$ is the reduction coefficient which describes the effect of the pollution control.

In the described example for each k, t the value of is bounded from below by 0 and from above by 100. Its physical units are *tons*. The variables are taken as parameters, for each k, t they have the nominal value 100 and the units *tons*.

The mathematical description of this part of the model consists of the formula (1) and the information in the paragraph above. Such notation allows for arbitrary large number of regions and time periods — only the 'second' part of the formula (1), namely the receipt for changes of k and t, needs proper modification. Such formulation is referred to in this paper as being at the 'mathematical level'.

However, at the 'computer input level' explicit formulae for must be given for each pair of indexes k, t: $S11 = S11p(1 - p11)$, $S12 = S12p(1 - p12)$ etc. The variable $S11$ corresponds to $S_{1,1}$, $S11p$ corresponds to $S^p_{1,1}$ etc. Similarly the user has to repeat definitions of bounds, value and units for each scalar variable. In this case the number of variables is multiplied by 6. But, as in mentioned in [2]: "The problem ... should be considered in many time periods of one year duration". Thus, in more precise model the expansion factor would be much higher.

The main goal in the construction of the IAC-DIDAS family of systems was to achieve most effective and user friendly interaction. However, the maximum size of models, which can be effectively investigated in such systems is limited by the potential of the host computer. One possible scenario for overcoming this limitation is as follows. In the first phase a reasonably simplified model (tailored to the capabilities of the IBM-PC class of computers) is investigated in the user-friendly environment. After gaining sufficient feeling of the model properties another — larger — model is prepared and eventually solved on a more powerful computer with some suitable software (if both of them are attainable). Typically the software used will be some general purpose optimization package. Thus the following points must be highlighted:

— The model formats in both systems are usually different. The problem is how easy, or rather how cumbersome, is to produce the second model on the basis of the first one and to verify their correspondence,

— It may happen that a user is interested in the investigation of a whole class of models, differing e.g. by the number of discretization points. This task, if not properly helped by the system, can be boring, time consuming and result in various errors.

When eventually the results of an optimization experiment with a model are obtained, the problem of their efficient interpretation arises. The interaction with the user in this phase suffers if the used system is of general nature; again, the problem of a task-specific, intelligent interpretation of general data formats is important here.

The paper contains the proposal of a new standard for model definition, on higher level of notation. It is especially suited for models describing dynamic processes or other phenomena described by recurrence equations. The key issue of this standard is the concept of defining and handling of subscripted variables. They allow for concise formulation of the model, with strict correspondence between mathematical formulation and formulation at the program input. Those variables definitions can be automatically 'expanded' into sets of definitions of conventional scalar variables. The 'size' of the expanded model depends only on some easily adjustable parameters. The expanded model can be presented in the form of a text file. However, it is possible to provide program modules, which could transform an expanded model directly into the form acceptable by other, more 'classic' DSS or optimization systems.

2 Subscripted variables — syntax and semantics

2.1 Definition of Syntax

A typical ('scalar') variable in a DSS system is characterized by its name, a formula defining its value, values of the lower and upper bounds and optionally a string with denotation of physical units. All these items have to be transferred to a subscripted variable, with as much resemblance to mathematical notation as possible. The following solution was proposed and implemented.

The syntax will be presented in the notation of Modified Backus-Naur Form. The following meta-symbols will be used:

=	—	denotes the definition,
|	—	separates alternative options within the clause,
"..."	—	terminal symbols are quoted,
(...)	—	exactly one of the enclosed alternatives must be selected,
[...]	—	denotes zero or one occurrence of the enclosed subclause,
{...}	—	denotes zero or any number of occurrence of the enclosed subclause.

A subscripted variable definition can be conceptually divided into two parts. In the first one the names of subscripts are defined, together with their ranges and patterns of their change. The syntax of the first part is as follows:

```
for_sequence = for_phrase { for_phrase } "define"
for_phrase   = "for" contr_var_name ("=" | ":=") signed_integer
               ("to" | "downto") signed_integer "do" |
               "for" contr_var_name ("=" | ":=") signed_integer
               "step" signed_integer ("until" | "to")
                signed_integer "do"
```

Here are some examples:

```
for i:= 2 to 7 do define,
for k = 5 downto -5 do define,
for ab = 3 step 5 until 15 do define,
for a := 1 to 8 do
    for b := 13 downto 4 do define.
```

The syntax is rather self explaining. Introduction of two pairs of synonyms (`:=` and `=` , `until` and `to`) eliminates the most probable source of syntactic errors. The subscripts are referred to as control variables.

The semantics of this part is as follows. The declaration introduces names of subscripts. Their scope is limited to the second part of the present declaration. In another declarations they can be freely used. The subscripts are restricted to the integer type. It seems that will be sufficient in nearly all cases. The definition describes the value of the increment of a control variable (e.g. $+1$ in the first example, -1 in the second, $+5$ in the third) and initial and final values. Note that in the case of *step until* pattern the final value may never be achieved, e.g. in the third example the variable *ab* will assume values of 3, 8 and 13. Thus no restriction is put on the allowed ranges of indexes (as e.g. only positive values or additionally starting with value one etc.). The user can express his

problem just as it is most conveniently described in mathematical terms, without need for any normalization. It seems that the proposed patterns are sufficiently flexible for typical applications. However only the practice will show, whether some other more sophisticated strategies will be needed.

The last problem concerns definitions which introduce more than one subscript. In such cases the order of change has to be prescribed. In the experimental implementation the Pascal-like solution was chosen, i.e. the right-most variable is changing fastest. It is easy to produce the Fortran-like solution, i.e. with reversed order of change. The significance of this problem will appear more visible when the second part of the definition will be presented.

In the second part the following items are defined:

- name of the indexed variable,
- the formula defining its value,
- the formula defining its lower bound,
- the formula defining its upper bound,
- units (or any other description).

Note that bounds are allowed to be expressions while for scalar variables they are usually constant values.

The most important item is the concept of an indexed name. Here is the corresponding syntax:

```
indexed_name = name one_index_segment { [plain_name]
               one_index_segment }  [ plain_name ]
one_index_segment = "[" expression { "," expression } "]"
name         =  letter { letter | number | "_" }
plain_name = (letter | number | "_" ) { letter | number | "_" }
```

Here are some examples:

```
ab[x+2,y-1] ,          ab[x+2][y-1]  ,
jp[c,d]_[x] ,          xyz[i,j]12[k]i .
```

The prefix of an indexed name must be a 'usual' name, i.e. it must begin with a letter. The separators between indexed segments, referred to in the syntax definition as *plain_name*, can start with any character admissible in names. A separator may be missing,as e.g. in the second example. Such case is considered equivalent to the first example. The *one_index_segment* contains in square brackets one or any greater number of expressions, separated by commas. The syntax of these expressions will not be pursued farther, they are built of constants and subscript names and must evaluate to an integer value.

The formula definition may follow the := or = symbol or alternatively may appear after the system prompt. It is assumed to be:

```
formula = any sequence of text and indexed_names
```

It must of course represent a sensible mathematical expression. However, as one does not know at the moment of formula analysis, which programming system will be the potential consumer of the output of described piece of software, it is impossible to prescribe the detailed syntax without restricting the potential clients. Therefore all parts of an inscription representing a mathematical expression, besides indexed names, are referred to as 'text'. This includes denotations of mathematical operators, functions, constants and other variable names. In the presented version a formula is a single expression. It is possible to introduce compound expressions by prescribing some syntactic brackets (e.g. **begin** — **end** or << — >>) and allowing a sequence of assignment statements. In that way a number of local scalar variables would be introduced, with scope limited to the compound expression. However, such constructs would not merge well with the rules of the present members of the IAC-DIDAS family and are not considered in this paper. Similarly, it is possible to define several indexed variables within scope of one control sequence. This variant will be considered in later versions of software.

The definition of lower and upper bounds of an indexed variable (e.g. given after the system prompt) is the same as of a formula defining this variable, i.e. it can be an expression. All these definitions belong to the scope of the preceding control sequence. Physical units of an indexed variable belong formally to this scope too. However, as they are important only in specific model, they are assumed to be constant strings.

The limiting case of an indexed variable definition is the definition of a scalar variable. Any practical system must distinguish this case and handle it properly.

2.2 Definition of semantics

Now follows the definition of semantics. The definition described above is treated as a short-hand notation equivalent to some of scalar variable definitions. Their number results from the control structure part (*for_sequence*) of the definition of an *indexed_variable*. Their names, formulae and bounds are created by subsequent substitutions of the values of subscripts. Such process of creation is called in the following the expansion of an indexed variable. Below it will be described in more detail.

2.2.1 Generation of names of variables

New (scalar) variable names are generated from the name of an indexed name in the following manner:

— each index name is replaced by string of digits which denotes its value,

— in case of a negative value the minus character is replaced by the underscore character,

— the brackets are removed from the name.

Some comments are necessary here:

— A name generated in such a way may be identical to the name of some scalar variable of a model. Such conflicts can be resolved in many ways. The simplest one is to put the demand that prefixes of indexed names should be different from prefixes of all other variables. Such check could be made automatically. In the other strategy such restriction would be absent but the check would be made for possible conflicts. When a conflict would be found either some warning should be issued or automatic name correction (with asking for approval) should be done.

— In some cases the names generated along the above mentioned principles may be not unique, e.g. $ab[x,y]$ for $x = 2$, $y = 35$ and $x = 23$ and $y = 5$ will produce the same name $ab235$. Such situations can be detected automatically. In such case, either the user can be asked to split the two variable index segment and supply a *plain_name* separator in between e.g. $ab[x]\_[y]$, or a separator may be inserted automatically. The user can be asked (perhaps at the beginning of an editing session) for the text of such separator. Finally, separators may be inserted permanently. In such case the generated names are best readable but are also longer. This may lead to a conflict with some consumer software if it has some limitation of the allowable length of names of variables.

— The expressions appearing in index positions must evaluate to an integer constant. The evaluation must be performed in the described program. There is a danger, that the rules of priority and associativity of operators assumed in the program will be not the same as supposed by the user, especially priority of unary minus relative to other operators and the associativity of the power operator. This problem can eventually be resolved by producing parameterized parsing and calculating procedures. The user could then be asked by the program for his preferences or the user could have the possibility to modify the default settings as an option. In special cases the expression may just be a constant. This is the simplest way for introducing e.g. initial or boundary conditions.

— An optional check could be made, whether in every occurrence of an indexed variable, in the expression at given index position there appears the same set of control variables.

2.2.2 Generation of formulae and bounds

The indexed names appearing in the expressions defining formulae and bounds are treated according to the technique described above. Note that flexibility of control structures and lack of restrictions on the form of index expressions in the defined variable (i.e. on the'left side' of the 'assignment') allow for convenient definition of evolution processes in the form $a[t] = \cdots a[t-1] \cdots$ or $b[t+1] = \cdots b[t] \cdots$ or adjoint processes $c[t-1] = \cdots c[t] \cdots$ in the unique manner.

The remaining parts of the text denoting the expressions cannot be evaluated (at least in typical cases) but must be treated by symbolic manipulation methods. Two different strategies (and several intermediate ones) are possible here:

— The pattern-matching methods are used. In this case the pattern is the name of an index variable and the source is the text fragment of a formula definition. In classical pattern matching algorithms no structure analysis of the source is performed. Such a solution is not satisfactory in the presented application since an index name may accidentally be a part of another name. This situation can be resolved by the analysis of two characters adjacent to the found match. The found matches are simply replaced by the denotation of the actual value of an index variable.

— The methods of symbolic computation are used. In this case the text fragments are analyzed, transformed into appropriate structures. Matching elements of the structures are replaced and the resulting structure is simplified and converted back into text.

Each of these methods has its advantages and disadvantages. The symbolic method gives the result in neater form but, as all possible calculations are performed, is vulnerable to possible discrepancies in the understanding of properties of mathematical operators. In the first method, on the contrary, no calculations are performed so that the above mentioned problems do not exist. However the produced text is more complicated. Therefore, some mixed strategies can be thought of, e.g. with partial analysis of structure and performing only undoubtful calculations as e.g. the replacement of the phrase $+i*other\_variable*j+$ by $+k*other\_variable+$ where k denotes the value of $i*j$.

An expression denoting the value of an indexed variable and expressions denoting its bounds are treated differently. The first can only partially be evaluated while the latter must evaluate to a real value. Therefore only the names of local control variables (indexes) are allowed to appear in them.

2.2.3 Order of expansion

The ultimate goal of the expansion process is to produce a model, containing only scalar variables, which can be explicitly solved. This property may depend however on the order of substitution of indexes by their values. Two regular strategies for expansion were presented when presenting control structures. They are sufficient in simple cases. However in general more subtle patterns would be appropriate. This problem will be the subject of further investigations.

3 Applications of subscripted variables

The concept of subscripted variables can be used at several 'levels':

— It can be eventually be embedded into DSS systems to form a new generation of them. For the purpose of explanation let a subscripted variable be understood as time-dependent variable in a model of a dynamic process. If one visualizes a typical column of scalar outcome variables, then a subscripted outcome variable would be represented by one ore more entries in this column, which represent functionals defined on this process. The axis of time and eventually other spatial axes can be thought of as being perpendicular to this visible column — somewhere behind the screen. However, it would additionally complicate programs which are already rather sophisticated. The speed of calculations could be diminished since another level of indirection would be added in the access process to values of variables used by the solver.

— It can be used as a standard for model generation. Such an editor can be equipped with output generation modules (or independent specialized programs can be written) for producing output files which can be used as input files in another systems. The needed prerequisite is detailed knowledge of the format of these files. When it is not met, still a text file with definitions of scalar variables can be produced. It could be introduced to another system by hand. The potential areas of applications are:

  — linear static systems, if (at least significant part) coefficients can be expressed analytically,

  — linear dynamic systems,

— nonlinear dynamic systems.

The form of output may either be system specific file or some standard as e.f. MPS file.

— This concept can be used for very effective generation of series of greater and greater models. In the described implementation, it is sufficient to change initial and final values in control structures. However, one of fields of present investigations is the parameterization of the model by these items. With such device the 'growth' of the model would be almost trivial task. If several output modules would be available, the target software and the target computer could (ideally) be chosen depending on the model size.

4 Short program description

An experimental program was written to test the feasibility of ideas presented in this paper and experiment with various technical solutions. It supports the edition of variable (scalar and indexed) definitions and expansion of scalar variables. Definitions of variables read from a file or entered from the keyboard are transformed into dynamic structures. The process of expansion creates new structures, corresponding to variables of scalar type. The algorithm of Knuth-Morris-Pratt [1] was adapted to special needs of the program. Since the program is subject to changes and experiments, only the decription from the user's point of view is presented here.

The main screen shows the definition of a current variable. The control is achieved by means of the following control keys:

←↑↓→		— browse through the list of variables,
Home, Ctrl←		— move to the first variable on list,
End, Ctrl→		— move to the last variable on list,
Del		— delete current variable from list,
Ins		— insert new current variable. Program asks whether it should be inserted before or after current variable,
Enter		— edit current variable definition,
Alt **R**		— read model from disk,
Alt **W**		— write model to disk,
Alt **P**		— print model ,
Alt **E**		— expand current (subscripted) variable.
	↓↑	— browse through expanded variables,
	Alt **P**	— print expanded variables,
	Esc	— return to 'father' fariable,

Alt M	— merge models. Program asks whether the new read model should be inserted before or after current model,
Esc, F10	— quit.

5 Sample results

Here is a simple definition of an indexed variable:

```
for a:=2 to 3 do
 for b:=8 step 6 until 18 do
   define
     jp[a+3]_[b+1]:=  sin(a/(b+3.5))*jp[a-1]_[b-2]
     Units=km/h
     Lower bound=  -3*a/b
     Upper bound=  (a+b)^2
```

In the process of expansion the following variables are produced:

```
jp5_9 := sin(2/(8+3.5))*jp1_6
Units=km/h
Lower bound=  -3*2/8
Upper bound=  (2+8)^2

jp5_15 := sin(2/(14+3.5))*jp1_12
Units=km/h
Lower bound=  -3*2/14
Upper bound=  (2+14)^2

jp6_9 := sin(3/(8+3.5))*jp2_6
Units=km/h
Lower bound=  -3*3/8
Upper bound=  (3+8)^2

jp6_15:= sin(3/(14+3.5))*jp2_12
Units=km/h
Lower bound=  -3*3/14
Upper bound=  (3+14)^2
```

This example illustrates various substitution possibilities. Current values of control variables are substituted into index expressions *and evaluated* while in other places substituted *and not evaluated.* Another feature is also visible. The user can produce a sequence of variable definitions which do not form a consistent model. Algorithms for variables ordering, checking consistency etc. will be described in the following paper.

6 Conclusions

The introduced subscripted variables seem to be a convenient tool for the descriptions of models, especially dynamic ones. The experimental model editor proved the practical feasibility of the concept. It is believed that generators of model files for IAC-DIDAS-N, IAC-DIDAS-L1, IAC-DIDAS-L2 and eventually other systems will enlarge the scope of applications of the latter. The proposed notation is really concise and users may by tempted to generate really large models. Two limiting factors will then probably appear. The first one will be the computing power of available machines and the second — difficultiesin the interpretation of results. While the first item is beyond the scope of this contract, the second will be tried to deal with in the future.

References

[1] Knuth D.E., Morris J.H., Pratt V.R. (1977). Fast Pattern Matching in Strings, *SIAM J.Comp.*, Vol.6, pp.323-350.

[2] Kreglewski T., Paczynski J., Granat J., Wierzbicki A.P. (1988), IAC - DIDAS - N: A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models with Nonlinear Model Generator Supporting Model Analysis, WP-88-112, IIASA.

[3] Lewandowski A., Wierzbicki A.P.(1990), Decision Support Systems Using Reference Point Optimization. In: A.Lewandowski, A.P.Wierzbicki (Eds), Aspiration Based Decision Support Systems, Springer, Lecture Notes in Economics and Mathematical Systems, Vol. 331, pp.3-20.

Interactive Graphics for Aspiration Based Decision Support Systems

Janusz Granat
Institute of Automatic Control
Warsaw University of Technology
00665 Warsaw, Poland

1 Introduction

This report presents some graphic tools for improving user interface for aspiration based decision support systems. Recently, the methodology of such systems have been advanced [11] and several systems implementing this methodology have became available [9]. The interactive scanning of efficient outcomes is one of the phases of work with those systems. This phase consist of the following steps:

1. Specification of reference levels,
2. Computation of corresponding efficient solutions (results),
3. Evaluation of results by decision maker and return to the first step.

The existing experience of working with DIDAS systems [11] shows that scanning of efficient solutions should be improved. Among other approaches, graphic methods can be used for developing this phase. There are three different aspects of graphical information presentation in Aspiration Based Decision Support Systems [10]:

- The presentation of results in graphic form to improve understanding of these results as well as of relationships between components of the decision problem being solved (objectives, decision variables etc.) ,
- Providing support for dynamic interaction between user and Decision Support Systems during specification of reference levels. This function contributes to the learning process by allowing an analysis of the history of interaction which has resulted in achieving the current state of the decision making process.
- A possible use of graphic interaction during the specification of reference levels, together with an interpretation of multiple reference levels (aspiration and reservation, possibly other levels) in terms of fuzzy set theory.

The paper concentrates on two first aspects listed above.

2 Formulation of the decision making problem

The decision making problem will be formulated as multiobjective optimization problem:

$$\max_{x \in X} \underline{f}(\underline{x}), \qquad \underline{x} \in X \tag{1}$$

where

$\underline{x}$ - is the vector of decision variables,
$\underline{f}$ - is the vectors of objectives to be minimized,
X - is a set of admissible decisions.

An optimal solution that maximizes all the objective functions simultaneously does not necessarily exist. Therefore the concept of efficiency or Pareto optimality [16, 17] is used in multiobjective optimization and the final decision is usually made among the set of efficient solutions. The set of efficient solutions can be searched by using interactive procedures. Although there is a wide variety of such procedures, the reference point approaches are often used. The user can specify two kinds of reference levels: aspiration levels and reservation levels for all objectives. The aspiration levels represent the levels that user would like to attain, whereas reservation levels cold be interpreted as 'soft' lower limits for objectives. An achievement function that can be interpreted as a specific utility function explicitly dependent on the aspiration and reservation levels can be constructed[11].
An example of such a function is:

$$s(q, \overline{q}, \overline{\overline{q}}) = (\min_{1 \le i \le p} z_i + \frac{\rho}{p} \sum_{i=1}^{p} z_i)/(1 + \rho)$$

$$z_i = \begin{cases} \gamma((q_i - q_{i,min})/(\overline{q}_i - q_{i,min}) - 1) & \text{if } q_i \le \overline{q}_i \\ (q_i - \overline{q}_i)/(\overline{\overline{q}}_i - \overline{q}_i) & \text{if } \overline{q}_i < q_i < \overline{\overline{q}}_i \\ \beta(q_i - \overline{\overline{q}}_i)/(q_{i,max} - \overline{\overline{q}}_i) + 1 & \text{if } \overline{\overline{q}}_i \le q_i \end{cases}$$

$0 < \rho \le p$
$\beta \ge 0,\ \gamma > 0$
$\overline{q}_i$ - reservation level
$\overline{\overline{q}}_i$ - aspiration level
ρ - weighting coefficient
p - number of objectives

An efficient solution that strictly corresponds to the user-specified reference levels can be calculated by minimizing this function. In order to provide the user with an information about reasonable ranges of efficient outcomes, the utopia and nadir points can be computed. The utopia point is the result of maximizing each objective separately. The nadir point is the estimation of the lower bounds on efficient outcomes.

There are several methods of organizing interaction with the user:

1. The user can scan efficient solutions by changing reference levels and then he selects a proper decision. This process can be supported by a graphic comparison of the results obtained thus far. This method has been implemented in systems of DIDAS family.

2. Sometimes, the user might wish to learn more about the shape of the efficient frontier by looking at various cuts through the efficient frontier. After observing several such cuts the user makes a proper decision. This method has been used e.g. by Korhonen [6].

3. Some additional data or dependencies can be displayed for the user in order to provide information about a reasonable direction of changing of reference levels (e.g. biplot method proposed in [10]).

We can thus distinguish several basic data (vectors), which can be presented to the user in various form (e.g. numerically, graphically):

- vectors of aspiration levels;
- vectors of reservation levels;
- utopia point;
- nadir point;
- efficient solutions (results);
- upper bounds;
- lower bounds.

The next two chapter describes some applications of the graphic methods to visualize such data and information.
The following example will be used for demonstration of graphic methods described:
Example 1.
The multiobjective nonlinear programming problem is formulated as follows:

$$\begin{aligned} \max obj1 &= x_1 \\ \max obj2 &= x_2 \\ \max obj3 &= x_3 \end{aligned}$$

Bounds on variables and outcomes:

$$\begin{aligned} 0 \leq x_1^2 + x_2^2 + x_3^2 \leq 1 \\ 0 \leq x_1 \leq 1 \\ 0 \leq x_2 \leq 1 \\ 0 \leq x_3 \leq 1 \end{aligned}$$

3 Presentation Graphics

In the phase of scanning efficient solutions, the decision maker should evaluate results obtained thus far and specify new reference levels. This work can be supported by displaying the following information:

- comparison of results;
- cuts through the efficient frontier;
- some additional information (e.g. trajectories for dynamic problems)

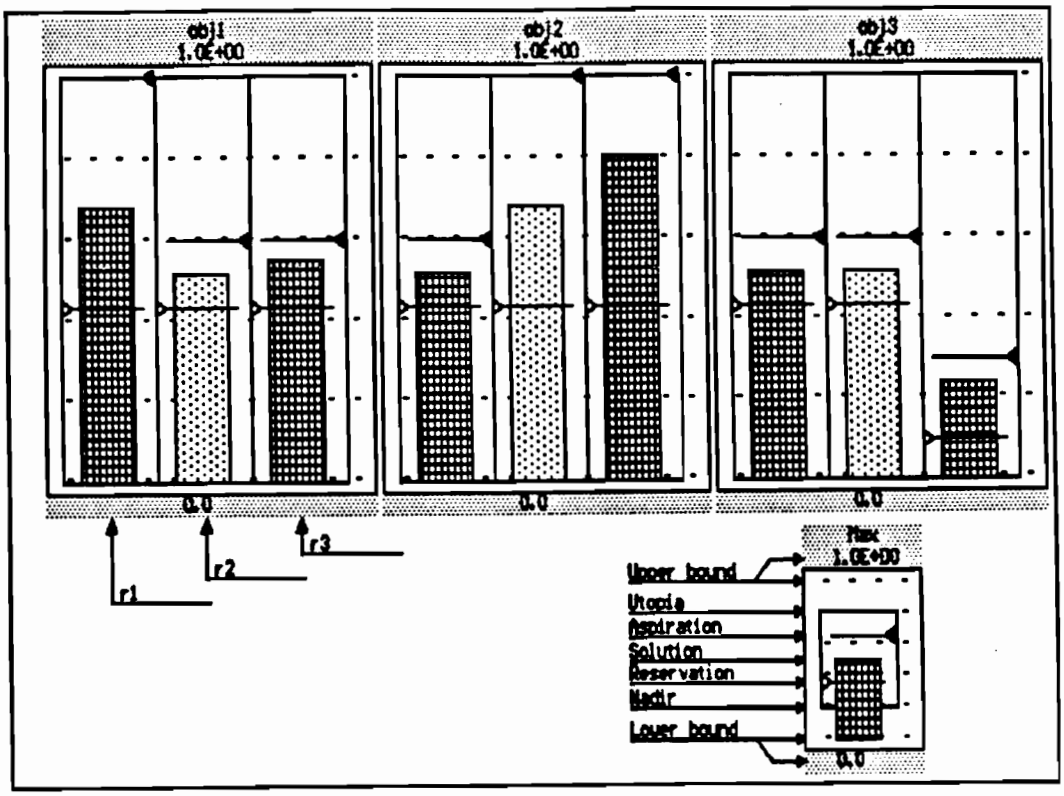


Figure 1: Comparison of results

3.1 Comparison of results

In IAC-DIDAS-N several results and several objectives can be displayed simultaneously in graphical bar form. Additionally other basic data are displayed. Fig.1. presents three successive results obtained during the process of the interaction. The numerical results are presented in Table 1.

name of the result	name of the objective	aspiration level	solution	reservation level
r1	obj1	1.000	0.683	0.436
	obj2	0.600	0.516	0.436
	obj3	0.600	0.516	0.436
r2	obj1	0.600	0.516	0.436
	obj2	1.000	0.683	0.436
	obj3	0.600	0.516	0.436
r3	obj1	0.600	0.548	0.436
	obj2	1.000	0.801	0.436
	obj3	0.300	0.240	0.100

Table 1

The results for all objectives are displayed in the form of bars in the frames described by the names of the objectives. Oldest result appears on the left side. The most recent result is displayed on the right side. A thin rectangle marks the range between the nadir and the utopia points. The aspiration and reservation levels are marked by a lines and a triangles. The triangles which marks aspiration levels are filled.

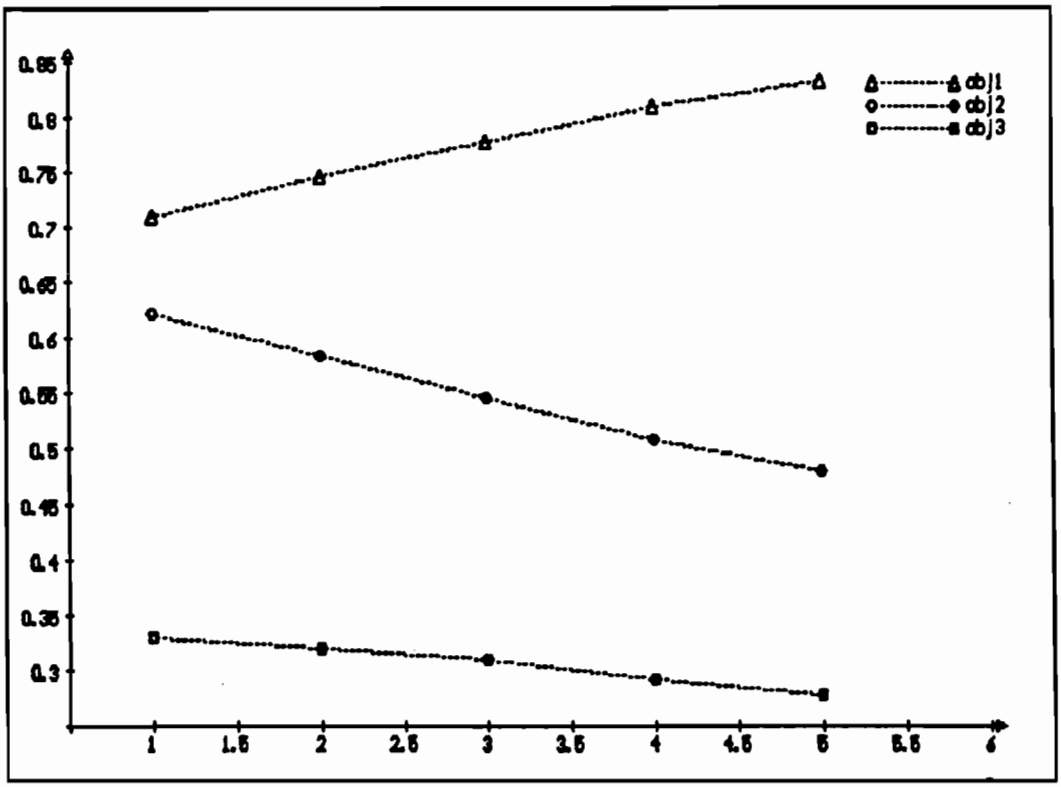


Figure 2: A directional scan of efficient outcomes

3.2 Directional scan of the efficient outcomes

The available systems of DIDAS family respond with only one efficient solution to each specification of reference levels by the user. Sometimes, however, he might wish to learn more by looking at various cuts through the efficient frontier that can be computed by the system and displayed graphically. If the number of objectives is small, a very useful information is the graph $q_i = f(q_j)$ (where q_i, q_j- are two selected objectives). Various procedures can be applied to generate such cuts (see [12, 13, 15, 1]). This representation can be also calculated by a more general procedure of the multidimensional scan described below. The system can produce a cut through the efficient frontier approximately along a ray starting from an efficient outcome point $\overline{q}^{bas}$ in a direction δ specified by the user. In such a case, the additional reference points are determined by:

$$\overline{q}(k) = \overline{q}^{bas} + \frac{k}{N}\tau\delta, \qquad 1 \leq k \leq N \tag{2}$$

where

τ - a step-size coefficient
δ - a direction in the objective space
N - a number of optimization computations

In the new experimental version of IAC-DIDAS-N, directional scan of the efficient outcomes is implemented. Fig.2. has been obtained for the Example 1. The following data have been used during the calculation:
$\overline{q}^{bas} = (0.5, 0.5, 0.5), \quad \delta = (1, -1, 0), \quad \tau = 0.05, \quad N = 5$

3.3 Display of trajectories for dynamic problems

Sometimes the system should show for the user other dependencies specific for the problem being solved. Let us consider the following example of the deterministic formulation of

the flood control problem for one water reservoir[4]:

$$\min \quad \max_{k=1,\ldots,T} u(k) \tag{3}$$

$$\max \quad w(T) \tag{4}$$

$$w(k+1) = w(k) + d(k) - u(k) \tag{5}$$

$$w(1) = w_1 \tag{6}$$

$$w_{\min} \leq w(k) \leq w_{\max}, \qquad k = 1,\ldots,T \tag{7}$$

$$u_{\min} \leq u(k) \leq u_{\max}(w(k)), \quad k = 1,\ldots,T \tag{8}$$

where:

w - the level of water stored in the reservoir
d - water inflow to the reservoir
u - water release from the reservoir
k - current period of time
T - the control horizon (we assume $T = 10$)

This problem has been solved by IAC-DIDAS-L2, which accepts the following formulation of a multiobjective linear programming problem:

$$\min q = Cx \tag{9}$$

with the set of admissible decisions:

$$X = \{x \in R^n : x^{lo} \leq x \leq x^{up}; y^{lo} \leq Ax \leq y^{up}\} \tag{10}$$

where:

q - objectives
x - vector of decisions variables
x^{lo}, y^{lo} lower bounds
x^{up}, y^{up}- upper bounds

After reformulation of the problem (3),...,(8) to the IAC-DIDAS-L2 acceptable form we have the following vector of decision variables:

$$x = [u(1), u(2), \ldots, u(9), w(2), w(3), \ldots, w(10), z] \tag{11}$$

where z is an artificial variable

$$u(k) \leq z, \quad k = 1,\ldots,T \tag{12}$$

We can see that the vector of the decision variables actually consist of 9 points of release trajectory and of 9 points of storage trajectory. Together, such trajectories give useful information (Fig. 3) to the users of DSS, who deal with flood control problems.

4 Interactive graphics

4.1 A short review of existing systems

Interactive graphics is one of the most powerful tools for designing user interfaces. Some systems for multicriteria decision making utilizing interactive graphics have been built but only three of them were based on aspiration level methodology:

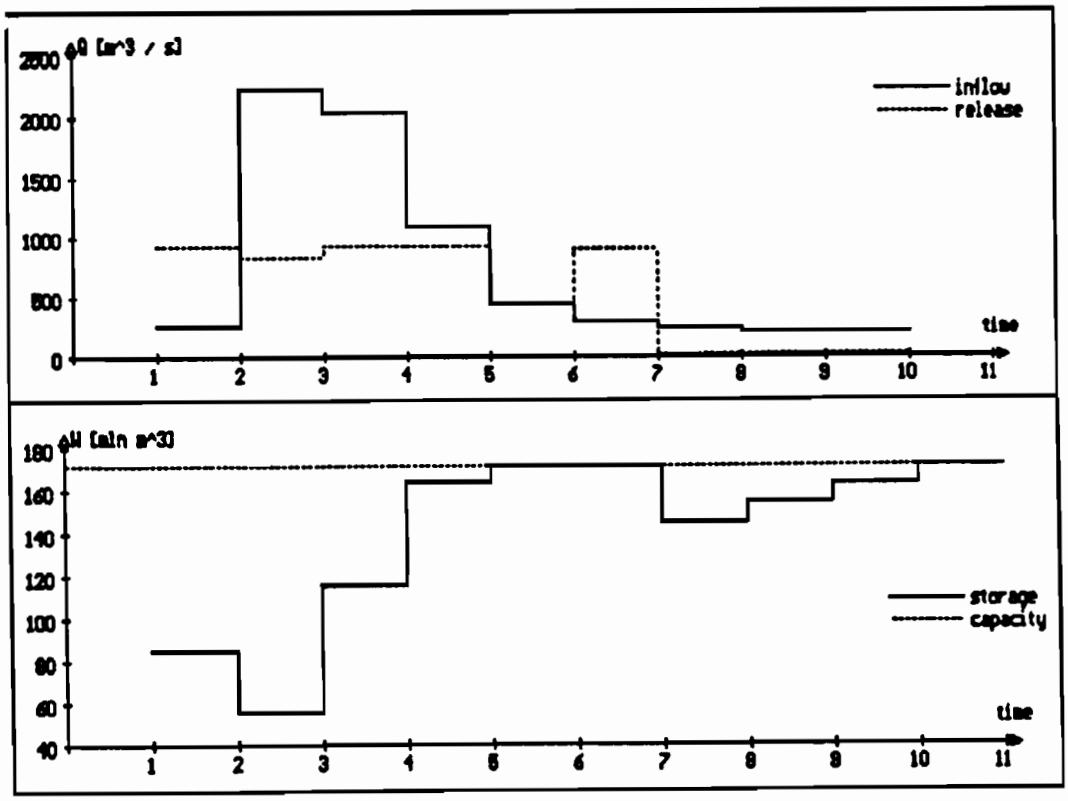


Figure 3: Some trajectories for the flood control problem

VIG (Korhonen 1986). A Visual Interactive Approach to Goal Programming. A visual interactive method [7] has been used for solving linear multicriteria mathematical programming problems. The idea of this method is to iteratively project an unbounded line segment in the criterion space onto the efficient surface. This projection is obtained by using an achievement scalarizing function (13) and a parametric programming problem of minimizing this function:

$$s(q, z, w) = \max_{i \in I}(z_i - q_i)/w_i, \quad I = \{i | w_i > 0\} \tag{13}$$

$$z = q^{k-1} + td^k, \quad q \in Q = \{f(x) | x \in X\} \tag{14}$$

w - is a weighting vector
z - is an arbitrary vector in criterion space
q - is a vector of objective function values
t - parameter t is increased from 0 to infinity

The VIG system uses a so-called Pareto Race method to find the most preferred solution. In Pareto Race the user can control the speed and the direction of scanning of the Pareto set. The current solution is presented in a horizontal bar chart form.

VIMDA (Korhonen 1986) A Visual Interactive Method for Discrete Alternatives. This system can be used for solving discrete deterministic multicriteria problems. The theoreticalfoundations of the VIMDA are the same as for VIG. The values of the criteria of the alternatives corresponding to a specified by the user reference direction are plotted on the screen. The values of the criteria of the subsequent alternatives are connected with a line. The graphic cursor can be moved to any point on the curve and the corresponding numerical values of objectives are displayed. The decision maker can choose the most preferred alternative among the alternatives presented at this display.

IAC-DIDAS-L2 (Rogowski, Sobczyk and Wierzbicki 1988) It is a system for interactive, multiobjective analysis for models describing substantive aspects of the decision situation that are represented in linear programming form. The reference point optimization theory has been applied. In this system, it is possible to modify reference point values and run single optimization in the graphic mode. Solutions are displayed in the form of bars similar to these on the Fig.1. The reference point can be set by moving the marker up and down on the right hand side of the bar. Then the optimization can be started. After calculation, new bars representing the newly obtained solution appear on the right side of the frames described by the names of objectives.

4.2 Dynamic BIPLOT as an Interaction Interface

A convenient technique for graphical data analysis has been proposed by Gabriel[2]. A $n \times m$ matrix Y of data is considered. In our case, this matrix might represent some set of nondominated solutions with columns corresponding to objectives and rows corresponding to subsequently generated solutions. If Y is of rank r, it can be expressed as the product of an $n \times r$ matrix G and $r \times m$ matrix H':

$$Y = G H' \tag{15}$$

A element y_{ik} would be expressed as:

$$y_{ik} = g_i h'_j \tag{16}$$

In other words, the vectors $g_1, \ldots, g_n$ are assigned one to each row of Y and the vectors $h_1, \ldots, h_m$ one to each column of Y. The most helpful visual representations will be obtained when $r = 2$, since the vectors g and h can be plotted then on a plane giving a representation of the elements of the matrix Y. Such a plot is called a biplot since it allows the rows and columns to be plotted jointly. Matrices of ranks higher than two cannot be represented exactly by a biplot. If a matrix Y can be satisfactorily approximated by a matrix $Y_{[2]}$ of rank two, the biplot of $Y_{[2]}$ may be a useful approximate visual representation of the matrix Y.

The best-known method for lower-rank approximation is a method based on singular value decomposition theory [3]. We want to find a matrix $Y_{[2]}$ of rank 2 which minimizes:

$$\| Y - Y_{[2]} \|^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} (y_{ij} - y_{[2]ij}) \tag{17}$$

The matrix Y can be represented using the singular value decomposition as follows:

$$Y = \sum_{\alpha=1}^{r} \lambda_\alpha p_\alpha q'_\alpha \tag{18}$$

where λ_α denotes the singular values of the matrix Y, p_α denotes the singular column and q_α denotes the singular row satisfying the following relations:

$$\begin{aligned}
p'_\alpha Y &= \lambda_\alpha q'_\alpha, && (19)\\
Y q_\alpha &= \lambda_\alpha p'_\alpha, && (20)\\
Y Y' p_\alpha &= \lambda_\alpha^2 p_\alpha, && (21)\\
Y Y' q_\alpha &= \lambda_\alpha^2 q_\alpha, && (22)\\
\lambda_1 \geq \cdots \lambda_r &\geq 0, && (23)\\
p'_\alpha p_\epsilon = q'_\alpha q_\epsilon &= \delta_{\alpha,\epsilon}, && (24)
\end{aligned}$$

where $\delta_{\alpha,\epsilon}$ is the Kronecker's delta. Any solutions of a pair of equations (19) and (20), (19) and (21) or (20) and (22) will satisfy the remaining equations. A last squares approximation to Y by $Y_{[2]}$ is:

$$Y_{[2]} = \sum_{\alpha=1}^{2} \lambda_\alpha p_\alpha q'_\alpha \tag{25}$$

The absolute measure of the goodness of fit of the above approximation can be expressed as:

$$\rho_s^2 = \frac{\sum_{\alpha=1}^{2} \lambda_\alpha^2}{\sum_{\alpha=1}^{r} \lambda_\alpha^2} \tag{26}$$

Let as assume that the matrix Y is normalized, i.e. mean value of each objective has been subtracted from the corresponding column of the matrix Y, then

$$S = \frac{1}{n} Y'Y \tag{27}$$

can be interpreted as the variance-covariance matrix. Let us consider the rank 2 approximation of matrix Y

$$Y_{[2]} \sim Y \tag{28}$$

and let λ_1, λ_2 denote two largest singular values. Than the matrix $Y_{[2]}$ can be factorized as:

$$Y_{[2]} = GH' \tag{29}$$

$$G = (p_1, p_2)\sqrt{n} \tag{30}$$

$$H = \frac{1}{\sqrt{n}}(\lambda_1 q_1, \lambda_2 q_2) \tag{31}$$

Therefore, if we consider such a definition of the matrix $Y_{[2]}$, the following conclusions can be derived from the general properties of the singular values decomposition and factorization:

$$Y \sim GH' \tag{32}$$

$$YS^{-1}Y' \sim GG' \tag{33}$$

$$S \sim HH' \tag{34}$$

where $\sim$ stands for is approximated by means of a last squares fit of rank two. Any approximate biplot of Y gives the following approximations of the values of matrix Y:

$$y_{ij} \sim g'_i h_j, \tag{35}$$

as well as of covariances, variances and correlation coefficients

$$s_{j,g} \sim h'_j h_g \tag{36}$$

$$s_j^2 \sim \| h_j \|^2 \tag{37}$$

$$r_{j,g} \sim \cos(h_j, h_g) \tag{38}$$

Moreover, the Euclidean distance between vectors g_i approximates the standardized (Machalanobis) distance between observations:

$$d_{i,j} \sim \| g_i - g_j \| \tag{39}$$

where

$$d_{i,j} = (y_i - y_j)' S^{-1} (y_i - y_j) \tag{40}$$

Lewandowski and Granat(1989) presented a technique for using BIPLOT for building a graphical interactive user interface. Especially important for using biplot as the interactive interface are the following properties:

- elements of data matrix can be approximated as orthogonal projections of vectors h on principal component directions,
- correlation coefficients between variables are approximated by the angle between vectors h,
- the variance of a variable is approximated by the length of a corresponding vector h.

The BIPLOT interface has been integrated with IAC-DIDAS-L2 and IAC-DIDAS-N. Some further possibilities of using biplot for modifying aspiration levels can be considered:

1. We assume that we have computed an efficient solution, corresponding to a given vector of aspiration levels.
2. We calculate N efficient solutions by perturbing the given aspiration level ($N \geq M$, where M is the number of objectives) to obtain an initial matrix Y.
3. We compute the biplot approximation and display the biplot screen, see Fig.4. Then we can move a cursor (using mouse or keyboard); simultaneously a point representing the cursor is projected on directions generated by h vectors thus giving values of objectives corresponding to this point. The numerical values of objectives are displayed on the left side of the screen. If the approximated values of objectives are satisfactory, we can press a mouse button and the values of objectives will be moved to the spreadsheet as a new vector of aspiration levels, see Fig.5.
4. We can calculate a new efficient objective outcome corresponding to the new vector of aspiration levels.
5. If the new solution is not satisfactory, we can repeat the action from the point 2 or 3. If we chose point 2, the results used previously will not be displayed. If we chose point 3, the new result will be added to the previous ones and to biplot screen.

A numerical example

The results has been obtained for Example 1.

1. Initial efficient solution: (0.542, 0.542, 0.542) .
2. The efficient solutions obtained by perturbing aspiration levels are presented in Table 2.

name of the result	name of the objective	aspiration level	solution
r1	obj1	1.000	0.642
	obj2	0.718	0.542
	obj3	0.718	0.542
r2	obj1	0.718	0.542
	obj2	1.000	0.642
	obj3	0.718	0.542
r3	obj1	0.718	0.642
	obj2	0.718	0.642
	obj3	1.000	0.542

Table 2

The following matrix Ycan be built from the obtained results:

$$Y = \begin{bmatrix} 0.576 & 0.576 & 0.576 \\ 0.642 & 0.542 & 0.542 \\ 0.542 & 0.642 & 0.542 \\ 0.542 & 0.542 & 0.642 \end{bmatrix} \tag{41}$$

The biplot of matrix Y is presented on Fig.4.

3. In this step we chose the aspiration levels: (0.514, 0.603, 0.611)
4. The new outcome solution is (0.521, 0.599, 0.607)
5. We assume that the new solution is satisfactory.

4.3 Specification of the reference levels in terms of fuzzy set theory

Another promising option may be an application of fuzzy sets theory to the specification of reference levels. The user can define his reference levels by specifying the membership functions for each of the objectives. This process may be supported by graphic tools. Fig.6. presents a proposal of an interaction window. Moving points marked by circles the membership functions can be changed. The membership functions in Fig.6. have the following simple analytical form:

$$m_i = \begin{cases} 0 & \text{if } q_i \leq q_i' \\ (q_i - q_i')/(q_i'' - q_i') & \text{if } q_i' < q_i < q_i'' \\ 1 & \text{if } q_i'' \leq q_i \end{cases}$$

More complicated forms of membership functions can be also treated in e similar way.

5 Conclusions

This paper shows that there are many approaches to building man-machine interfaces in decision support systems. Some of the users might prefer one of the methods of interaction,

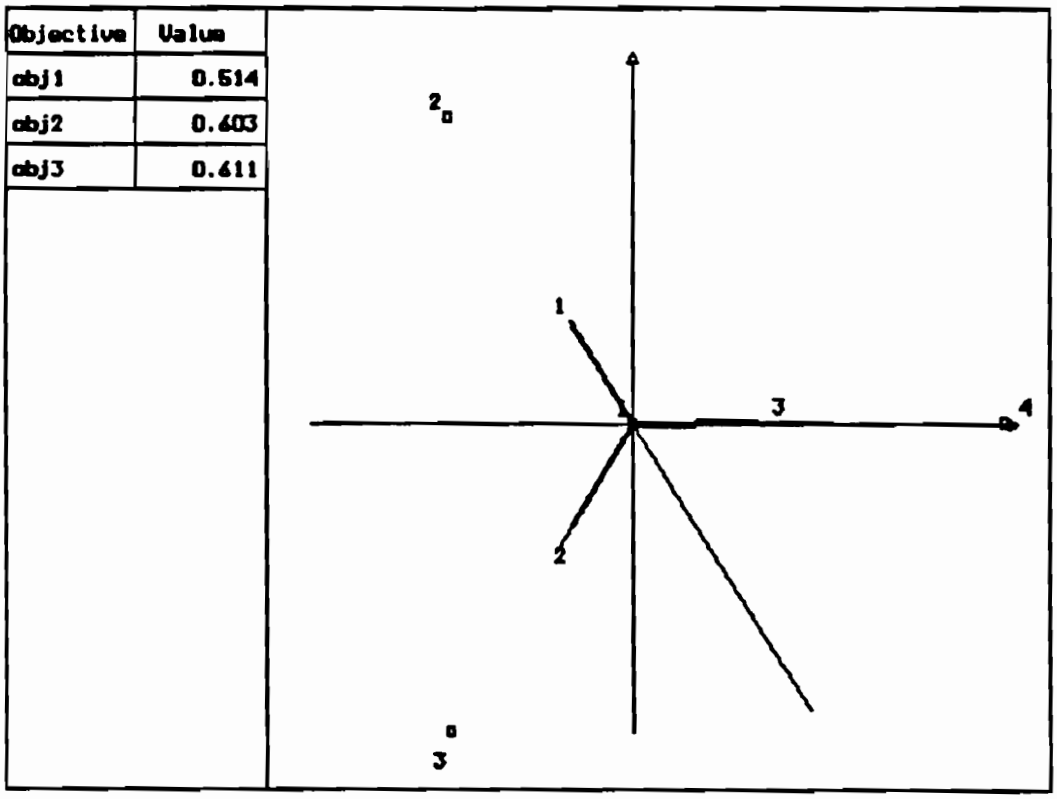


Figure 4: The biplot interaction screen

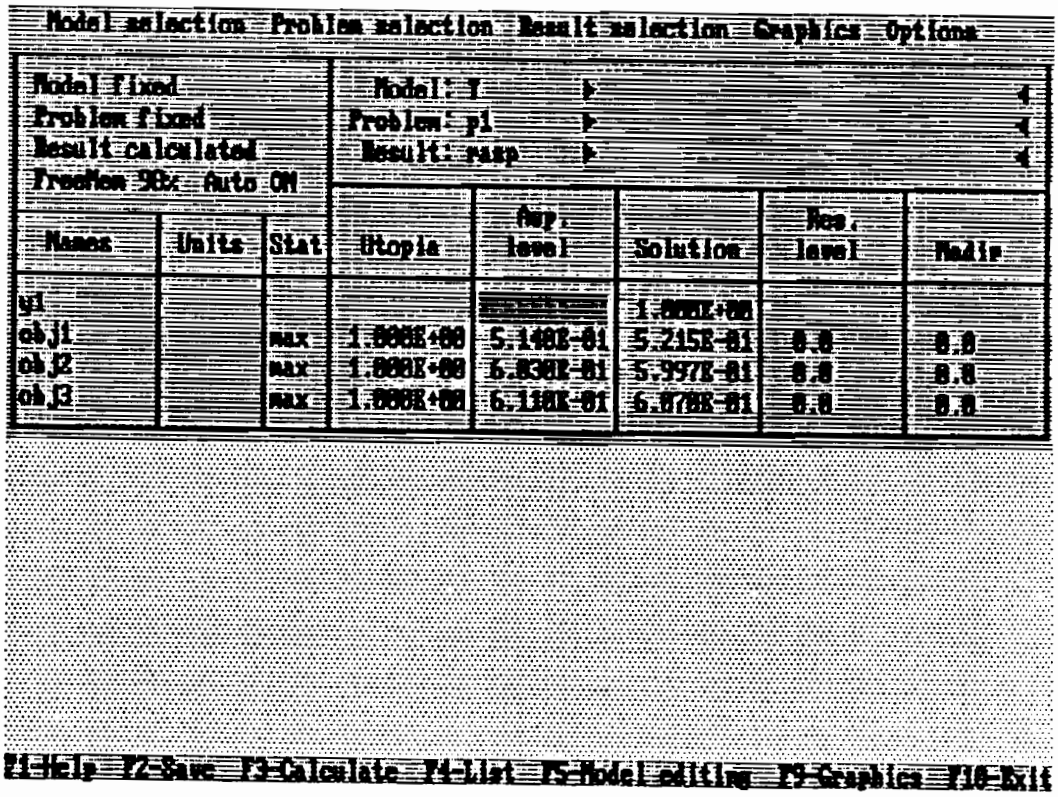


Figure 5: IAC-DIDAS-N spreadsheet

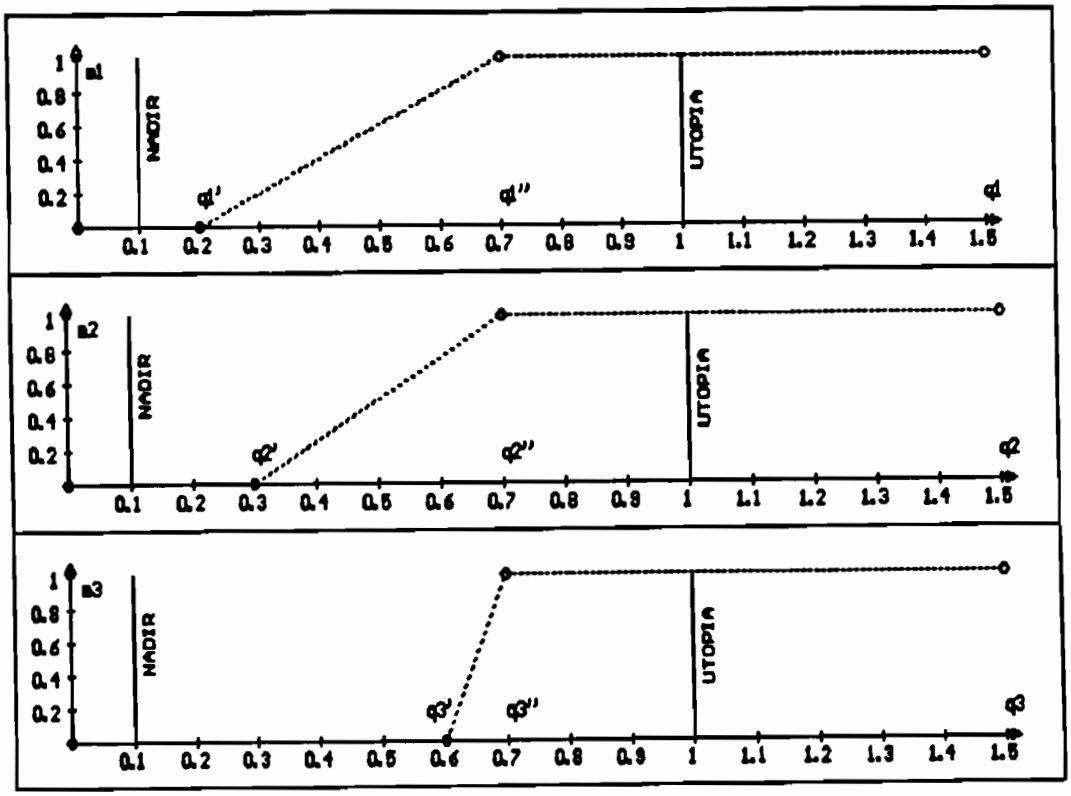


Figure 6: Membership function for objectives

an other might prefers quite a different method. Experience in building such systems shows that it will be very useful to develop the system, which could provide various interaction tools for the user. A special attention should be paid to the design of an architecture of such a system and to the principles of system's components. This paper is a first step in this direction. Section two describes the elementary data, that are important from an interaction point of view. The next sections how this data can be presented for the user and how the interaction can be organized.

References

[1] Fruhwirth B.,Burkhard R.E.,Rote.G.(1989) Approximation of convex curves with application to bicriterial minimum cost flow problem, EJOR, 42, p326-338.

[2] Gabriel K.R. (1971) The biplot graphic display of matrices with application to principal component analysis, Biometrica, 58, 33, p453.

[3] Golub G.H. Reinsch C. (1970) Singular value Decomposition and Last Squares Solutions, Numerische Mathematik 14, p403-420.

[4] Karbowski A. (1990) Application of IAC-DIDAS-L2 to Optimization of Water Releases From a Retention Reservoir During Flood, In preparation.

[5] Korhonen P.J.,(1986) VIG, User's Guide.

[6] Korhonen P.J. (1986) Solving Discrete Multiple Criteria Problems by Using Visual Interaction in Fandel G. at. al. (Eds.) Large Scale Modelling and Interactive Decision Analysis, Springer, LNEMS, 273.

[7] Korhonen P.J., Laakso J.(1986) A visual interactive method for solving the multiple criteria problem, EJOR 24, p277-278.

[8] Kreglewski T., Paczynski J., Granat J., Wierzbicki A.P. (1988), IAC - DIDAS - N: A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models with Nonlinear Model Generator Supporting Model Analysis, WP-88-112, IIASA.

[9] Lewandowski A. (1988), Short Software Descriptions, WP-88-109, IIASA.

[10] Lewandowski A., Granat J. (1989), Dynamic BIPLOT as the Interaction Interface for Aspiration Based Decision Support Systems, International Workshop on Multiple Criteria Decision Support, 6-10 Aug 1989, Helsinki.

[11] Lewandowski A., Wierzbicki A.P. (Eds.) (1990), Aspiration Based Decision Support Systems, Springer, LNEMS 331.

[12] Payne H.J. Polak E., Collins D.C, Meisel W.S. (1975), An Algorithm for Bicriteria Optimization Based on the Sensitivity Function., IEEE Transactions on Automatic Control,August 1975.

[13] Polak E. (1975) On the approximation of solutions to multiple criteria decision making problem, in Beckman M., Kunzi H.P (Eds), Multiple Criteria Decision Making, Kyoto 1975.

[14] Rogowski T., Sobczyk J., Wierzbicki A.P., (1988) IAC-DIDAS-L Dynamic Interactive Decision Analysis and support system linear version, WP-88-110, IIASA.

[15] Solanki R.S., Cohon J.L. (1989) Approximating the noninferior set in linear biobjective programs using multiparametric decomposition, EJOR, 41, p355-366.

[16] Sawaragi Y., Nakayama H., Tanino T. (1985), Theory of Multiobjective Optimization, Academic Press.

[17] Steuer R.E.(1986) Multiple Criteria Optimization: Theory, Computation and Application, John Wiley and Sons.

Part II
Example of Prototype Decision Support Systems (DSS) and Applications

I A C - D I D A S - N
A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models Version 4.0

T. Kreglewski, J. Granat, A. P. Wierzbicki

Institute of Automatic Control, Warsaw University of Technology

Abstract

This paper presents introductive and user documentation — including extended summary, theoretical manual, short user manual and description of illustrative examples — for a version of decision analysis and support systems of DIDAS family that is designed for multicriteria analysis of nonlinear models on professional microcomputers. This version has been developed in the years 1986–1990 in the Institute of Automatic Control, Warsaw University of Technology, under a joint research program with the Systems and Decision Sciences Program of the International Institute for Applied Systems Analysis. It can be run on professional microcomputers compatible with IBM--PC--XT or AT (with Hercules Graphics Card, Color Graphics Adapter or Enhanced Graphics Adapter and, preferably, with a numeric coprocessor and a hard disk) and supports graphical representation of results of interactive multicriteria analysis. Moreover, this version called IAC-DIDAS-N is provided with a new nonlinear model generator and editor that support, in an easy standard of a spreadsheet, the definition, edition and symbolic differentiation of nonlinear substantive models for 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. Optimization runs involved in interactive, multiobjective decision analysis are performed by a solver, that is, a version of nonlinear programming algorithm specially 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 was further developed and several improvements were added. The system is permanently updated and developed. Currently (starting from October 1990) the version 4.0 of the system is released. Most of enhancements added in this version are not directly visible to the user. They influence the efficiency of the system.

1 Extended summary

In many complex decision problems involving economic, environmental and technological decisions as well as in complex engineering design, decision maker needs some 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 situation that can be formalized mathematically and computerized. Although 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 thus 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 (Dreyfus, 1984) — the interaction of the decision maker and the team of analysts with 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, 1987) — are specially designed to support interactive work with a substantive model while using multicriteria optimization tools, but they stress the learning aspects of the work, such as the right of a decision maker to change his priorities and preferences after learning new facts. DIDAS systems can be used either by analysts who want to analyze 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 — see (Lewandowski et al., 1987). The IAC-DIDAS-N version is designed to support models of multiobjective nonlinear programming type. Although some nonlinear DIDAS versions were 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 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 may be changed during parametric or sensitivity analysis) — and *outcome variables* that can be subdivided into *floating outcomes* (used either as model constraints or only for the easiness of definition of the nonlinear model or even only as additional information 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 as the user may 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.

There are many examples of decision problems that can be analyzed by the use of a substantive model of multiobjective nonlinear programming type; for example, DIDAS-type systems with multiobjective nonlinear programming models were used in analyzing 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 of acid deposition in forest soil (see Hettelingh and Hordijk, 1987). The user can also define substantive models of multiobjective nonlinear programming type for his own problems and analyze them with the help of IAC-DIDAS-N.

A typical procedure of working with the IAC-DIDAS-N system consists of several phases. In the first phase a user, mostly 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 was not 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 derivatives analytically; thus, this way of substantive model preparation 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, lower and upper bounds on all variables, as well as reference levels (reservation levels for stabilized outcomes, aspiration and reservation levels for maximized and minimized outcomes) and 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 needn't laboriously calculate many derivatives and check whether he has not made any mistake; he must only define model equations or outcome functions (whereas a recursive, but explicit form of such functions is allowed) and make sure that these functions are differentiable and admissible for the symbolic differentiation program — that admits functions from a rather wide class. Moreover, the spreadsheet format allows also to display the automatically determined formulae for derivatives. The size of substantive models that can be defined in the spreadsheet is limited only by the size of microcomputer memory, but reasonable models of nonlinear programming type that can be usefully analyzed on microcomputers should not be too large anyway. The user of IAC-DIDAS-N can also have several substantive models recorded in special model directories, use old models 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 appropriate directory.

In further phases of the 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 may be many multiobjective analysis problems related to the same substantive model. The specification of a multiobjective problem consists in designating optimized outcomes (objectives) among 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 outcome when specifying a multiobjective analysis problem.

For a given definition of the multiobjective analysis problem, the decisions and out-

comes in the model are subdivided into two categories: these 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 the main functions of DIDAS-type systems is computation of efficient decisions and outcomes — interactively following various instructions of the user — and their presentation 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 experience 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.

A multiobjective problem definition usually admits many efficient decisions and outcomes; the user should first learn about *ranges* of changes of outcomes and *bounds* on *efficient outcomes*. Calculations of these bounds 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 is also special command that automatically performs all necessary calculations.

The command "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 point* for the point in outcome space composed of such outcomes). During "utopia" calculations some approximations of worst possible efficient values are also obtained. The point in outcome space composed of the worst efficient values is called *nadir point*, however its exact calculation is a very difficult computational task — for nonlinear models there is even no constructive method for such calculation. The approximation of nadir point components obtained during utopia point calculations is rather too optimistic. The decision maker or the analyst can change, according to their knowledge, obtained nadir values.

The utopia and nadir points give important information to the user about reasonable ranges of (efficient) decision outcomes; in order to give him also information about a reasonable compromise efficient solution, a *neutral efficient solution* can also be computed in the initial analysis phase due to a special command. The neutral solution is an efficient solution situated 'in the middle' of the range of efficient outcomes; the precise meaning of being 'in the middle' is defined by the distances between the utopia and (the approximation of) the nadir point components. After analyzing the utopia point, the nadir point and the 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 trade-off between these objectives. Each change of the definition of the substantive model or of the multiobjective analysis problem, however, actually necessitates a repetition of the initial analysis phase.

The third phase of the work with the IAC-DIDAS-N system consists in interactive scanning of efficient outcomes and decisions, guided by the user who specifies two *reference points* called *reservation point* and *aspiration point* in the objective space, i.e. *reservation levels* and *aspiration levels* for each objective; the system admits also for a more simple option of specifying only one reference (aspiration or reservation) level for

some or even for all objectives. The user already has reasonable knowledge about the range of possible outcomes and thus, he can specify his reference levels: aspiration levels that he would like to attain and reservation levels that he would like to satisfy in any case. The utopia and the nadir points could be used as initial values for the aspiration point and the reservation point, respectively. However, because the neutral solution has also been calculated, the system suggests to the user another, more adequate initial aspiration point: an unattainable outcome point closer to the efficient solutions than the utopia point, and more adequate initial reservation point: an attainable outcome closer to the efficient solutions than the nadir point.

IAC-DIDAS-N utilizes the aspiration and the reservation levels as parameters 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 references.

If the aspirations are not attainable and the reservations are attainable (that is a typical and recommended case), then the response of the system is a solution with attainable, efficient outcomes that are either between the aspiration and reservation points or uniformly as close as possible to the former one. 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. If the reservations are 'too high' (if they correspond to outcomes that are not attainable), then the response of the system is an efficient solution with outcomes that are uniformly worse than the reservations. The precise meaning of the uniform approximation or improvement depends on *scaling units* for each objective that are defined automatically in the system basing on the differences between the utopia point, the current aspiration point and the current reservation point, therefore, implicitly defined by the user. This automatic definition of scaling units has many advantages to the user who is not only free from specifying scaling units but also has a better control over the selection of efficient outcomes by changing reference levels.

After scanning several representative efficient solutions and outcomes controlled by changing references, the user typically learns enough either to subjectively select an actual decision (which need not correspond to the decisions proposed in the system, since even the best substantive model can differ from real decision situation) or to select an efficient decision proposed by the system as a basis for actual decisions. Rarely, the user can still be uncertain what decision is to be chosen; for this case, several additional options might have been included in a system of DIDAS-type. Such options should consist of two more sophisticated scanning rules: 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 a *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 may represent the best solution for the preferences of the user. However, these additional options have not been implemented in IAC-DIDAS-N, because the experience with DIDAS-type systems shows that these options are rarely useful.

2 Theoretical manual

The standard form of a multiobjective nonlinear programming problem is defined as follows:

$$\underset{x \in X}{\text{maximize}} \quad [q = f(x)]; \quad X = \{ x \in R^n : g'(x) = 0, g''(x) \leq 0 \} \tag{1}$$

where $x \in R^n$, $q \in R^p$, $f : R^n \to R^p$ is a given function (assumed to be differentiable), $g' : R^n \to R^{m'}$ and $g'' : R^n \to R^{m''}$ 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 $\hat{q} = f(\hat{x})$, $\hat{x} \in X$ 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 was only required that there were 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 outcomes is denoted by

$$Q = \{ q \in R^p : q = f(x), x \in X \} \tag{2}$$

and so called *positive cones*

$$D = R^p_+ = \{ q \in R^p : q_i \geq 0, i = 1, \ldots, p \}, \quad \widetilde{D} = R^p_+ \setminus \{0\}, \quad \widetilde{\widetilde{D}} = \operatorname{int} R^p_+ \tag{3}$$

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 $q - \hat{q} \in \widetilde{D}$ and $q > \hat{q}$ as $q - \hat{q} \in \widetilde{\widetilde{D}}$), then the sets of efficient outcomes $\hat{Q}$ and of weakly efficient outcomes $\hat{Q}^w$ can be written as:

$$\hat{Q} = \{ \hat{q} \in Q : (\hat{q} + \widetilde{D}) \cap Q = \emptyset \} \tag{4}$$

$$\hat{Q}^w = \{ \hat{q} \in Q : (\hat{q} + \widetilde{\widetilde{D}}) \cap Q = \emptyset \} \tag{5}$$

where $\emptyset$ denotes an empty set.

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 may 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:

$$D_\epsilon = \{ q \in R^p : \operatorname{dist}(q, D) \leq \epsilon \|q\| \} \tag{6}$$

where any norm in R^p is used, also for the definition of the distance between q and D. The corresponding, modified definition of D_ϵ-efficiency:

$$\hat{Q}^{p\epsilon} = \{ \hat{q} \in Q : (\hat{q} + \widetilde{D}_\epsilon) \cap Q = \emptyset \}; \quad \widetilde{D}_\epsilon = D_\epsilon \setminus \{0\} \tag{7}$$

applies to properly efficient outcomes that have trade-off coefficients a priori bounded by approximately ε and $1/\varepsilon$; such outcomes are also called *properly efficient with* (a priori) *bound* (see Wierzbicki, 1986).

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 $\hat{Q}^{p\varepsilon}$ he may 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 on all decision and outcome variables:

$$y = g(x); \quad x^{\mathrm{lo}} \leq x \leq x^{\mathrm{up}}; \quad y^{\mathrm{lo}} \leq y \leq y^{\mathrm{up}} \tag{8}$$

where $g : R^n \to R^m$ 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 only to constraints and which correspond to objectives, is flexible and can be modified by the user. There are only inequality constraints in the definition of substantive model (8), but equality constraints for some outcomes can be easily written as

$$y_i^{\mathrm{lo}} \leq y_i \leq y_i^{\mathrm{up}} \quad \text{with} \quad y_i^{\mathrm{lo}} = y_i^{\mathrm{up}} \quad \text{for some } i \tag{9}$$

Denote the vector of p objective outcomes by $q \in R^p \subset R^m$ (some of the objective variables may 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:

$$q = f(x) \tag{10}$$

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:

$$X = \{ x \in R^n : x^{\mathrm{lo}} \leq x \leq x^{\mathrm{up}};\ y^{\mathrm{lo}} \leq g(x) \leq y^{\mathrm{up}} \} \tag{11}$$

Moreover, the objective outcomes are not necessarily maximized; some of them can be minimized, some maximized, some stabilized or kept close to given *stabilization levels* (that is, minimized if their value is above stabilization level and maximized if their value is below stabilization level). All these possibilities can be summarized by introducing a different definition of positive cone D:

$$\begin{aligned} D = \{ q \in R^p : \ & q_i \geq 0, && 1 \leq i \leq p' \ ; \\ & q_i \leq 0, && p'+1 \leq i \leq p'' \ ; \\ & q_i = 0, && p''+1 \leq i \leq p \ \} \end{aligned} \tag{12}$$

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 (6), although the cone itself changes appropriately. Actually, the user has 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 efficient and properly efficient outcomes for the multiobjective problem.

For given some stabilization levels q_i^s for stabilized objectives and the requirement that these objectives should be minimized above and maximized below stabilization levels, the set of efficient outcomes can be defined only relative to the stabilization levels. However, since the user can define stabilization levels arbitrarily, of interest here is the union of such relative sets of efficient outcomes. Let $\widetilde{D} = D \setminus \{\emptyset\}$ and $\widetilde{D}_\varepsilon = D_\varepsilon \setminus \{\emptyset\}$; then, for arbitrary stabilization levels, the outcomes efficient or properly efficient with bound can be defined, as before, by the relations (4) or (7). 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 outcomes with equality constraints (9); 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 stabilization levels, but they can differ from these 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 removing equality constraints for some outcomes and putting these outcomes into the stabilized objective category. Outcomes with inequality constraints can be converted in the same way to either minimized or maximized outcomes.

By adding shifting scales, adding a number of proxy variables and changing the interpretation of the function g, the substantive model formulation (8), (9), (10), (11) together with its positive cone (12) and the related concept of efficiency can 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 the user-controlled selection of an efficient decision and outcome from the set (4) or (7). For stabilized objective outcomes, the user can change the related stabilization levels in order to influence this selection; *it is assumed here that he will do so for all objective outcomes*, that is, *he will use the corresponding reference levels in order to influence the selection of efficient decisions.*

For minimized and maximized objectives the user can specify two kinds of reference levels: *aspiration levels* denoted here $\bar{q}_i$ or $\bar{q}$ as a vector called *aspiration point* and *reservation levels* denoted $\bar{\bar{q}}_i$ or $\bar{\bar{q}}$ as a vector called *reservation point*. The aspiration levels represent the levels that the user would like to attain (although the aspiration point as whole is not attainable in most cases), whereas the reservation levels could be interpreted as 'soft' lower limits for objectives (for maximized objectives; upper limits for minimized objectives). Reservation levels $\bar{\bar{q}}_i$ for maximized objectives should be 'below' the aspiration levels $\bar{q}_i$ $(\bar{\bar{q}}_i < \bar{q}_i,\ i = 1,\ldots,p')$, whereas reservation levels $\bar{\bar{q}}_i$ for minimized objectives should be 'above' the aspiration levels $\bar{q}_i$ $(\bar{\bar{q}}_i > \bar{q}_i,\ i = p'+1,\ldots,p'')$. If these conditions are not satisfied for some objectives, system automatically changes $\bar{q}_i$ or $\bar{\bar{q}}_i$.

For each stabilized objective q_i the user can specify the *lower reservation level* denoted $\bar{\bar{q}}_i^l$ and the *upper reservation level* denoted $\bar{\bar{q}}_i^u$. It is assumed that the stabilization level q_i^s is given implicitly as the mean value of two reservation levels $q_i^s = (\bar{\bar{q}}_i^l + \bar{\bar{q}}_i^u)/2$, thus, the user defines the *reservation range* around the stabilization level. Moreover, the system defines internally the *lower aspiration level* $\bar{q}_i^l = q_i^s - \delta(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2$ and the *upper*

aspiration level $\bar{q}_i^u = q_i^s + \delta(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2$, thus, the *aspiration range* is δ times narrower than the reservation range with q_i^s being the center of both ranges. The coefficient δ has the default value 0.1 and can be changed by the user during the interactive process.

The aspiration and reservation points, called jointly *reference points*, are both user-selectable parameters (for minimized and maximized objectives; for stabilized objectives two reservation levels are user-selectable). 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 by maximizing an *order-approximating achievement function* (see Wierzbicki, 1983, 1986). There are several forms of such functions; properly efficient outcomes with approximate bound $\varepsilon, 1/\varepsilon$ are obtained when maximizing a function of the following form:

$$\begin{aligned} s(q, \bar{q}, \bar{\bar{q}}) \;=\; & \min\left(\min_{1\le i\le p''} z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i),\; \min_{p''+1\le i\le p} z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) \right) + \\ & + \varepsilon \left(\sum_{i=1}^{p''} z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) + \sum_{i=p''+1}^{p} z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) \right) , \end{aligned} \tag{13}$$

where the parameter ε should be positive, even if very small; if this parameter is equal to zero, then the above function is not order-approximating any more, but *order-representing*, and its maximal points can correspond to weakly efficient outcomes.

The functions $z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$ for maximized objectives ($i = 1, \ldots, p'$) are defined by:

$$z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) = \min\left((q_i - \bar{\bar{q}}_i)/s_i',\, 1 + (q_i - \bar{q}_i)/s_i''\right) \tag{14}$$

and the functions $z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i)$ for minimized objectives ($i = p' + 1, \ldots, p''$) are defined by:

$$z_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) = \min\left((\bar{\bar{q}}_i - q_i)/s_i',\, 1 + (\bar{q}_i - q_i)/s_i''\right) \tag{15}$$

while the functions $z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u)$ for stabilized objectives ($i = p'' + 1, \ldots, p$) are defined by:

$$\begin{aligned} z_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) &= \min\left(z_i^l, z_i^u\right) \\ z_i^l &= \min\left((q_i - \bar{\bar{q}}_i^l)/s_i',\, 1 + (q_i - \bar{q}_i^l)/s_i''\right) \\ z_i^u &= \min\left((\bar{\bar{q}}_i^u - q_i)/s_i',\, 1 + (\bar{q}_i^u - q_i)/s_i''\right) \end{aligned} \tag{16}$$

where

$$\begin{aligned} \bar{\bar{q}}_i^l &= q_i^s - \delta(q_i^s - \bar{\bar{q}}_i^l), \\ \bar{\bar{q}}_i^u &= q_i^s + \delta(\bar{\bar{q}}_i^u - q_i^s), \\ q_i^s &= (\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2. \end{aligned} \tag{17}$$

The coefficients $s_i' > 0$, $s_i'' > 0$ in (14), (15) and (16) are scaling units for all objectives and are determined automatically in the IAC-DIDAS-N system to obtain the following common *absolute achievement measure* for all *individual criterion achievement functions* $z_i(q_i, \cdot, \cdot)$:

$$z_i = \begin{cases} 1 + \eta & \text{if} \quad q_i = q_i^{\text{best}} \quad (q_i^s \text{ for stabilized objectives}) \\ 1 & \text{if} \quad q_i = \bar{q}_i \quad (\bar{\bar{q}}_i^l \text{ or } \bar{\bar{q}}_i^u \text{ for stabilized objectives}) \\ 0 & \text{if} \quad q_i = \bar{\bar{q}}_i \quad (\bar{\bar{q}}_i^l \text{ or } \bar{\bar{q}}_i^u \text{ for stabilized objectives}) \end{cases} \tag{18}$$

where q_i^{best} is the upper limit (for maximized objectives; lower limit for minimized objectives) of all attainable efficient values of objective q_i and $\eta > 0$ is an arbitrary coefficient.

For minimized or maximized objectives ($i = 1, \ldots, p''$), scaling coefficients s_i' and s_i'' depend on relations between aspiration level $\bar{q}_i$, reservation level $\bar{\bar{q}}_i$ and upper limit q_i^{max} (for maximized objectives; lower limit q_i^{min} for minimized objectives) of all attainable efficient values of objective q_i :

$$\begin{aligned} s_i' &= \bar{q}_i - \bar{\bar{q}}_i, \quad s_i'' = (q_i^{max} - \bar{q}_i)/\eta, \quad \text{if} \quad 1 \le i \le p', \\ s_i' &= \bar{\bar{q}}_i - \bar{q}_i, \quad s_i'' = (\bar{q}_i - q_i^{min})/\eta, \quad \text{if} \quad p'+1 \le i \le p''. \end{aligned} \tag{19}$$

For stabilized objectives ($i = p''+1, \ldots, p$), scaling coefficients s_i' and s_i'' depend on the distance between $\bar{\bar{q}}_i^l$ and $\bar{\bar{q}}_i^u$ (i.e. reservation range) and on the user-defined coefficient δ (i.e. relations between aspiration and reservation ranges):

$$\left.\begin{aligned} s_i' &= (1-\delta)(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/2 \\ s_i'' &= \delta(\bar{\bar{q}}_i^u - \bar{\bar{q}}_i^l)/(2\eta) \end{aligned}\right\} \quad \text{if} \quad p''+1 \le i \le p. \tag{20}$$

Parameter η in (18), (19) and (20) is selected according to current relations between $\bar{q}_i$, $\bar{\bar{q}}_i$, q_i^{max}, q_i^{min} and the value of coefficient δ :

$$\eta = \min\left(\min_{1 \le i \le p'} \frac{q_i^{max} - \bar{q}_i}{\bar{q}_i - \bar{\bar{q}}_i}, \min_{p'+1 \le i \le p''} \frac{\bar{q}_i - q_i^{min}}{\bar{\bar{q}}_i - \bar{q}_i}, \frac{\delta}{1-\delta} \right) \tag{21}$$

The system checks and does necessary projections for three sets of conditions that must hold for this selection of s_i' and s_i'' :

$$\begin{aligned} \bar{\bar{q}}_i < \bar{q}_i < q_i^{max} \quad &, \text{if} \quad 1 \le i \le p', \\ q_i^{min} < \bar{q}_i < \bar{\bar{q}}_i \quad &, \text{if} \quad p'+1 \le i \le p'', \\ \bar{\bar{q}}_i^l < \bar{\bar{q}}_i^u \quad &, \text{if} \quad p''+1 \le i \le p. \end{aligned} \tag{22}$$

The achievement function $s(q, \bar{q}, \bar{\bar{q}})$ can be maximized with $q = f(x)$ over $x \in X$; however, the function (13) 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 (13) in the system can be converted automatically to an equivalent differentiable nonlinear programming problem by introducing proxy variables and substituting the min operation in (13) by a number of additional inequalities. If the coefficient ε is positive ($\varepsilon > 0$), then the achievement function has the following properties (see Wierzbicki, 1986):

a) For any arbitrary aspiration and reservation points satisfying conditions (22), not necessarily restricted to be attainable ($\bar{q} \in Q$, $\bar{\bar{q}} \in Q$) or not attainable ($\bar{q} \notin Q$, $\bar{\bar{q}} \notin Q$), each maximal point $\hat{q}$ of the achievement function $s(q, \bar{q}, \bar{\bar{q}})$ with $q = f(x)$ over $x \in X$ is a D_ε-efficient solution, that is, a properly efficient solution with trade-off coefficients bounded approximately by ε and $1/\varepsilon$.

b) For any properly efficient outcome $\hat{q}$ with trade-off coefficients bounded by ε and $1/\varepsilon$, there exist such aspiration $\bar{q}$ and reservation $\bar{\bar{q}}$ points that the maximum of the achievement function $s(q, \bar{q}, \bar{\bar{q}})$ 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 some attainable but not efficient reservation point $\bar{\bar{q}}$ and an aspiration point $\bar{q}$ that in itself is such properly efficient outcome, $\bar{q} = \hat{q}$, and if conditions (22) are satisfied, then the maximum of the achievement function $s(q, \bar{q}, \bar{\bar{q}})$, equal to one, is attained precisely at this point.

c) If the aspiration point $\bar{q}$ is 'too high' (for maximized outcomes; 'too low' for minimized outcomes), then the maximum of the achievement function, smaller than one, is attained at an efficient outcome $\hat{q}$ that best approximates uniformly, in the sense of scaling units s_i', the aspiration point. If the aspiration point $\bar{q}$ is 'too low' (for maximized outcomes; 'too high' for minimized outcomes), then the maximum of the achievement function, larger than one, is attained at an efficient outcome $\hat{q}$ that is uniformly, in the sense of scaling units s_i'', 'higher' than the aspiration point.

d) By changing his aspiration $\bar{q}$ and reservation $\bar{\bar{q}}$ points, the user can continuously influence the selection of the corresponding efficient outcomes $\hat{q}$ that maximize the achievement function, provided the maximum is unique and the set $\hat{Q}^{p\varepsilon}$ is connected.

The parameter ε in the achievement function determines bounds on trade-off coefficients: if an efficient solution has trade-off coefficients that are too large or too small (say, lower than 10^{-6} or higher than 10^{6}) then for the decision maker it does not differ from weakly efficient outcomes — some of its components can 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 (13).

The achievement function (13) can be transformed to an equivalent form when taking into account the scaling coefficients determined by (19) and (20) and assuming, for simplicity, that the parameter $\varepsilon = 0$:

$$s(q, \bar{q}, \bar{\bar{q}}) = 1 + \eta - \max\left(\max_{1 \le i \le p''} \tilde{z}_i(q_i, \bar{q}_i, \bar{\bar{q}}_i), \max_{p''+1 \le i \le p} \tilde{z}_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) \right) \tag{23}$$

with

$$\begin{aligned} \tilde{z}_i(q_i, \bar{q}_i, \bar{\bar{q}}_i) &= \max\left(w_i', w_i''\right), \qquad 1 \le i \le p'', \\ \tilde{z}_i(q_i, \bar{\bar{q}}_i^l, \bar{\bar{q}}_i^u) &= \max\left(w_i^{-\prime}, w_i^{-\prime\prime}, w_i^{+\prime\prime}, w_i^{+\prime}\right), \qquad p''+1 \le i \le p, \end{aligned} \tag{24}$$

where

$$w_i' = \begin{cases} \eta + \dfrac{\bar{q}_i - q_i}{\bar{q}_i - \bar{\bar{q}}_i}, & \text{if} \quad 1 \le i \le p', \\ \eta + \dfrac{q_i - \bar{q}_i}{\bar{\bar{q}}_i - \bar{q}_i}, & \text{if} \quad p'+1 \le i \le p'', \end{cases} \tag{25}$$

$$w_i'' = \begin{cases} \eta \dfrac{q_i^{\max} - q_i}{q_i^{\max} - \bar{q}_i}, & \text{if} \quad 1 \le i \le p', \\ \eta \dfrac{q_i - q_i^{\min}}{\bar{q}_i - q_i^{\min}}, & \text{if} \quad p'+1 \le i \le p'', \end{cases} \tag{26}$$

$$\left.\begin{aligned} w_i^{-\prime} &= \eta + \frac{\bar{\bar{q}}_i^l - q_i}{\bar{\bar{q}}_i^l - \bar{q}_i^l} \\ w_i^{-\prime\prime} &= 2\eta\,\frac{q_i^g - q_i}{\bar{q}_i^u - \bar{q}_i^l} \\ w_i^{+\prime\prime} &= 2\eta\,\frac{q_i - q_i^g}{\bar{q}_i^u - \bar{q}_i^l} \\ w_i^{+\prime} &= \eta + \frac{q_i - \bar{q}_i^u}{\bar{\bar{q}}_i^u - \bar{q}_i^u} \end{aligned}\right\} \quad \text{if } p'' + 1 \le i \le p, \tag{27}$$

with q_i^g, $\bar{q}_i^l$ and $\bar{q}_i^u$ given by (17).

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 parameters $\bar{q}$ or $\bar{\bar{q}}$, 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 of the user to adjust 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 (23) has been also developed and is actually used in IAC-DIDAS-N. This *smooth order-approximating achievement function* has the form:

$$\begin{aligned} s(q, \bar{q}, \bar{\bar{q}}) = 1 + \eta - \Bigg\{ & \sum_{i=1}^{p''} \left[(\max(0, w_i'))^\alpha + (w_i'')^\alpha \right] + \\ & + \sum_{i=p''+1}^{p} \left[\left(\max(0, w_i^{-\prime})\right)^\alpha + \left(\max(w_i^{-\prime\prime}, w_i^{+\prime\prime})\right)^\alpha + \left(\max(0, w_i^{+\prime})\right)^\alpha \right] \Bigg\}^{1/\alpha} \end{aligned} \tag{28}$$

where w_i', w_i'', $w_i^{-\prime}$, $w_i^{-\prime\prime}$, $w_i^{+\prime\prime}$ and $w_i^{+\prime}$ are given by (25), (26) and (27).

The parameter $\alpha \ge 2$ is responsible for the approximation of the function (13) or (23) by the function (28): if $\alpha \to \infty$ and $\varepsilon \to 0$, then these functions converge to each other (if taking into account the specific definition of scaling coefficients in (13)). However, the use of too large parameter α results in badly conditioned problems when maximizing function (28), hence $\alpha = 4 \div 10$ are suggested to be used, the default value is $\alpha = 10$. During numerical computations a slightly simpler *scalarizing function* is used and minimized:

$$\begin{aligned} \tilde{s}(q, \bar{q}, \bar{\bar{q}}) = & \sum_{i=1}^{p''} \left[(\max(0, w_i'))^\alpha + (w_i'')^\alpha \right] + \\ & + \sum_{i=p''+1}^{p} \left[\left(\max(0, w_i^{-\prime})\right)^\alpha + \left(\max(w_i^{-\prime\prime}, w_i^{+\prime\prime})\right)^\alpha + \left(\max(0, w_i^{+\prime})\right)^\alpha \right] \end{aligned} \tag{29}$$

The function (29) must be minimized with $q = f(x)$ over $x \in X$, while X is determined by simple bounds $x^{\text{lo}} \le x \le x^{\text{up}}$ as well as by inequality constraints $y^{\text{lo}} \le g(x) \le y^{\text{up}}$ (or

equality constraints for some i such that $y_i^{lo} = y_i^{up}$). In the shifted penalty technique, the following function is minimized instead:

$$p(x, \xi', \xi'', u', u'') = \tilde{s}(f(x), \bar{q}, \bar{\bar{q}}) + \\ + \frac{1}{2} \sum_{i=1}^{p} \xi_i' \left(\max(0, g_i(x) - y_i^{up} + u_i') \right)^2 + \\ + \frac{1}{2} \sum_{i=1}^{p} \xi_i'' \left(\max(0, y_i^{lo} - g_i(x) + u_i'') \right)^2 \tag{30}$$

where ξ', ξ'' are penalty coefficients and u', u'' are penalty shifts. This function is minimized with respect to x such that $x^{lo} \leq x \leq x^{up}$ while applying conjugate gradient directions, projected on these simple bounds if some of them become active. When a minimum of this penalty function with given penalty coefficients and given penalty shifts (the latter are initially equal to 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 minimizing the scalarizing function (29) 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.

Another advantage for the user is that he is bothered neither with the definition of derivatives of penalty function (30), needed in the conjugate gradient method, nor even with the definition of the derivatives of constraint functions $g_i(x)$ and outcome functions $f(x)$. This is the unique feature of IAC-DIDAS-N system: 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 that uses shifted penalty technique.

The only parameter that may influence the interaction of the system with the user is the parameter α in the smooth scalarizing function (29). Thus, the user can select this parameter; if this parameter is very large, his control of efficient outcomes obtained by minimizing (29) is somewhat easier, but the solver may take long time or produce not quite robust results in this case. The user has also access to some other parameters of the optimization procedures; it is needed in cases of especially difficult optimization problems.

The minimization of a scalarizing 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, or *floating* (that is, displayed for user's information only, but not included as optimized objectives; such outcome should be defined as minimized or maximized but with neither aspiration level nor reservation level defined).

The scalarizing function of the form (29) uses two kinds of additional information:

- bounds for efficient outcomes: 'upper' bounds for maximized outcomes, 'lower' bounds for minimized outcomes. These bounds must be determined once for the given multiobjective analysis problem.

- user-supplied reference levels: aspiration level and reservation level for each minimized or maximized outcome, two reservation levels for each stabilized outcome. The user changes reference levels (aspiration, reservation or both) several times during the interactive analysis of the multiobjective problem, however some initial values should be determined in the system.

In the initial analysis phase of the work with the IAC-DIDAS-N system the bounds for efficient outcomes are calculated: the 'upper' (in the meaning of the 'best' attainable) and the 'lower' (in the meaning of the 'worst' attainable and efficient). The former is determined exactly (with given numerical accuracy), whereas the latter is only approximated, because there is no constructive way to determine it exactly for nonlinear multicriteria problems.

The 'upper' bound for efficient solutions is 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), while all other objectives (including stabilized ones) should be considered as floating or free. The scalarizing function (29) is not used during these calculations, objective functions $q_i = f_i(x)$ are used in the penalty function instead of $\tilde{s}$ (with the plus sign if the objective under consideration should be minimized or with the minus sign if it should be maximized). If there are no stabilized outcomes, the results of such optimizations form a point that limits from 'above' (for maximized outcomes; from 'below' for minimized outcomes) the set of efficient outcomes $\widehat{Q}$, but this point almost never (except in degenerate cases) is in itself an attainable outcome; therefore, it is called the *utopia point*. The total number of optimization runs in utopia point computations is $p'' + 2(p - p'')$.

During all these computations, the 'lower' bound for efficient outcomes can be also estimated, just by recording the lowest (for maximized objectives; highest for minimized objectives) efficient outcomes that occur in subsequent optimizations (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, strict 'lower' bound for efficient outcomes — called *nadir point* $\hat{q}^{\mathrm{nad}}$ — only if $p'' = 2$; for larger number of maximized and minimized objectives, particularly for nonlinear models, this procedure can give misleading results. In further computations appropriate components of $\hat{q}^{\mathrm{uto}}$ and $\hat{q}^{\mathrm{nad}}$ are used as components of q^{max} and q^{min} in the scalarizing function (29).

In very rare and rather degenerate cases some components $\hat{q}_i^{\mathrm{nad}}$ of the nadir point estimation and corresponding components $\hat{q}_i^{\mathrm{uto}}$ of the utopia point can have the same value — it may happen if, for example, the structure of the substantive model results in the set (2) with empty interior. In such case the user can update manually these nadir point components according to his knowledge, otherwise the IAC-DIDAS-N system assumes such outcomes to be floating (they are not included in the scalarizing function (29) regardless of its type — maximized, minimized or stabilized) but checks their values at each efficient solution whether they are still equal to the values $\hat{q}_i^{\mathrm{nad}} = \hat{q}_i^{\mathrm{uto}}$.

The approximate bounds $\hat{q}^{\mathrm{uto}}$ and $\hat{q}^{\mathrm{nad}}$ once computed and presented to the user can be utilized in various ways. First, their appropriate components are used as components of q^{max} and q^{min} in the scalarizing function (29). Second way consists in computing a *neutral efficient solution*, with objectives situated approximately 'in the middle' of the efficient set. For this purpose, the aspiration point $\bar{q}$ is set very close to the utopia point $\hat{q}^{\mathrm{uto}}$ (only for maximized or minimized outcomes; for stabilized outcomes upper and lower limits of efficient outcomes are used as appropriate reservation levels $\bar{\bar{q}}_i^{\mathrm{l}} = \hat{q}_i^{\mathrm{min}}$ and $\bar{\bar{q}}_i^{\mathrm{u}} = \hat{q}_i^{\mathrm{max}}$) and the reservation point $\bar{\bar{q}}$ is set very close to the nadir point $\hat{q}^{\mathrm{nad}}$ (only for maximized and

minimized objectives). By minimizing the scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$ 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. Basing on the neutral efficient solution $\hat{q}^{\mathrm{neu}}$ and bounds on efficient objectives $\hat{q}^{\mathrm{uto}}$ and $\hat{q}^{\mathrm{nad}}$, system proposes to the user the following initial values for aspiration levels $\bar{q}_i$ and reservation levels $\bar{\bar{q}}_i$ for maximized and minimized objectives:

$$\left.\begin{aligned} \bar{q}_i &= \hat{q}_i^{\mathrm{neu}} - (\hat{q}_i^{\mathrm{neu}} - \hat{q}_i^{\mathrm{uto}})/3 \\ \bar{\bar{q}}_i &= \hat{q}_i^{\mathrm{neu}} + (\hat{q}_i^{\mathrm{neu}} - \hat{q}_i^{\mathrm{uto}})/3 \end{aligned}\right\} \quad \text{if} \quad 1 \le i \le p'' \tag{31}$$

and the following initial values for lower $\bar{\bar{q}}_i^l$ and upper $\bar{\bar{q}}_i^u$ reservation levels for stabilized objectives:

$$\left.\begin{aligned} \bar{\bar{q}}_i^l &= \hat{q}_i^{\mathrm{neu}} - \Delta_i \\ \bar{\bar{q}}_i^u &= \hat{q}_i^{\mathrm{neu}} + \Delta_i \end{aligned}\right\} \quad \text{if} \quad p'' + 1 \le i \le p \tag{32}$$

where

$$\Delta_i = 0.5 \min \left(\hat{q}_i^{\mathrm{neu}} - q_i^{\min}, q_i^{\max} - \hat{q}_i^{\mathrm{neu}} \right)$$

These values, although rather arbitrary, constitute a good starting point for further interaction.

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 aspiration point $\bar{q}$ (and, optionally, the reservation point $\bar{\bar{q}}$) for maximized and minimized objectives and reservation levels $\bar{\bar{q}}^l$ and $\bar{\bar{q}}^u$ for stabilized objectives in the minimized scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$. It can be shown (see Wierzbicki, 1986) that the best suited choice for this purpose is the choice of scaling units s_i' and s_i'' that are not constant, but are changed implicitly by the user and depend on differences between current values of aspiration and reservation levels and utopia point components either according to (19) and (20) or, equivalently, as a result of using the scalarizing function (29) with (25), (26) and (27) provided that conditions (22) hold. The interpretation of such way of setting scaling units is that the user attaches implicitly more importance to reaching an aspiration component $\bar{q}_i$ if he places it close to the known utopia component; in such case, the corresponding scaling unit becomes smaller and the corresponding objective component weighs stronger in the scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$. Thus, this way of *scaling relative to utopia-reference difference* takes into account the implicit information given by the user in the relative position of the aspiration point. The only drawback of the described choice of scaling units are strong inequalities in conditions (22), not convenient for the user and for the numerical application. Therefore, $q_i^{\max}$ and $q_i^{\min}$ in (25) and (26) are not taken directly as appropriate components of $\hat{q}^{\mathrm{uto}}$ and $\hat{q}^{\mathrm{nad}}$, but slightly displaced utopia and nadir points are used instead in the current system implementation:

$$\begin{aligned} q_i^{\max} &= \hat{q}_i^{\mathrm{uto}} + 0.01(\hat{q}_i^{\mathrm{uto}} - \hat{q}_i^{\mathrm{nad}}), \quad \text{if} \quad 1 \le i \le p', \\ q_i^{\min} &= \hat{q}_i^{\mathrm{uto}} - 0.01(\hat{q}_i^{\mathrm{nad}} - \hat{q}_i^{\mathrm{uto}}), \quad \text{if} \quad p' + 1 \le i \le p''. \end{aligned} \tag{33}$$

It is assumed now that the user selects the aspiration and reservation components satisfying $\hat{q}_i^{\mathrm{nad}} \le \bar{\bar{q}}_i < \bar{q}_i \le \hat{q}_i^{\mathrm{uto}}$ for maximized outcomes and $\hat{q}_i^{\mathrm{uto}} \le \bar{q}_i < \bar{\bar{q}}_i \le \hat{q}_i^{\mathrm{nad}}$ for minimized outcomes and $\bar{\bar{q}}_i^l < \bar{\bar{q}}_i^u$ for stabilized outcomes (if he does not, the system automatically does necessary projections). If the user specifies only one reference value for some

objective, then the system determines the second value internally, thus the same two reference level scalarizing function can be used. For maximized and minimized objectives missing reservation levels are calculated using formulae:

$$\bar{\bar{q}}_i = \begin{cases} \bar{q}_i - (q_i^{\max} - \bar{q}_i), & \text{if} \quad 1 \le i \le p', \\ \bar{q}_i + (\bar{q}_i - q_i^{\min}), & \text{if} \quad p'+1 \le i \le p'', \end{cases} \tag{34}$$

whereas missing aspiration levels are calculated using formulae:

$$\bar{q}_i = \begin{cases} 0.5(q_i^{\max} + \bar{\bar{q}}_i), & \text{if} \quad 1 \le i \le p', \\ 0.5(\bar{\bar{q}}_i + q_i^{\min}), & \text{if} \quad p'+1 \le i \le p''. \end{cases} \tag{35}$$

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

For analyzing the sensitivity of an efficient solution to changes in the proportions of outcomes, a *multidimensional scan* of efficient outcomes can be applied in IAC-DIDAS-N. This operation consists in selecting an efficient outcome, accepting it as a base $\bar{q}^{\text{bas}}$ for aspiration points, and performing p (or p'') additional optimization runs with the aspiration points determined by:

$$\bar{q}_j = \bar{q}_j^{\text{bas}} + \beta \left(\hat{q}_j^{\text{uto}} - \hat{q}_j^{\text{nad}} \right), \quad \bar{q}_i = \bar{q}_i^{\text{bas}}, \quad i \ne j, \quad 1 \le i \le p \tag{36}$$

where β is a coefficient determined by the user, $-1 \le \beta \le 1$; if the aspiration components determined by (36) are outside the range $\hat{q}_j^{\text{nad}}$, $\hat{q}_j^{\text{uto}}$, they are projected automatically on this range; the reservation point is kept constant ($\bar{\bar{q}} = \hat{q}^{\text{nad}}$) during this procedure. The aspiration components for stabilized outcomes may or may not be perturbed in this operation. The efficient outcomes, resulting from the minimization of the scalarizing function $\tilde{s}(q, \bar{q}, \bar{\bar{q}})$ with such perturbed aspiration points, are typically also perturbed mostly along their respective components, although other their components may also change.

For analyzing 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 IAC-DIDAS-N. This operation consists again in selecting an efficient outcome, accepting it as a base $\bar{q}^{\text{bas}}$ for aspiration points, selecting another aspiration point $\bar{q}$, and performing a user-specified number K of additional optimizations with aspiration points determined by:

$$\bar{q}(k) = \bar{q}^{\text{bas}} + \frac{k}{K} \left(\bar{q} - \bar{q}^{\text{bas}} \right), \quad 1 \le k \le K \tag{37}$$

The efficient solutions $\hat{q}(k)$ obtained through minimizing the scalarizing function $\tilde{s}(q, \bar{q}(k), \bar{\bar{q}})$ with such aspiration points (and constant reservation point $\bar{\bar{q}} = \hat{q}^{\text{nad}}$) constitute a cut through the efficient set $\widehat{Q}$ 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 with convergence to the most preferred solution is choosing aspiration points as in (37) 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. It results in convergence even if the user makes stochastic errors in determining next directions of improvement of aspiration points, and 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 scan nor directional scan are implemented in the current version of IAC-DIDAS-N, though they could be included in later versions.

3 Short user manual

3.1 Introduction

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.

The system can be run on an `IBM--PC--XT, AT` or a compatible computer with Hercules Graphics Card, Color Graphic Adapter or Enhanced Graphics Adapter and, preferably, with a numeric coprocessor and a hard disk. If a numeric coprocessor is available, then the system takes advantage of the coprocessor computational capacity, otherwise the system uses built-in software emulator of the numeric coprocessor with less computational power. The system is recorded on one diskette. The diskette contains the compiled code of the program together with some data files with demonstrative examples of nonlinear models. When the installation of of the system in the user directory on a hard disk (or less preferably on a working diskette) is done (using `INSTALL` batch file contained on the diskette — see the installation guide in the Appendix A), the program can be activated by the command `DIDASN` at the `DOS` prompt.

System supports the following general functions:

- definition and edition of a substantive model of the decision situation in a user-friendly way using a spreadsheet and a screen window editor.
- simulation of the model. All numerical errors can be fixed directly. This is performed by model debugger with visualisation of outcome formulae and their derivatives, automatic error tracking together with forward and backward step by step calculations option.
- specification of a multiobjective decision analysis problem related to the substantive model. This is performed by specific features of spreadsheet edition.
- 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 and graphical form.

- interactive analysis of the problem with the stress on learning by the user of possible efficient decisions and outcomes, organized as system's response to user-specified aspiration and reservation levels for objective outcomes. The IAC-DIDAS-N system responds with efficient solutions and objective outcomes obtained by the maximization of an achievement function that is parameterized by the user-specified aspiration and reservation points. A nonlinear programming algorithm called solver performs the maximization. The interactive analysis is supported by entering user data into specific cells in the spreadsheet, executing commands from the menu and using graphical representation of results.

The menus of IAC-DIDAS-N are organized as pull-down tree-structured menus and they perform various functions used in several phases of the interactive analysis process. Most of the functions of the model edition phase as well as of the phase of the decision problem specification and the problem initial analysis 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 lower and upper bounds, for defining user names of objective outcomes and their types, reference points, utopia point, for solutions corresponding to the reference points). The functions of the interactive analysis phase are executed by macrocommands involved by menus and various function keys; the user can get various help displays that suggest in an easy fashion the commands useful in a current phase of the work with the system.

IAC-DIDAS-N system has been developed in the Institute of Automatic Control, Warsaw University of Technology, Warsaw, Poland which has the authorship rights, under the 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 near Vienna, Austria, which has the copyright for this system in international distribution. Please contact Methodology of Decision Analysis Project of SDS Program at IIASA, A–2361 Laxenburg, Austria.

3.2 Phases of the work

The work with a IAC-DIDAS-N system consists of three phases:

1. model edition phase
2. problem definition and initial analysis phase
3. interactive analysis and comparison of results phase

All these phases are supported in the system and an explicit command is required to move from one phase to another. Moreover, system checks whether requested move is possible and gives appropriate error messages or asks for additional confirmation. There are two logical spreadsheets: a model editing spreadsheet and an interactive analysis spreadsheet. The former is used mostly to perform all the system functions in the first phase of the work, whereas the latter performs all the system functions during two other phases.

System invoked without arguments always starts with phase 1 and permits of the move to phase 2 only if the model definition is complete. A complete model consists of three groups of obligatory data:

- valid formulae for all defined outcomes (rows of the spreadsheet),

- lower and upper bounds (that do not contradict each other) for all variables,
- values for all used parameters.

Optionally, model can contain some other, user-supplied data:

- lower and upper bounds (that do not contradict each other) for outcomes,
- names of variables, parameters and outcomes (user names override standard system names of the form **x1**, **x2**, ...for variables, **z1**, **z2**, ...for parameters and **y1**, **y2**, ...for outcomes). The names must be unique within the model.
- units for variables, parameters and outcomes. This information can be included to improve the understanding of the model in the spreadsheet, but it is not used by the system. The only exception is the use of outcome units for special scaling method in graphical representation.
- lower and upper bounds for parameters; not used in the current system implementation but planned to be used in parametric analysis in future system implementations.
- short (up to 30 characters) model description; it is displayed in the spreadsheet and printed together with the print-out of the model. It may be used as an extension to the model name, that is too short (up to 8 characters) to be enough meaningful.
- five parameters that are used to tune the nonlinear solver (see the next section). If some of them are not given, then current default system values are stored together with other model data.

To store the edited model on a disk, the name of the model must be supplied by the user. Therefore, while using < F2 > (Save) or < Alt M F > (Model selection — Fix and save) commands (see section 3.6), system asks for the name (up to 8 characters; it must be a valid DOS file name). However, there is an important distinction between these two commands. The former just saves the model as it is, whereas the latter first checks the model and either displays an error message if the model is not complete or changes the status of the model to 'fixed' and stores the complete model. Both commands check, whether the given name is not the same as the model file name already existing in the current model directory on a disk. If it is the same, then the system asks for additional confirmation. Answering 'yes' means, that the system deletes the file with the same name containing the previous model definition together with all related problem definitions and result data.

Successful 'Fix and save' command for the model moves the system automatically to the second phase of the work. Obligatory elements of the model definition and bounds for outcomes (if given) cannot be changed now. The command < Alt M N > (Model selection — New) must be used to move back to the first phase to allow any change in this part of the model definition, but it will be treated as a definition of a new model. The command < Alt M R > (Model selection — Reset) can also be used to move back to the first phase, but it deletes all model data and starts a new model definition from the scratch. Optional parts of the model definition (except the bounds for outcomes) can be changed even if the model is fixed, because they do not affect computational characteristics of the model. Such changes in the model definition can be stored on a disk using the command < F2 > (Save). In particular changes in the rows' and columns' order done with the < Alt F R M > (Format — Rows — Move) and < Alt F C M > (Format — Columns —

Move) commands are admissible for a fixed model and are stored with the < F2 > (Save) command.

Model stored on a disk can be restored using the command < Alt M G > (Model selection — Get from disk). The model is restored together with the information, whether it was fixed or not. Thus, after restoring not fixed model the system is still in the phase 1, whereas after restoring a fixed model the system moves automatically to the phase 2. It is also possible to restore a model immediately while invoking the IAC-DIDAS-N system — the name of the model must be given as the first argument in the DOS command line (for example, to restore automatically the model `DEMO`, invoke the system with the command `DIDASN DEMO`).

Edited model (not necessarily fixed) can be printed using the command < Alt M P > (Model selection — Print). Model stored on a disk, but actually not interesting to the user can be deleted from a disk together with all related problem definitions and result data using the command < Alt M E > (Model selection — Erase).

Second phase of the work with the IAC-DIDAS-N system lies in a specification of a decision problem to be analyzed, for already defined and 'fixed' model. The complete definition of a decision problem consists of two parts: user-supplied part and system-supplied part. The user-supplied part must contain two kinds of information: a selection of outcomes to be minimized objectives, maximized objectives or stabilized ones (defined by the contents of the 'Stat' column in the spreadsheet) and values of lower and upper bounds on outcomes, if they are not given in the model definition. Moreover, the user can add some other data:

- lower and upper bounds for outcomes, even if already given in the model definition; the new values override correspondent data in the model definition,
- short (up to 30 characters) problem description; it is displayed in the spreadsheet and printed together with the print-out of the problem. It may be used as an extension to the problem name, that is too short (up to 8 characters) to be enough meaningful.
- five parameters that are used to tune the nonlinear solver (see the next section); these values override correspondent data either in the model definition or default system values.
- new bounds (updates) of the system-supplied approximation of the nadir point.

For a given set of selected objectives and bounds for objectives (and optional bounds redefinitions and some solver parameters) system performs initial analysis of the decision problem: calculation of ranges of efficient solutions (utopia and nadir points) and calculation of neutral solution being the starting point for further interaction during the third phase of the work (see the theoretical manual). Two parts of this analysis can be performed jointly using the command < F3 > (Calculate) or separately using first the command < Alt P U > (Problem selection — Utopia) and next the command < Alt P T > (Problem selection — neuTral). The system-supplied part of a complete problem consists of results of these calculations. If the calculations are performed separately, then after calculating the utopia point (but prior to the neutral solution calculation) the user can modify values of the nadir point approximation using the command < Alt P A > (Problem selection — nAdir).

To store the edited problem on a disk, the name of the problem must be supplied by the user. Therefore, while using < F2 > (Save) or < Alt P F > (Problem selection — Fix

and save) commands, system asks for the name (up to 8 characters). However, there is again an important distinction between these two commands. The former just saves the problem as it is, whereas the latter first checks the problem and either displays an error message if the problem is not complete or changes the status of the problem to 'fixed' and stores the complete problem. The file containing the model definition is automatically updated if any changes are made in its optional part. Both commands check, whether the given name is not the same as the problem name already existing for the currently used model. If it is, then system asks for additional confirmation. Answer 'yes' means, that the system deletes previous problem definition together with all related result data.

Successful 'Fix and save' command for the problem moves the system automatically to the third phase of the work. None of the problem elements can be changed now. The command < Alt P N > (Problem selection — New) must be used to move back to the second phase to allow any change in the problem definition, but it will be treated as a definition of a new problem. The command < Alt P R > (Problem selection — Reset) can also be used to move back to the second phase, but it deletes all problem data and starts new problem definition from the scratch.

Problem stored on a disk can be restored using the command < Alt P G > (Problem selection — Get from disk). The problem is restored together with the information, whether it was fixed or not. Thus, after restoring not fixed problem the system is still in the phase 2, whereas after restoring a fixed problem the system moves automatically to the phase 3. It is also possible to restore a problem immediately while invoking the IAC-DIDAS-N system — the name of the model must be given as the first argument in the DOS command line and the name of the problem must be given as the second argument in the DOS command line (for example, to restore automatically the problem `FIRST` defined for the model `DEMO`, invoke the system with the command `DIDASN DEMO FIRST`).

The result of neutral solution calculation is stored as a result with a system-defined name 'Neutral'. It is stored and restored each time when a fixed problem related to it is stored and restored.

Edited problem (not necessarily fixed) can be printed using the command < Alt P P > (Problem selection — Print). Moreover, if the problem is fixed, then the neutral result can also be printed using the command < Alt R P > (Result selection — Print). Problem stored on a disk, but actually not interesting to the user can be deleted together with all related result data using the command < Alt P E > (Problem selection — Erase).

Third phase of the work with the IAC-DIDAS-N system lies in an interactive analysis of the decision problem already defined and 'fixed', for defined and 'fixed' model. The only values, that are changed by the user during this phase, are aspirations and reservations for minimized or maximized objectives as well as lower and upper reservations for stabilized objectives (see the theoretical manual). The complete result consists of these user-supplied data together with the system's response — an efficient solution calculated using scalarizing function parameterized with these data. Two commands, either < F3 > (Calculate) or < Alt R C > (Result selection — Calculate), can be used to start the calculations.

Moreover, the user can add short (up to 30 characters) result description; it is displayed in the spreadsheet and printed together with the print-out of the result. It may be used as an extension to the result name, that is too short (up to 8 characters) to be enough meaningful.

To store the calculated result on a disk, the name of the result must be supplied by the user. Therefore, while using < F2 > (Save) or < Alt R S > (Result selection — Save and

new) commands, system asks for the name (up to 8 characters). There is no difference between these two commands. Both commands first check the result and either display a message if the result is not calculated or store the calculated result. The model definition is automatically updated if any changes are made in its optional part. Both commands check, whether the given name is not the same as the result name already existing for the currently used problem and model. If it is, then system asks for additional confirmation. Answer 'yes' means, that the system deletes the previous result data. Moreover, the name 'Neutral' is reserved and cannot be used.

Two kinds of data are stored and restored as results; these are efficient values of objectives together with values of other, non-objective outcomes and values of variables related to the efficient solution. Values of variables are loaded into the spreadsheet (row Values in the model editing spreadsheet) following each successful calculation of an efficient solution and while restoring a result from a disk. The obtained or restored values are used as a starting point for the calculation of next efficient solution.

If the user finds, that calculated result is not interesting, it is possible to clear it either using the command < Alt R N > (Result selection — New) that clears only the system response or using the command < Alt R R > (Result selection — Reset) that clears also user-supplied part of the problem definition. However, only results stored on a disk can be compared using graphical representation.

Result stored on a disk can be restored using the command < Alt R G > (Result selection — Get from disk). It is also possible to restore a result immediately while invoking the IAC-DIDAS-N system — the name of the model must be given as the first argument in the DOS command line, the name of the problem must be given as the second argument in the DOS command line and the name of the result must be given as the third argument in the DOS command line (for example, to restore automatically the result `R1` obtained for the problem `FIRST` defined for the model `DEMO`, invoke the system with the command `DIDASN DEMO FIRST R1`).

Calculated result can be printed using the command < Alt R P > (Result selection — Print). Result stored on a disk, but actually not interesting to the user can be deleted from a disk using the command < Alt R E > (Result selection — Erase).

3.3 Editing with the spreadsheet

Two spreadsheets used in the IAC-DIDAS-N system are rather specialized. They differ from the standard ones (like Lotus 1–2–3) in two aspects. First, in the implemented spreadsheets there are predefined types of contents of all cells: there are dedicated cells for storing text, other cells for storing numbers and other ones for storing formulae. Secondly, the IAC-DIDAS-N has an integrated compiler with a symbolic differentiation facility, that compiles the formulae and produces binary codes for calculations of formula value and for calculations of all derivatives. Therefore, two kinds of operation are defined for spreadsheet cells with formulae: compilation and calculation. Version 4.0 of the IAC-DIDAS-N uses new compiler that generates directly binary code for numeric coprocessor; calculations of formula and derivative values are more than ten times faster then in previous versions of the system with interpreted internal code.

The top screen line in both spreadsheets contains pull-down menu entries and the bottom line contains the function keys meanings.

Both spreadsheets are built from two partially independent parts. First three columns from the left are common for both spreadsheets and have as many rows as outcomes in the model. If the model has more than 13 outcomes, then only 13 are displayed and the

spreadsheet is scrolled up and down according to moves of the spreadsheet marker. These three columns are used to enter and display outcomes' definition: name, unit and status, from left to right, respectively. However, the status is the element of a problem definition. Therefore, it can be accessed only from the interactive analysis spreadsheet. The fourth column of the model editing spreadsheet contains outcome formulae, but only their values are displayed; to display and edit the formula one cell from this column must be selected with the spreadsheet marker and then the < Enter > key causes the display of formula in a window.

In the upper left corner of both spreadsheets the status of the model, problem and result are displayed together with the amount of free memory in the relation to the amount of free memory available after the system initialization. For example, FreeMem 78 % means that only 22% of the available memory is currently used for model, problem and result definitions. If the displayed value is below 20% then it is not recommended to use the system , because of large amount of memory required temporarily during compilation of formula, optimization and graphical representation. The behavior of the system is not completely predicable in some out of memory situations. Another status value displayed in the upper left corner of the screen is the flag Auto `ON/OFF` toggled with the function key < F8 > or with the command < Alt S A > (Switches — Auto `ON/OFF`). This flag reflects the state of the automatic spreadsheet recalculation feature. If it is `ON`, then the spreadsheet is recalculated after each change of any cell, otherwise, the recalculation is only on explicit request command < F3 > (Calculate).

The second part of the model editing spreadsheet has as many columns as variables, parameters and outcomes; at the top of each column a type designator is displayed: 'var', 'par' or 'out', respectively. Each column contains: name, unit, upper bound, value, lower bound and values of derivatives, from top to bottom, respectively. First five items are elements of model definition, except values of variables that are accessible and can be changed in any phase of the work. Cells with values of outcomes are only for display and their contents can not be edited. Only up to four columns are displayed at a time. If there are more than four columns, the spreadsheet is scrolled left and right according to moves of the spreadsheet marker. Just above the area with derivative values the texts 'Partial derivative values' or 'Total derivative values' are alternatively displayed. They are toggled with the function key < F7 > or with the command < Alt S T > (Switches — Totals `ON/OFF`) and reflect the type of derivatives that are calculated and displayed. Values of partial or total derivatives are only displayed in the spreadsheet. To display a formula of a derivative an appropriate cell must be selected with the spreadsheet marker and then the < Enter > key causes the display of formula in a window. Formulae for partial derivatives can only be displyed.

The model debugger can be used if either a formula or a derivative of a formula is displayed in the editor window. According to the information displayed on the bottom of the window, < Alt D > keys activates step by step execution of currently displayed formula. < PgUp > key causes next step to be performed; partial results are displayed in additional window and the part of formula that is calculated in current step is highlighted. < PgDn > key causes the same steps to be performed backward. < Alt D > keys cause restart of the debugger from the beginning of the currently displayed formula. < Enter > key or < Esc > key terminate the debugger session.

The second part of the interactive analysis spreadsheet has seven columns, but only five of them are displayed at a time. Thus, it is scrolled one column left or one column right according to moves of the spreadsheet marker. Contents of these seven columns depend on the type of outcome and can be different for different rows of the spread-

sheet. Descriptions of the columns change according to the type of outcome which the spreadsheet marker is currently pointing to. For outcomes not selected as objectives the spreadsheet columns contain (from left to right): upper bound, blank (column with empty cells), blank, last calculated or loaded value of solution, blank, blank, lower bound. For maximized objectives there are there: upper bound, utopia value, aspiration level, last solution value, reservation level, nadir value and lower bound. For minimized objectives there are there: lower bound, utopia value, aspiration level, last solution value, reservation level, nadir value and upper bound. At last, for stabilized objectives there are there: upper bound, upper utopia value, upper reservation level, last solution value, lower reservation level, lower utopia value and lower bound. At a first look, this arrangement seems to be very complicated, but one can easily find that all values, that should be mutually compared, are placed side by side and that in most cases all values at each row either decrease or increase while moving from left to right. If the problem is not fixed, then the columns with aspirations and reservations (for minimized and maximized objectives; with lower and upper reservations for stabilized objectives) are not displayed, thus the remaining five columns are displayed all the time and there is no need to scroll the spreadsheet horizontally. At the top of this part of the interactive analysis spreadsheet there are displayed names and descriptions of current model, problem and result.

To move the spreadsheet marker eight cursor keys can be used to perform four regular and four oblique directions of move (for example, < Home > cursor key moves the marker in the upper-left direction). Moreover, it is also possible to perform fast jumps from one part of the spreadsheet to another (only in the model editing spreadsheet). Fast jumps from the right part to the left part (namely to the formulae column) and back, without change of the row and without scrolling the right part, are performed with the key combinations < Ctrl Left > and < Ctrl Right >, whereas fast jumps from the bottom part to the top part (namely to the value row) and back, without change of the column and without scrolling the bottom part, are performed with the key combinations < Ctrl PgUp > and < Ctrl PgDn >.

The way data are entered into a particular cell depends on a type of data that the cell is destined to store. To edit the contents of a particular cell one must move the spreadsheet marker to this cell. The edition of contents of a spreadsheet cell is performed by use of two editors: formula editor for edition of formula (only the formulae column in the model edition spreadsheet) and the cell editor for all other cells. In the latter case there are two possibilities: the modification of the previous contents or the input of new contents. The < Enter > key causes the entry into the cell editor using the previous contents of the cell. Any other key is treated as the first input character of the new contents; the previous contents are then deleted. The cell editor allows for the use of standard editing keys (< Backspace >, < Del >, < Left >, < Right >, < Home >, < End >) in two modes: insert mode and replace mode that are toggled with the < Ins > key. Current mode is indicated with the shape of the cursor — block cursor means insert mode, underline cursor means replace mode. There are two keys that ends the cell edition: < Enter > key means storing a new data, < Esc > key means restoring the previous contents.

To start the edition of a formula only the < Enter > key can be pressed, thus it is always the edition of the previous contents of the cell. The formula editor is a full featured window editor. Standard editing keys (< Backspace >, < Del >, < Left >, < Right >, < Up >, < Down >, < Home >, < End >, < PgUp >, < PgDn >) can be used in two modes exactly like in the cell editor. The functions of two terminating keys are also the same. Moreover, to facilitate the edition of several formulae that are very similar or even with some identical parts, the concept of a buffer has been implemented.

Any part of currently displayed formula can be marked (displayed in a reverse video mode): its beginning is marked with < F7 > function key and the end is marked with < F8 > function key. Marked area can be copied (duplicated) into the current cursor position by use of < F10 > function key. Marked area can be deleted by use of < F6 > function key. However, to avoid unintentional deletions the cursor must be at the position just following the end of the marked block, otherwise, an appropriate message is displayed and the block is not deleted. Marked area can be copied (duplicated) into the buffer using the function key < F5 > and then restored in the current cursor position using the function key < F10 >. The contents of the buffer are displayed in a window at the bottom of the screen. This window is closed (without deleting the contents of the buffer) either by use of the function key < F9 > or when the formula editor is left, and opened again by use of the function key < F9 >. If the buffer contents is too long to fit the size of the window, then < Up > and < Down > cursor keys are used to scroll it up and down, one line at a time.

3.4 Usage of the nonlinear solver

The nonlinear solver used in the IAC-DIDAS-N system is rather fast and robust. Its operation, however, depends on some parameters that should be sometimes adjusted to the properties of a particular model and problem. For small models (consisting of not more than ten variables and outcomes) the default system values of the parameters can be successfully used. There are four ways of changing values of these parameters: permanent, for a model, for a problem and for a result. Parameters are changed using menu entry Options in the either of spreadsheets. If any parameter is changed, then, while leaving the menu, system asks 'Do you want to make these changes permanent (Y/N) ?'. If the answer is yes (< Y > or < y >), then the program updates itself on a disk in such a way, that the new values become default values in all subsequent runs of the system, otherwise, the changes are only temporary, valid till the end of a current run of the system. Values of the parameters are stored together with the model definition and with the problem definition, these values are restored while the model or problem are loaded.

The first three parameters are used in stop tests of the solver:

Accuracy — the norm of gradient of the penalty function at the solution point must be not greater than the Accuracy value. The default value is 10^{-4}, but for large, highly nonlinear models this value should be changed to 10^{-2} or even more.

Violation — the nonlinear constraints (bounds on outcomes) at the solution point must not be violated more than Violation value. The default value is 10^{-4}, but this value should be also increased for large models with many nonlinear constraints.

Iterations — the limit of iterations (recalculations of the spreadsheet). The default value is 1000.

The last two parameters define the shape of the scalarizing function:

Scaling exp. — It is the parameter α in the scalarizing function (see theoretical manual). The default value is 10; it should be decreased to 4 or 6 for large models (it must be an even number).

Ratio Asp/Res — It is the parameter δ in the scalarizing function (see theoretical manual). The default value is 0.1 and may be changed within a range 0.01–0.9. The selection of this value is rather a matter of taste and does not depend on the size of the problem.

Although there are two independent stop tests (Accuracy check together with Violation check and Iterations limit check) the user can interrupt the running optimization process by pressing < Ctrl Break > key combination. The best point obtained till this moment is then displayed as a solution, but the system does not treat this point as a solution.

Optimization runs are sometimes very time consuming, therefore, to give the user an information that the system has not crashed, a sequence of letters is displayed. Each time, when the solver stops the move in a current direction, a next letter is displayed. The letters are displayed in the right half of the second to the last line on the screen starting with the letter 'A'. If all 40 positions are filled with 'A', they are cleared and letters 'B' ('C', 'D' etc.) are displayed.

There are five possible messages displayed at the end of optimization run. These are:

Optimal solution found — it is the most desirable message.

Required accuracy not attainable — it means that the solution has been found with the full numerical accuracy, but due to round-off errors the required accuracy has not been obtained. The Accuracy and Violation parameters should be increased.

Iterations limit — solution not found — the Iterations parameter should be increased.

Interrupted with < Ctrl-Break > — the user pressed the < Ctrl Break > key combination.

Solver error NNN — this message should never appear[1].

3.5 Graphical representation of results

Several objectives and several results can be displayed simultaneously in a graphical bar form. The system has some internal rules of selecting results and objectives to be displayed. These rules can be summarized as follows: display up to 10 objectives and as many results as possible. The < F9 > (Graphics) command is used to display bar representation using these rules. However, the command < Alt G O > (Graphics — Objectives selection) can be used to manually select objectives to be displayed — up to 10 objectives can be marked and next displayed. The command < Alt G R > (Graphics — Results selection) can be used to manually select results to be displayed — up to 10 results can be marked and next displayed. The user selection overrides the system rules, thus, subsequent executions of < F9 > (Graphics) command cause the display of user selected objectives and results.

Two scaling methods are implemented in the graphical representation. In the first scaling method (Normal scale `ON`) each objective is scaled independently: the maximal level (top of the bar picture) is equal to the upper bound on displayed objective and the minimal level (bottom of the bar picture) is equal to the lower bound on displayed objective. This scaling method is very simple, but has two important drawbacks. First,

[1]Please let us know if it does (NNN is the error number).

very often the bounds' range is much wider than the range of efficient solutions, therefore, all solutions together with utopia and nadir points can be represented as one point of he bar. Secondly, the objectives that form trajectories (in dynamic cases) are scaled independently and cannot be compared.

In the second scaling method (Normal scale OFF) the range of efficient changes is determined for each objective independently:

- for minimized objectives the maximal level is equal to the value of the reservation level or to the value of the solution, whichever is greater; the minimal level is equal to the value of the utopia level.
- for maximized objectives the maximal level is equal to the value of the utopia level; the minimal level is equal to the value of the reservation level or to the value of the solution, whichever is less.
- for stabilized objectives the maximal level is equal to the value of upper reservation level or to the value of the solution, whichever is greater; the minimal value is equal to the value of lower reservation level or to the value of the solution, whichever is less.

Next, for all groups of objectives with the same contents of the Units column in the spreadsheet, the common scale is determined as the distance from the lowest minimal level to the highest maximal level of all objectives within the group.

3.6 Menu and function keys description

Most commands are executed by entering into a pull-down menu, moreover some most frequently used commands are also accessible by function keys. There are two sets of pull-down menu entries and two sets of function key commands for two existing spreadsheets — model edition spreadsheet and multiobjective analysis spreadsheet; in particular, function keys are used to select one of the spreadsheets.

Pull-down menu commands can be invoked in two ways — using < Alt > key or using < Esc > key. Acting in the former way the user must press the key related to the first (highlighted) letter of the pull-down menu entry (eg. < S > key for Switches menu entry) while holding down the < Alt > key — such action will be denoted as < Alt X > where X means the letter (eg. < Alt S >). Acting in the latter way the user must press < Esc > key and the last used pull-down menu is entered — thus this way is useful for repeated commands.

In both ways, < Right > and < Left > cursor keys can be used to move from one menu entry to another. Commands within current menu entry can be selected either by moving the menu marker with < Up > and < Down > cursor keys and pressing the < Enter > key or by pressing the key related to the first (highlighted) upper case letter of the command name (in most cases it is the first letter of the command). Selection means the execution of the command or entry to the next menu level. Pressing the < Esc > key, while in a menu, causes exit from the current level of the menu either to the previous menu level or to the spreadsheet. If the menu window is too small to display all items then up and down arrows appear in the bottom line of the menu window and the menu is scrolled vertically according to the moves of the marking bar.

In the following description menu entries are terminated by a colon whereas commands are terminated by a dot. In the former case related menu items are listed below the description of the menu entry.

3.6.1 Menu for model edition

< Alt M > **Model selection:** — commands for operations on models as whole entities

- **Get from disk:** — displays menu of currently defined models; selected model is loaded from a disk into the spreadsheet
- **Fix and save.** — checks the completeness of the model, fixes it and saves on a disk (asks for model name if not named previously)
- **Description.** — asks for brief description of the model (up to 30 characters long)
- **New.** — un-fixes the model, resets the name and the description; the model in the spreadsheet remains unchanged
- **Reset.** — un-fixes the model, resets the name and the description, deletes all data in the spreadsheet, resizes the spreadsheet to one-row and one-column
- **Erase:** — displays menu of currently defined models; selected model is deleted together with all related problems and results
- **Print.** — prints current model

< Alt F > **Format:** — operations on rows or columns of the spreadsheet

- **Rows:** — changes the number or the order of rows in the spreadsheet
  - **Insert.** — inserts blank, highlighted row in the spreadsheet; this row can be moved up and down within a spreadsheet by using < Up > and < Down > cursor keys; < Alt I > makes insertion permanent, < Esc > key cancels it
  - **Delete.** — highlights a row; highlighting horizontal bar can be moved up and down within a spreadsheet by using < Up > and < Down > cursor keys, < Alt D > deletes highlighted row (if not referenced in other rows) and related column, < Esc > key cancels the command

Move. — highlights a row; the highlighting horizontal bar can be moved up and down within a spreadsheet by using < Up > and < Down > cursor keys, < Alt S > selects highlighted row. The selected row can then be moved within a spreadsheet; < Alt P > places the highlighted row in a current place, < Esc > key cancels the command

Columns: — changes the number, order or type of columns in the spreadsheet

Insert. — inserts blank, highlighted column in the spreadsheet; this column can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt V > fixes the column as a variable, < Alt P > fixes the column as a parameter (system asks for its initial value), < Esc > key cancels it

Delete. — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt D > deletes selected column (if not referenced in the spreadsheet; columns related to outcomes cannot be deleted), < Esc > key cancels the command

Move. — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt S > selects highlighted column. The selected column can then be moved within a spreadsheet; < Alt P > places the highlighted column in a current place, < Esc > key cancels the command

to Var. — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt C > changes the type of column from a previous type to a variable type (if it was an outcome then related row is deleted), < Esc > key cancels the command

to **P**ar. — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt C > changes the type of column from a previous type to a parameter type (if it was an outcome then related row is deleted), < Esc > key cancels the command

to **O**ut. — highlights a column; highlighting vertical bar can be moved left and right within a spreadsheet by using < Left > and < Right > cursor keys; < Alt C > changes the type of column from a previous type to an outcome type (related row is created), < Esc > key cancels the command

< Alt S > **S**witches: — influence numerical calculations in the spreadsheet

Totals on/off. — toggles calculation and display of values of either partial or total derivatives

Auto on/off. — switches on and off the automatic recalculation of the spreadsheet after each change of its contents

< Alt C > **C**alculate. — recalculates the whole spreadsheet

< Alt L > **L**ist: — displays the menu of models stored on a disk, for the selected model displays the menu of problems stored on a disk, for the selected problem displays the list of results stored on a disk, < Enter > key selects, < Esc > key moves back

< Alt O > **O**ptions: — changes colors and some problem-dependent parameters in the nonlinear programming solver

Colors: — enables changes of foreground and background colors or attributes of all items displayed on the screen during the work of the program in an easy, interactive way; all changes are immediately visible on the screen (see Appendix B for details)

Accuracy. — changes accuracy of optimization; if this value is too high, then results may be misleading, if this value is too low, then optimization time may be too large

Violation. — changes acceptable violation of bounds on outcomes; if this value is too high, then results may be misleading, if this value is too low, then optimization time may be too large

Iterations. — changes limit of iteration number (recalculations of the model) during each optimization run

Scaling exponent. — changes coefficient α in the scalarizing function

Ratio Asp./Res. — changes ratio of the width of aspiration and reservation ranges for stabilized objectives

3.6.2 Function keys for model edition

< F1 > — context sensitive help.

< F2 > — save. — saves model (if changed), problem (if changed) and result (if changed); asks for names (if not named previously)

< F3 > — calculate. — the same as < Alt C >

< F4 > — list. — the same as < Alt L >

< F6 > — go to the interactive analysis spreadsheet.

< F7 > — totals on/off. — the same as < Alt S T >

< F8 > — auto on/off. — the same as < Alt S A >

< F10 > — exit to DOS.

3.6.3 Menu for interactive analysis

< Alt M > **Model selection:** — the same as in the model edition spreadsheet

< Alt P > **Problem selection:** — commands for operation on problems as whole entities

Get from disk: — displays menu of currently defined problems (for current model), selected problem is loaded from a disk into the spreadsheet

Fix and save. — checks the completeness of the problem, fixes it and saves on a disk (asks for problem name if not named previously)

Description. — asks for brief description of the problem (up to 30 characters long)

New. — un-fixes the problem, resets the name and the description; the problem in the spreadsheet remains unchanged

Reset. — un-fixes the problem, resets the name and the description, deletes all problem data in the spreadsheet

Utopia. — checks the completeness of the problem and calculates the utopia point, approximates the nadir point (see theoretical manual)

n**A**dir. — enters special spreadsheet editing mode that enables user updates of the nadir point values, < Esc > key exits this mode

neu**T**ral. — checks the completeness of the problem and calculates the neutral solution (see theoretical manual)

Erase: — displays menu of currently defined problems (for current model); selected problem is deleted together with all related results

Print. — prints current problem

< Alt R > **R**esult selection: — commands for operations on results as whole entities

Get from disk: — displays menu of currently defined results (for current model and for current problem); selected result is loaded from a disk into the spreadsheet

Save and new. — checks the completeness of the result, saves it on a disk (asks for result name if not named previously) and resets the name and comment

Description. — asks for brief description of the result (up to 30 characters long)

New. — resets the name and the description; the result data in the spreadsheet remains unchanged

Reset. — resets the name and the description, clears all result data in the spreadsheet

Calculate. — checks the completeness of the result data and calculates the efficient solution (see theoretical manual)

Variables. — displays a window with values of variables related to current efficient solution

Erase: — displays menu of currently defined results (for current model and for current problem); selected result is deleted

Print. — prints current result

< Alt G > **Graphics:** — selects the results and objectives to be displayed, switches the scaling method and starts the display

Display. — displays the graphical representation of the results

Result selection: — displays menu of currently defined results (for current model and for current problem); selected results are displayed on the screen, < Enter > key selects (up to 10 results can be selected), < Esc > key moves back to the Graphics menu

Objectives selection: — displays menu of objectives; selected objectives are displayed on the screen, < Enter > key selects, < Esc > key moves back to the Graphics menu

Normal scale — toggles the scaling methods

< Alt O > **Options:** — the same as in the model edition spreadsheet

3.6.4 Function keys for interactive analysis

< F1 > — context sensitive help.

< F2 > — save. — the same as in the model edition spreadsheet

< F3 > — calculate. — calculates utopia point (if not calculated), neutral solution (if not calculated) and efficient solution (if not calculated), performs all necessary checks the completeness of model, problem and result data

< F4 > — list. — the same as < F4 > or < Alt L > in the model edition spreadsheet

< F5 > — go to the model edition spreadsheet.

< F9 > — graphics. — displays the graphical representation of the results using either default selection rules or last user-defined selection

< F10 > — exit to DOS.

3.7 Syntax of formulae

Outcome formulae entered into the spreadsheet are standard arithmetic expressions with some possible extensions. Five binary arithmetical operators can be used: addition '+', subtraction '-', multiplication '*', division '/' and power '^' , moreover an unary minus can be used, with higher precedence than binary operators. Standard arithmetical rules are used for operator precedence and calculation order, parenthesis can be inserted to imply specific order of calculations. There is only one restriction for the use of these operators: a sequence of two power operators $x \hat{} y \hat{} z$ is not allowed — either operator

together with its arguments must by enclosed in parenthesis to explicitly define the order of calculations, i.e. it must be written as $(x \hat{} y) \hat{} z$ or $x \hat{} (y \hat{} z)$.

There are several built-in functions that can be used in outcome formulae, ten functions with one argument *abs*, *arctan*, *cos*, *exp*, *ln*, *log*, *signum*, *sin*, *sqr*, *sqrt* and two functions with two arguments *min*, *max*. Moreover, there is a predefined constant *Pi*. Functions *abs*, *signum*, *min*, *max* should be used with caution because they are nondifferentiable. Logical structures of the form *if logical expression then arithmetic expression else arithmetic expression* or *if logical expression then arithmetic expression elsif logical expression then arithmetic expression else arithmetic expression* should be used also with caution for the same reason. Up to ten levels of *elsif* are allowed. Relations of the form *a* < *b* (where *a* and *b* are any arithmetical expressions; possible relation operators are: '<' '<=' '=' '<>' '>=' '>') and of the form *a* in [*b*,*c*] (where *a*, *b* and *c* are any arithmetical expressions; '[' and ']' mean closed interval, '(' and ')' can also be used and mean open interval; possible forms are: 'in [.,.]' 'in [.,.)' 'in (.,.]' 'in (.,.)') and logical operators 'and' 'or' 'xor' 'not' are allowed in logical expressions.

4 Illustrative examples

4.1 Testing Example

This example has been chosen because its multiobjective analysis is simple and can be performed analytically. It serves to test the correctness of the installation of the program and to check whether the hardware and software IBM--PC compatibility of your computer is sufficient to use the DIDASN program.

The model has four variables (xa xb xc xd), two parameters (za zb) and three outcomes (obj1 obj2 wrk).

The model is defined as follows:

Outcome equations:

```
obj1  = (xa - 1) ^ 2 + za * (xb - 1) ^ 2 + (xc - 1) ^ 2 +
        (xd - 1) ^ 2 + wrk
obj2  = xa ^ 2 + za * xb ^ 2 + xc ^ 2 + xd ^ 2 + wrk
wrk   = zb * (xa - xb) ^ 2 + (zb - za) * (xc - xd) ^ 2
```

Bounds on variables and outcomes:

$$
\begin{array}{rcl}
-10 \leq & \text{xa} & \leq 10 \\
-10 \leq & \text{xb} & \leq 10 \\
-10 \leq & \text{xc} & \leq 10 \\
-10 \leq & \text{xd} & \leq 10 \\
-1 \leq & \text{obj1} & \leq 12 \\
-1 \leq & \text{obj2} & \leq 12 \\
0 \leq & \text{wrk} & \leq 100
\end{array}
$$

Values of parameters:

$$\mathtt{za} = 7$$
$$\mathtt{zb} = 100$$

Initial values of variables:

$$\mathtt{xa} = 1$$
$$\mathtt{xb} = 2$$
$$\mathtt{xc} = 3$$
$$\mathtt{xd} = 4$$

The multiobjective nonlinear programming problem is to minimize objectives `obj1` and `obj2`, while the outcome **wrk** is floating (free).

The Pareto frontier in the objective space for this example can be determined analytically and has the form:

$$\sqrt{\mathtt{obj1}/10} + \sqrt{\mathtt{obj2}/10} = 1$$

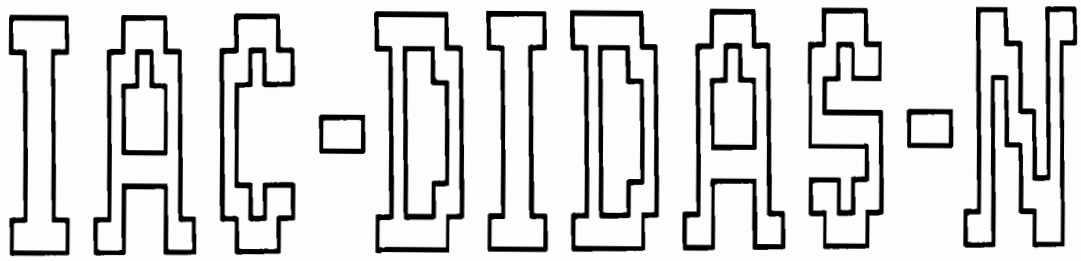
with the utopia point `( 0.0, 0.0 )`, nadir point `( 10.0, 10.0 )`, and the neutral solution point `( 2.5, 2.5 )`. Numerical results obtained during computation will be slightly different because of numerical errors and finite accuracy of calculations.

To go through the testing example, we will perform the following actions:

1. We activate the program `DIDASN` at the DOS prompt.
2. We get initial banner with the system name, a version number and an information about the authors (Fig. 1).
3. We press any key and get initial, smallest possible model editing spreadsheet (Fig. 2).
4. We load the model by pressing following keys:

< Alt M >	[model selection menu appears in the upper left corner of the screen]
< G >	[list of accessible models appears in the small window with the `DEMO` model name being the first; if it is not the first, then we select the `DEMO` name by moving the marking bar with < Up > and < Down > cursor keys]
< Enter >	[`DEMO` model is loaded and displayed]
< F5 >	[`DEMO` model is stored on a disk with the status 'fixed', thus the interactive analysis spreadsheet is automatically selected; we switch to the model edition spreadsheet (Fig. 3)]

5. Using the cursor movement keys we move to the row Values and to the column **xa**. The marked cell contains the current value of the variable **xa** — this value is 1.0. We enter a new value, just typing < 2 > and pressing the < Enter > key. The spreadsheet is immediately recalculated (Fig. 4).



Version 4.0 (October 1990)

Dynamic Interactive Decision Analysis and Support for Nonlinear models

developed for

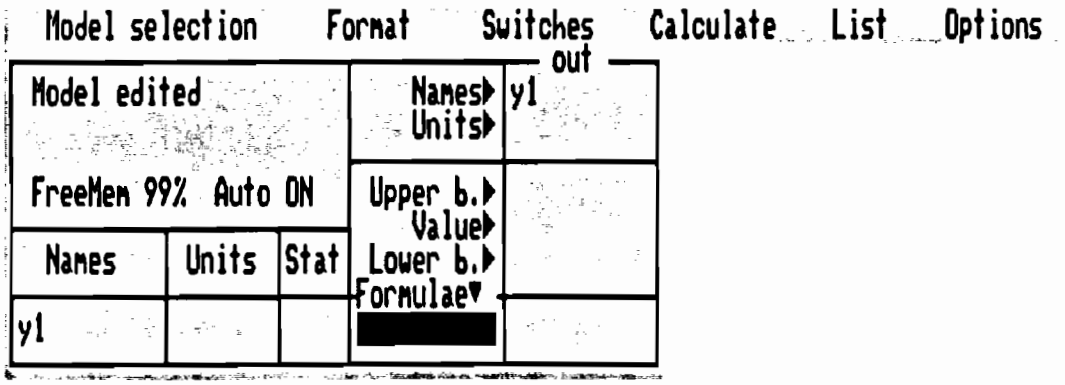
International Institute
for Applied Systems Analysis
Laxenburg, Austria

by

T.Kreglewski , J.Granat
Institute of Automatic Control
Warsaw University of Technology
Warsaw, Poland

Hit any key to continue

Figure 1: Initial screen



F1-Help F2-Save F3-Calculate F4-List F6-Multiobjective analysis F10-Exit

Figure 2: Initial spreadsheet

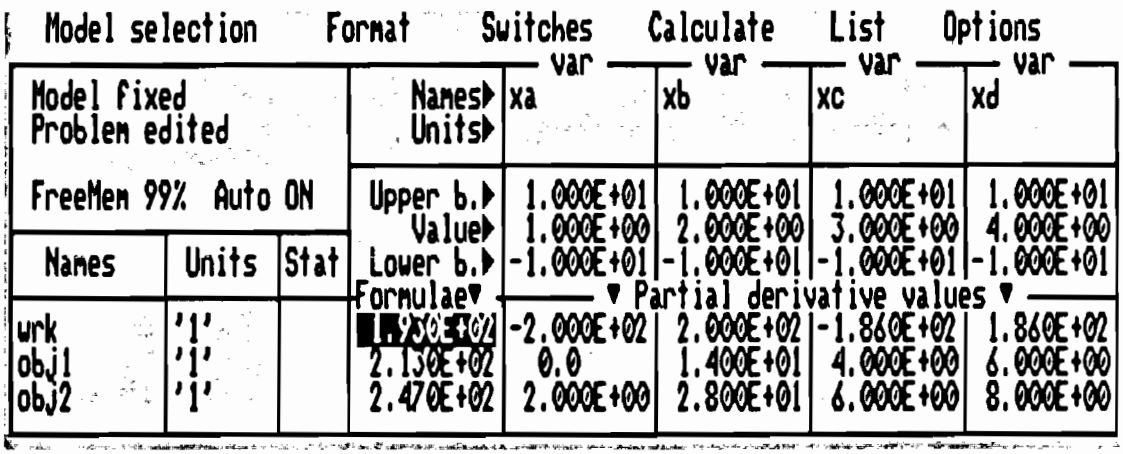


Figure 3: DEMO model loaded

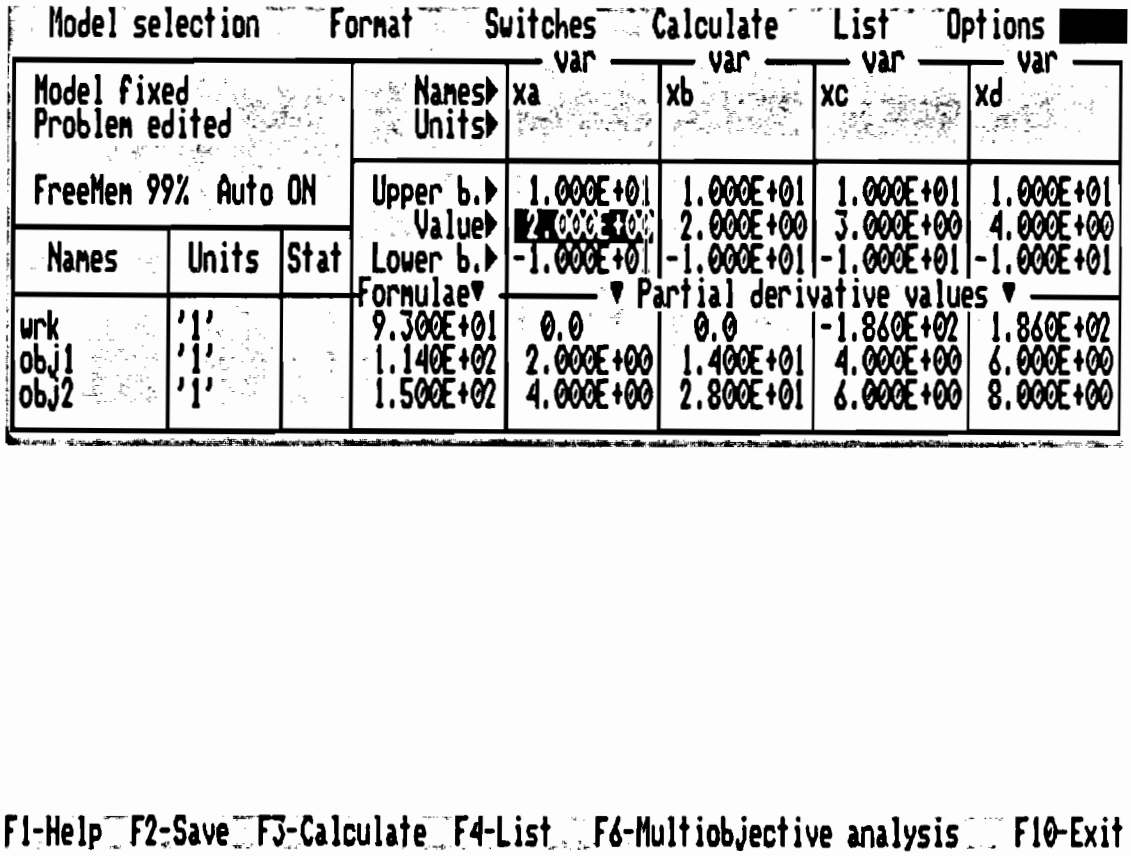


Figure 4: DEMO model recalculated

Model selection Problem selection Result selection Graphics Options

Model fixed
Problem fixed
Result calculated
FreeMem 99% Auto ON

Model: DEMO ▶ IAC-DIDAS-N testing example ◀
Problem: DEMO1 ▶ IAC-DIDAS-N problem example ◀
Result: Neutral ▶ ◀

Names	Units	Stat	Utopia	Asp. level	Neutral Solution	Res. level	Nadir
wrk	'1'				2.730E-15		
obj1	'1'	min	5.218E-27	1.667E+00	2.500E+00	3.333E+00	1.000E+01
obj2	'1'	min	6.613E-26	1.667E+00	2.500E+00	3.333E+00	1.000E+01

F1-Help F2-Save F3-Calculate F4-List F5-Model editing F9-Graphics F10-Exit

Figure 5: DEMO1 problem loaded

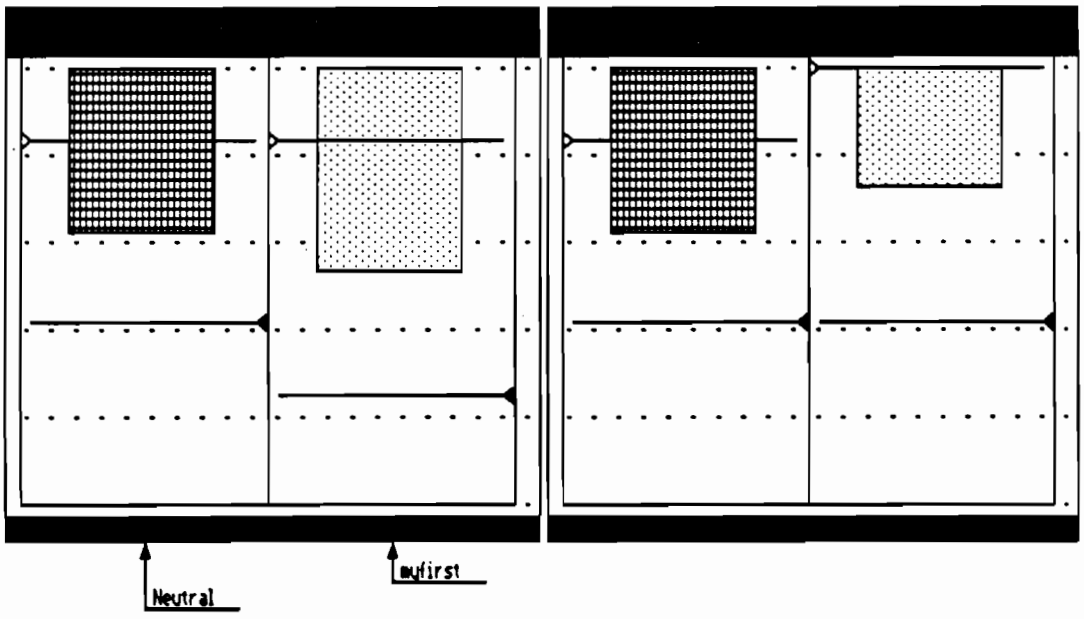


Figure 6: Graphical comparison

6. Now we switch to the interactive analysis spreadsheet by pressing the < F6 > key and we load example problem definition DEMO1 for the model DEMO by pressing < Alt P > < G > < Enter > keys. The problem definition together with utopia and nadir points, neutral solution and some proposed aspiration and reservation levels are then displayed (Fig. 5).

7. Now we will try to recalculate the problem and obtain the same results. We begin with the definition of a new problem by pressing < Alt P N > keys. Next we press < Alt P U > keys and the system determines the utopia point (analytical solution is (obj1, obj2) = (0.0, 0.0), obtained values are (3.944E-31, 3.790E-38) and the nadir point (analytical solution is (obj1, obj2) = (10.0, 10.0), obtained results are the same).

8. Let the system determine the neutral solution — we press < Alt P T > (It is also possible to press single function key < F3 > instead of < Alt P N > and < Alt P T > to calculate in sequence utopia point, nadir point and neutral solution). Analytical solution is x = (0.5, 0.5, 0.5, 0.5), (wrk, obj1, obj2) = (0.0, 2.5, 2.5), obtained results for obj1 and obj2 are exactly the same. We check results for variables by using the command < Alt R V > — they are exactly equal to the analytical ones. Now we fix the problem (by pressing < Alt P F > keys and by giving it any valid name — e.g. DEMO2). Before doing it we can submit some problem description (by < Alt P D >).

9. Let the system determine an efficient solution corresponding to aspiration level of objective obj1 changed from 1.6667 to 1.0 and reservation level of objective obj2 changed from 3.3333 to 4.0; we move the spreadsheet marker to the respective cells and enter new values. The < F3 > function key initiates calculations. After a while we check obtained results; they are 2.124 and 2.906 for objectives obj1 and obj2, respectively, and 9.313E-18 for outcome wrk. We check results for variables — they are 0.539114, 0.539114, 0.539114, 0.539114. We save obtained result (by < Alt R S > and by giving it any valid name — e.g. MYFIRST). Before doing it we can submit some result description (by < Alt R D >).

10. Now we can compare two already obtained results by using graphical representation — for this purpose it is enough to press < F9 > function key. We obtain a screen with bars representing our results, it is better to change the scaling method by pressing the key < F2 > (Fig. 6).

4.2 Tutorial example

Typical procedure of working with the DIDASN program and various aspects of the use of several program commands are discussed in this section. This discussion is done by exploiting a real-life example specially designed for this purpose. The model used in this example is a very rough approximation of the much more complicated model of acid deposition in forest soil, described by Hettelingh and Hordijk (1987).

4.2.1 Description of the model

We consider two regions (denoted by the index $k = 1, 2$) burning one type of fuel (say, coal) and emitting sulphur dioxide. The problem is, in fact, a dynamic one and should be

considered in many one year time periods; here we simplify it by considering only three periods (denoted by $t = 1, 2, 3$), each of them five years long.

The sulphur dioxide emission in each region and time period is determined by:

$$S_{k,t} = S^p_{k,t}(1 - p_{k,t}) \tag{38}$$

where $S^p_{k,t}$ is the potential emission, specified exogenously. It may be described in the form:

$$S^p_{k,t} = E_{k,t}\frac{z_{k,t}}{h_{k,t}}(1 - r_{k,t}) \tag{39}$$

where $E_{k,t}$ is the total energy production in region k and in time period t, $h_{k,t}$ is the heat content of the fuel, $z_{k,t}$ is the sulphur content of the fuel, $r_{k,t}$ is the reduction coefficient resulting from sulphur remaining in ashes. However, for the purpose of this simplified model, $S^p_{k,t}$ are assumed to be given as model parameters (for $k = 1, 2$ and $t = 1, 2, 3$). In the computerized model $S^p_{k,t}$ are denoted as parameter names **Sktp** and $S_{k,t}$ are denoted as outcome names **Skt** where k are digits $1, 2$ representing two regions and t are digits $1, 2, 3$ representing three time periods.

The reduction coefficients $p_{k,t}$ in (38) describe the effects of the pollution control measures. These coefficients serve as the main decision variables, therefore, there are actually six decision variables $p_{k,t}$ $(k = 1, 2,\ t = 1, 2, 3)$. In the computerized model they are denoted as variable names **pkt**.

It is assumed that the decision maker in k-th region is interested in:

- the costs of pollution control measures $C_{k,t}$ for each period;
- the level of pH (denoted here as $pH_{k,t}$ and denoted as outcome names **pHkt** in the model) in forest soil for each period;

or in two objective outcome trajectories each of three period length. In the DIDAS methodology, however, we investigate cooperative actions of both decision makers, therefore, the joined "decision maker" is interested in four outcome trajectories — two cost trajectories and two pH trajectories each of three period length, representing together twelve objective outcomes.

The cost $C_{k,t}$ (denoted in the model as outcome names **Ckt**) is function of the potential emission $S_{k,t}$ and of the reduction coefficient $p_{k,t}$. Actually, the situation is more complicated, since the costs have also dynamic character: there is a high investment cost of pollution control devices, but these devices are not so expensive in maintenance; on the other hand, once installed, the devices give defined coefficient $p_{k,t}$. However, when considering only five year periods we can apply much simpler model of the pollution control costs, understood as joined cost of investments and maintenance for five year period and dependent on the average reduction coefficient achieved during this period:

$$C_{k,t} = c_k S^p_{k,t}\frac{p_{k,t}}{2(1 - p_{k,t})} \tag{40}$$

where c_k is the cost of reducing the emission by half per one unit of potential emission. This is a very simple approximation of actual cost curves and it can be replaced by any other more exact approximation. The form of this approximation express, however, the fact that it becomes increasingly costly to obtain reduction coefficients close to 1 . Because of numerical reasons, the reduction coefficient mustn't be close to 1, thus it should be constrained in a range, say, $0 \leq p_{k,t} \leq 0.99$ (in the computerized model reduction coefficients are measured in percents and bounded from 0% to 99%).

The level of pH in forest soil is assumed to have more long-time dynamic aspects and thus it is modeled by dynamic equations. When approximating more complicated relations described in (Hettelingh and Hordijk, 1987), we must take into account that acid absorption and reduction capacities of forest soil are nonlinear, that they are the strongest in the carbonate range (pH = 8.0–6.2, but we will take pH = 7 as an upper bound), quite different and not so strong in the silicate range (pH = 6.2–5.0) and again the stronger in the cation exchange range (pH = 5.0–4.2) while any pH level below 4.0 might be considered as catastrophic. Therefore, instead of including more realistic and complicated models that may be considered in further variants of this application, in the tutorial example we consider only a nonlinear dynamic model for the pH range 7–4 of the approximate form:

$$pH_{k,t} = \left(pH_{k,t-1} + \alpha_k\,(7 - pH_{k,t-1})\right)\left(1 - \frac{3}{7}\,\Phi\left(\frac{D_{k,t}}{CAP_k}\right)\right) \tag{41}$$

where $D_{k,t}$ (denoted in the model as outcome names `DktR`) are sulphur deposits in given region and period, CAP_k are the five year carrying absorption capacities (if $D_{k,t} \geq CAP_k$ then it is assumed that $pH_{k,t}$ drops to 4 or below), and the function Φ expresses the essential nonlinearity of absorption and reduction of acid by forest soil. A convenient form of this function is:

$$\Phi(x) = \begin{cases} 0, & \text{if} \quad x \leq 0 \\ 3x^2 - 2x^3, & \text{if} \quad 0 \leq x \leq 1 \\ 1, & \text{if} \quad x \geq 1 \end{cases} \tag{42}$$

This is a twice-differentiable (except at $x = 0$ and $x = 1$, where it is only once-differentiable) spline function.

If $x = D_{k,t}/CAP_k = 0$, then the dynamic part of (41) illustrates the self-regeneration of forest soil with a regeneration coefficient α_k (during a five year period); this coefficient characterizes how many times the distance between pH=7 and the actual pH level will decrease in five years.

The initial value $pH_{k,0}$ (for $t = 0$) is given as a parameter for both regions $k = 1, 2$. It should be stressed again that the nonlinear dynamic model (41) is only a very rough approximation of actual forest soil chemistry and must be updated by specialists for more realistic policy analysis performed for other than only tutorial example purposes.

The sulphur deposits $D_{k,t}$ are the results of sulphur-emissions $S_{k,t}$ as determined by a deposition model, which, in this simplified case, is again assumed in the simplest possible form:

$$D_{k,t} = a_{k,1}S_{1,t} + a_{k,2}S_{2,t}, \qquad k = 1, 2 \tag{43}$$

where $a_{k,j}$ $(0 \leq a_{k,j} \leq 1)$ are transfer coefficients from region j to region k (for simplicity, we assume $a_{k,1} + a_{k,2} = 1$, $k = 1, 2$).

With such simplified model we can illustrate the issues of multiobjective dynamic and nonlinear analysis of the effects of pollution control. The analysts or the decision makers can jointly analyze in this model:

- what can be the maximal pollution reduction rates, if there are limited funds for pollution control in each of time periods, and what are the corresponding effects on forest soil acidity;
- what are the possibilities of multiobjective dynamic compromises between the trajectories of costs in all periods and trajectories of forest soil acidity.

For both purposes, the DIDAS methodology can be applied. The multiobjective analysis can be performed by specifying reference (aspiration or aspiration and reservation) trajectories for the costs and for the pH levels, while the DIDASN system can compute multiobjectively optimal (effective) trajectories for these variables that are consistent with the model (feasible) and, in a sense best, attuned to the reference trajectories.

4.2.2 Sample session

The model described in the previous section is already prepared as a disk file RAIN and can be loaded into the IAC-DIDAS-N spreadsheet by using the command < Alt M G > (Model selection — Get from disk). It is stored as a 'fixed' model, thus to make some experiments with the model we must use the command < Alt M N > (Model selection — New). Now we can change all upper and lower bounds, values of parameters and outcome formula. Following each change of variable or parameter value the spreadsheet is automatically recalculated. After some play with the model we load again the original one and start the second phase of the work.

We define the decision problem now. We select outcomes to be minimized or maximized. First we take into account only one region performances: we mark costs C11 C12 C13 as minimized and pH levels pH11 pH12 pH13 as maximized. Bounds on all outcomes are defined in the model and we don't redefine them. We ask the system to calculate Utopia and Nadir points by using the command < Alt P U > (Problem selection — Utopia) and after a while we get the results. Next we ask the system to calculate a compromise solution, so called neutral solution, being the starting point for further interaction. We enter the command < Alt P T > (Problem selection — neuTral) and again wait a while. When the neutral solution is calculated and displayed, the problem definition is finished and we can 'fix' the problem by using the command < Alt P F > (Problem selection — Fix and save). We are asked to give a name of the problem, it may be ONEREG. Basing on values of Utopia, Nadir and Neutral points the system proposes us initial values of aspiration and reservation levels. Further interaction consists in a sequence of three or four actions:

- modification of aspiration and/or reservation levels (it is enough to change only one value); it is obtained through the edition of appropriate spreadsheet cells.
- calculation of the efficient solution corresponding to current levels of aspirations and reservations. Optimization process is initiated by the use of the < F3 > (Calculate) command.
- if the result (efficient solution) is not satisfactory, we can discard it using the command < Alt R N > (Result selection — New) and go back to the first action. Otherwise, we save the result by using the command < Alt R S > (Result selection — Save and new).
- optionally, we can compare several results, obtained for current problem, using graphical representation; directly by the command < F9 > or with some selections of objectives and results to be displayed within < Alt G > (Graphics) menu.

Because all interesting results are stored on a disk, interaction session can be stopped at any time and next resumed.

Now we continue the interaction, but for the previously defined problem. We load the problem RAIN1 by using the command < Alt P G > (Problem selection — Get from

disk). The problem definition with calculated utopia and nadir values together with the neutral solution are loaded. There are twelve objectives now: costs C11 C12 C13 and pH levels pH11 pH12 pH13 for the first region and costs C21 C22 C23 and pH levels pH21 pH22 pH23 for the second region. Please observe, that neutral solution values for the first region are worse than in the previous problem. It is due to the fact that now the neutral solution is a compromise between interests of both regions.

We find, that costs in the second region are decisively too large, but pH in both regions can be accepted. Thus, we try to decrease costs in the second region by decreasing reservation levels for costs C21 C22 C23 from 2180 to 1500. We press < F3 > then we wait for the result and save it using the command < F2 >. To compare the result with the neutral solution we use graphical representation. First, we select objectives to be displayed — only ten of them can be displayed simultaneously. We enter the command < Alt G O > (Graphics — Objectives selection). The system displays the list of all twelve objectives with first ten being marked. Changes of pH in the last period are more important for us are than those in the first period. Therefore, we 'unmark' objectives pH11 pH21 and 'mark' objectives pH22 pH23. Both operations are performed by moving the marking bar with the < Up > and < Down > cursor keys and by pressing the < Enter > key. We needn't enter the Graphics — Results selection menu because currently there are only two results, that are automatically selected. Now we execute now the display command in the graphics menu and we obtain the bar representation of results. We can press the < F1 > (Help) function key to get help information on the meaning of several elements of the picture. However, we find the picture rather not legible. The standard scaling method (Normal scale ON) is based on the distances from lower to upper bounds. In our case, the changes of efficient values are much smaller than these distances. Thus, it seems that the second (Normal scale OFF) scaling method will be useful — it is based on distances between utopia and reservation values (or solution values if they are worse than reservations). The < F2 > key in the graphical representation toggles between both scaling methods; we press this key once.

Now the picture is legible and we can see that, in fact, the costs in the second region are decreased, but simultaneously the costs in the first region are increased and the pH levels are decreased. Now we can either increase back reservations for the costs in the second region or increase aspirations and/or reservations for the pH levels in both regions. We try to explore the second possibility. We increase reservations for pH12 pH13 from 5.983 to 5.990 and from 6.265 to 6.270, respectively, and we calculate the efficient solution again, save it and look at the graphical representation — we press in sequence three function keys: < F3 > (Calculate), < F2 > (Save) and < F9 > (Graphics). The previous selection of objectives is still active and now three results are displayed.

The pH levels are now acceptable, but the costs in the first region are very high. To balance the costs in both regions we slightly increase reservations in the second region (from 1500 to 1700) and we decrease reservations in the first region (from 1090 to 900). The fourth obtained result seems to be close to the acceptable solution of the multiobjective decision problem.

5 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 Eco-

nomic and Mathematical Systems 248).

Hettelingh, J. P. and L. Hordijk (1987). Environmental Conflicts: The Case of Acid Rain in Europe. RR-87-9, International Institute for Applied Systems Analysis, Laxenburg, Austria.

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 Modeling 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 Modeling 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).

Lewandowski, A., T. Kreglewski, T. Rogowski, A. P. Wierzbicki (1987). Decision Support Systems of DIDAS Family. In A. Lewandowski, A. P. Wierzbicki (eds), Theory, Software and Testing Examples for Decision Support Systems. WP-87-26, International Institute for Applied Systems Analysis, Laxenburg, Austria.

Lewandowski, A., A. P. Wierzbicki (1988). Aspiration Based Decision Analysis and Support, Part I: Theoretical and Methodological Backgrounds. WP-88-3, International Institute for Applied Systems Analysis, Laxenburg, Austria.

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 Modeling 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 Modeling* 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.

A Installation guide

The distribution diskette contains the following files:

`DIDASN.EXE`	— compiled code of the program
`DEMO.MOD`	— simple nonlinear model (testing example)
`RAIN.MOD`	— nonlinear model used in Tutorial Example
`READ.ME`	— last time notes and corrections
`INSTALL.BAT`	— batch command file to install the program and both models on a hard disk or on a working diskette.

To install the program and models:

- insert the distribution diskette in floppy disk drive,
- change current drive and current directory to the drive and directory where you want to install the program,
- enter the command
  `a:install a:`
  where `a` is a drive letter of the drive where the distribution diskette is inserted (typically it is just drive A)

Installation procedure makes the sub directory `DIDASN`, writes the program code (`DIDASN.EXE`) there, next makes subdirectory `DIDASN\MODELS`, and writes into it two files with demonstrative nonlinear models together with examples of problems and results.

B Selection of colors

The program can work on `IBM PC/XT/AT` or compatible computers with Hercules Graphics Card (`HGC`), Color Graphics Adapter (`CGA`) and Enhanced Graphics Adapter (`EGA` or `VGA`). The user can select (independently for each particular type of a graphics card) his own set of colors/attributes to display several items on the screen by using menu entry Options — command Colors.

There are 9 items with some restrictions on selections of their colors:

Spreadsheet cells
Spreadsheet lines
Spreadsheet labels
Spreadsheet marker
Editing windows
Message windows
Warning messages
Error messages
Screen background

First three items form spreadsheets and, therefore, they have the same background color, the spreadsheet marker should have color that make it visible in all spreadsheet cells (empty and not empty).

Highlighted double-triangles marker selects one of 9 items and can be moved with < Up > and < Down > cursor keys. For the selected item < Home > and < End > cursor keys change foreground color, whereas < PgUp > and < PgDn > cursor keys change background color. The selected item is displayed in the colors/attributes currently choosen for it. Simultaneously names of the colors (such as displayed on the CGA adapter) are displayed on the right.

< Esc > key cancels all colors changes and causes return to the spreadsheet. < Enter > key makes the changes effective; additionally, the program asks about making these changes permanent. In case of answer < Y > (or < y >) the changes are recorded on a disk and will also be effective during subsequent runs of the program.

C Alternative solver

New solver based on nondifferentiable optimization algorithms from NOA1 software package[2] was tested as an alternative to the standard one described in the Theoretical manual. Currently, it was implemented using non-standard hardware (OA-Link boards) as a remote solver running on different computer and interchanging all necessary data with the model created and calculated in the IAC-DIDAS-N system. The first results are hopeful, therefore, the NOA1 solver will be included in the future versions of the system. However, the NOA1 package is written in FORTRAN and cannot be directly connected to the IAC-DIDAS-N system written in TURBO PASCAL. Because of the size of both software packages some distributed, network hardware and software must be used.

[2]see: K. Kiwiel and A. Stachurski (1988). NOA1: A FORTRAN Package of Nondifferentiable Optimization Algorithms Methodological and User's Guide, WP-88-116. International Institute for Applied Systems Analysis, Laxenburg, Austria.

Dynamic Interactive Network Analysis System — DINAS version 3.0 (1990) User's Manual

Włodzimierz Ogryczak, Krzysztof Studziński

Krystian Zorychta

Institute of Informatics, Warsaw University.

Abstract

This paper describes the methodological background and user manual of the Dynamic Interactive Network Analysis System (DINAS) which enables the solution of various multiobjective transshipment problems with facility location using IBM-PC XT/AT microcomputers. DINAS utilizes an extension of the classical reference point approach to handling multiple objectives. In this approach the decision-maker forms his requirements in terms of aspiration and reservation levels, i.e., he specifies acceptable and required values for given objectives. A special TRANSLOC solver was developed to provide 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 simple and variable upper bounds (VUB & SUB). DINAS is prepared as a menu-driven and easy in usage system armed with a special network editor which reduces to minimum effort associated with input a real-life problem.

Version 3.0 is highly compatibile with the previous one. Differences between these versions are small, resulting from the introduction of some new features into the system.

1 Introduction

DINAS (Dynamic Interactive Network Analysis System) is a scientific transferable software tool which enables the solution of various multiobjective transshipment problems with facility locations. For a given number of fixed facilities and customers and for a number of potential facilities to be optionally located, DINAS provides you with a distribution pattern of a homogeneous product under multicriteria optimality requirements. While working in an interactive mode, you get optimal locations of the potential facilities and a system of optimal flows of the product between nodes of the transportation network.

With DINAS you can analyse and solve such problems as:

- the transportation problem with new supply and/or demand points location,
- the problem of warehouses location,

- the problem of stores location for the agricultural production,
- the problem of service centers location and districts reorganization,

and many other real–life distribution–location problems.

DINAS is implemented on IBM–PC XT/AT as a menu–driven and easy in usage system armed with a special network screen editor for a friendly data input and results examination. While working with DINAS you will get a permanent assistance by the help lines which will inform you about operations available at this moment. Moreover, at any moment you will have opportunity to get more general information from the help file.

This manual is organized as follows. In Chapter 2 a short characteristic of the system is given. There is also included the problem statement.

Chapter 3 provides you with the theoretical and methodological backgrounds of the DINAS system. It describes in details the mathematical formulation of the problem, the interactive procedure for handling multiple objectives as well as some sophisticated techniques of the solver. This chapter can be skipped by the users who are interested only in learning functions and operations of DINAS without a deep knowledge of the methodology.

Chapter 4 takes a part of a tutorial. The purpose of it is to help a new user become familiar with the DINAS system. It does not provide you with any formal commands description. It rather shows capabilities of the system. In this chapter we present in details using of DINAS to analyse a tutorial problem. The tutorial problem is constructed as a small part of the real–life problem of health service districts reorganization connected with a location of new health–care centers.

Chapters 5 and 6 describe in details all the operations performed in the DINAS system. For each operation they explain, in a rather formal way, the purpose of the operation, the execution process as well as troubleshooting. Thus these chapters can be regarded as an extended reference manual. Chapter 5 presents the main system operations whereas the operations connected with problem input and editing are considered in Chapter 6.

2 General information

2.1 The problem statement

DINAS works with problems formulated as multiobjective transshipment problems with facility location. A network model of such a problem consists of nodes 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, whereas 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 etc.). For this reason, potential nodes are organized in the so–called selections, i.e., sets of nodes with the multiple choice requirements. 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).

A homogeneous good is distributed along the arcs among the nodes. Each fixed node is characterized by two quantities: supply and demand on the good, but for the mathematical statement of the problem only the difference supply–demand (the so–called balance) is used. Each potential node is characterized by a capacity which bounds maximal good flow through the node. The capacities are also given for all the arcs but not for the fixed nodes.

A few 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 (the so–called cost coefficients, independently of their real character). The cost coefficient connected to an arc is treated as the unit cost of the flow along the arc. The cost coefficient connected to a potential node is considered as the fixed cost associated with locating of the node (e.g., an investment cost).

Summarizing, the following groups of input data define the transshipment problem under consideration:

- objectives,
- fixed nodes with their supply-demand balances,
- potential nodes with their capacities and (fixed) cost coefficients,
- selections with their lower and upper limits on number of active potential nodes,
- arcs with their capacities and cost coefficients.

In the DINAS system there are two restrictions on the network structure:

- there is no arc which directly connects two potential nodes;
- each potential node belongs to at most two selections.

The first restriction does not imply any loss of generality since each of two potential nodes can be separated by an artificial fixed node, if necessary. The second requirement is not very strong since in practical models usually there are no potential nodes belonging to more than two selections.

The problem is to determine the number and locations of active potential nodes and to find the good flows (along arcs) so as to satisfy the balance and capacity restrictions and, simultaneously, optimize the given objective functions. A mathematical model of the problem is described in details in Section 3.1.

2.2 DINAS and Multiple Criteria Decision Making

The problem under consideration is a specialized form of Multiple Criteria Decision Making (MCDM). The basic concept of the multiple objective optimization was introduced by Pareto over 80 years ago. He developed the idea of the so–called efficient (or Pareto–optimal) solution, that is a solution which cannot be improved in any objective without some other objective being worsened. However the set of all the efficient solutions is, in practice, extremely huge. Therefore development of practical MCDM tools has begun from the 70's when the computer technique reached a sufficient level for an efficient implementation of various interactive (decision support) systems.

The interactive system does not solve the multiobjective problem. It rather makes the user selecting the best solution during interactive work with the system. According to some user's requirements, the system generates various efficient solutions which can be examined in details and compared to each other. The user works with the computer in an interactive way so that he can change his requirements during the sessions.

DINAS is such an interactive decision support system. The DINAS interactive procedure utilizes an extension of the reference point optimization. The basic concept of that approach is as follows:

- the user forms his requirements in terms of aspiration and reservation levels, i.e., he specifies acceptable and required values for given objectives;

- the user works with the computer in an interactive way so that he can change his aspiration and reservation levels during the sessions.
- after editing the aspiration and reservation levels, DINAS computes a new efficient solution while using an achievement scalarizing function as a criterion in single-objective optimization (see Section 3.2 for more details).
- each computed efficient solution is put into a special Solution Base and presented to the DM as the Current Solution in the form of tables and bars which allow him to analyse performances of the Current Solution in comparison with the previous solutions.

A special TRANSLOC solver is included in the system to provide the multiobjective analysis procedure with optimal solutions to single–objective problems. The solver is hidden from the user but it is the most important part of the DINAS system. It is the numerical kernel of the system which generates efficient solutions. The concept of TRANSLOC is based on the branch and bound scheme with a pioneering implementation of the simplex special ordered network (SON) algorithm and with implicit representation of the simple and variable upper bounds (VUB & SUB). The mathematical backgrounds of the TRANSLOC solver are given in Chapter 3.

2.3 Differences between version 2.1 and 3.0

Despite several changes to the system both the versions are highly compatible. As you will see by reading through this manual most of the differences are small, resulting from the introduction of some new features. With the version 3.0 you will get additionally:

- additional windows in the network editor allowing to examine and modify data while visiting the list of nodes and the network scheme;
- additional approximative way of defining the user requirements without strenuous editing their numerical values;
- using mouse installation to control running of the system;
- generating standard MPS data file to use mainframe proffesional solvers for solving problems edited with the DINAS network editor.

3 Theoretical and methodological backgrounds of the system

3.1 The internal network model

For simplicity of the model representation and the solution procedure the network problem is transformed in DINAS into a special internal form. Namely, the potential nodes are transformed into artificial arcs.

The transformation is performed by duplication of all potential nodes. After the duplication is done 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 the 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:

$$\textit{minimize} \quad \sum_{(i,j)\in A\setminus A_a} f_{ij}^p x_{ij} + \sum_{(i,j)\in A_a} f_{ij}^p y_{ij}, \qquad p = 1,2,\ldots,n_0 \tag{1}$$

$$\textit{subject to} \quad \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i, \qquad i \in N \tag{2}$$

$$0 \le x_{ij} \le c_{ij}, \qquad (i,j) \in A \setminus A_a \tag{3}$$

$$0 \le x_{ij} \le c_{ij} y_{ij}, \qquad (i,j) \in A_a \tag{4}$$

$$g_k \le \sum_{(i,j)\in S_k} y_{ij} \le h_k, \qquad k = 1,2,\ldots,n_s \tag{5}$$

$$y_{ij} = 0 \text{ or } 1, \qquad (i,j) \in A_a \tag{6}$$

where the following notations are used:

n_0	number of objective functions,
N	set of nodes (including copies of potential nodes),
n_s	number of selections,
A	set of arcs (including artificial arcs),
A_a	set of artificial arcs,
f_{ij}^p	cost coefficient of the p–th objective associated with the arc (i,j),
b_i	supply-demand balance at the node i,
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 (2) with simple upper bounds (3) as well as a special discrete structure (5) — (6) connected to the network structure by variable upper bounds (4). While solving the model we have to take advantages of all these features.

3.2 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 which was introduced by Wierzbicki (1982). This concept was further developed in many papers and was used as a basis for construction of the software package DIDAS (Dynamic Interactive Decision Analysis and Support system). The DIDAS package proved to be useful in analysing conflicts and assisting in decision making situations (Grauer et al., 1984).

The basic concept of the reference point 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 sessions of the analysis.

In our system, we extend the DIDAS approach. The extension relies on additional use of reservation levels which allow the DM to specify necessary values for given objectives (Wierzbicki, 1986).

Consider the multi–objective program associated with the generalized network model:

$$\begin{array}{ll} \textit{minimize} & q \\ \textit{subject to} & q = F(x,y) \\ & (x,y) \in Q \end{array}$$

where

q represents the vector,

F is the linear objective vector–function defined by (1),

Q denotes the feasible set of the generalized network model, i.e., the set defined by conditions (2)—(6).

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, the following single objective programs are solved:

$$\min\{F^p(x,y) + \frac{r_0}{n_0}\sum_{i=1}^{n_0} F^i(x,y) : (x,y) \in Q\}, \qquad p = 1,2,\ldots,n_0 \tag{7}$$

where F^p denotes the p–th objective function and r_0 is an arbitrarily small number.

The so–called pay–off matrix

$$R = (q_{pj}), \qquad p = 1,\ldots,n_0; \qquad j = 1,\ldots,n_0$$

which yields information on the range of numerical values of each objective is then constructed. The p–th row of the matrix R corresponds to the vector (x^p, y^p) which solves the p–th program (7). 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 R, 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.

Taking into consideration the j–th column of the matrix R we notice that the minimal value in that column is $q_{pp} = q_p^u$.

Let q_j^n be the maximal value, i.e.,

$$q_j^n = \max_{1 \le p \le n_0} \{q_{pj}\}$$

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

$$q^u \leq q^a < q^r \leq q^n$$

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:

$$\begin{aligned} &\textit{minimize} && \max_{1\leq p\leq n_0} u_p(q,q^a,q^r) + \frac{r_0}{n_0}\sum_{p=1}^{n_0} u_p(q,q^a,q^r) \\ &\textit{subject to} && q = F(x,y) \\ &&& (x,y)\in Q \end{aligned} \tag{8}$$

where r_0 is an arbitrarily small number and u_p is a function which measures the deviation of results from the DM's expectations with respect to the p–th objective, depending on a 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 the DM needs.

The function $u_p(q,q^a,q^r)$ is a strictly monotone function of the objective vector q with value $u_p = 0$ if $q = q^a$ and $u_p = 1$ if $q = q^r$. In our system, we use (similarly as in Wierzbicki 1986) a piece–wise linear function u_p defined as follows:

$$u_p(q,q^a,q^r) = \begin{cases} a_p(q_p - q_p^a)/(q_p^r - q_p^a), & \text{if } q_p < q_p^a \\ (q_p - q_p^a)/(q_p^r - q_p^a), & \text{if } q_p^a \leq q_p \leq q_p^r \\ b_p(q_p - q_p^r)/(q_p^r - q_p^a) + 1, & \text{if } q_p^r < q_p \end{cases}$$

where a_p and b_p $(p = 1,2,\ldots,n_0)$ are given positive parameters. In the DINAS system, the parameters a_p and b_p are defined according to the formulae

$$a_p = \frac{a(q_p^r - q_p^a)}{(q_p^a - q_p^u)}$$

$$b_p = b(q_p^r - q_p^a)$$

where a and b are positive parameters computed as follows

$$a = 0.1 \min_{1\leq j\leq n_0} \frac{(q_j^a - q_j^u)}{(q_j^r - q_j^a)^2}$$

$$b = 10 \max_{1\leq j\leq n_0} \frac{1}{(q_j^r - q_j^a)}$$

The parameters a_p and b_p satisfy inequalities: $a_p < 1$ and $b_p > 1$, and thereby the achievement functions u_p are convex. Minimization of the function u_p is then equivalent

to minimization of a variable u_p defined as follows:

$$\begin{aligned} u_p &= v_p + b_p v_p^- - a_p v_p^+ && (9)\\ v_p - v_p^+ + v_p^- &= (q_p - q_p^a)/(q_p^r - q_p^a) && (10)\\ 0 \le\ v_p &\le 1 && (11)\\ v_p^+ \ge 0, &\quad v_p^- \ge 0 && (12) \end{aligned}$$

3.3 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 3.2 the TRANSLOC solver has to be able to solve two kinds of single–objective problems: the first one associated with calculation of the pay–off matrix (problems (7)) and the second one associated with minimization of the scalarizing achievement function (problems (8)). 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 the 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:

$$\begin{aligned} &\textit{maximize} && s && (13)\\ &\textit{subject to} && \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i, \quad i \in N && (14)\\ & && w_k + \sum_{(i,j)\in S_k} y_{ij} = h_k, \quad k = 1,2,\ldots,n_s && (15)\\ & && u_p - v_p + d_p^+ - d_p^- = 0, \quad p = 1,2,\ldots,n_0 && (16)\\ & && v_p - \frac{1}{a_p} d_p^+ + \frac{1}{b_p} d_p^- - \sigma_p \left(\sum_{(i,j)\in A\setminus A_a} f_{ij}^p x_{ij} + \sum_{(i,j)\in A_a} f_{ij}^p y_{ij} \right) = \delta_p, && \\ & && \qquad p = 1,\ldots,n_0 && (17)\\ & && u_0 - \sum_{p=1}^{n_0} u_p = 0 && (18)\\ & && s + z + \frac{r_0}{n_0} u_0 = 0 && (19)\\ & && 0 \le x_{ij} \le c_{ij}, \quad (i,j) \in A \setminus A_a && (20)\\ & && 0 \le w_k \le h_k - g_k, \quad k = 1,2,\ldots,n_s && (21)\\ & && x_{ij} \le c_{ij} y_{ij}, \quad (i,j) \in A_a && (22)\\ & && u_p \le z, \quad p = 1,2,\ldots,n_0 && (23)\\ & && y_{ij} = 0 \text{ or } 1, \quad (i,j) \in A_a && (24) \end{aligned}$$

and depending on the kind of optimization:

$$d_p^+ = 0, \quad d_p^- = 0, \quad p = 1,2,\ldots,n_0 \qquad (25)$$

for the utopia point calculation or

$$d_p^+ \ge 0, \quad d_p^- \ge 0, \quad 0 \le v_p \le 1, \quad p = 1,2,\ldots,n_0 \qquad (26)$$

for the achievement scalarizing function optimization, respectively, where: $\sigma_p = 1$ and $\delta_p = 0$ during utopia point calculation, $\sigma_p = 1/(q_p^r - q_p^a)$ and $\delta_p = -q_p^a/(q_p^r - q_p^a)$ during

the minimization of the achievement scalarizing function, whereas all the other quantities are the same as in Sections 3.1 and 3.2.

The above single-objective problem is a typical mixed integer linear program, i.e., it is a typical linear program with integrality 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 a 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.

Even for a small transshipment problem with facility location, the corresponding linear program (13) – (23) 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/AT microcomputers, it is necessary to take advantages of its special structure.

Note that the inequalities (20) – (21) and (25) or (26) are standard simple upper bounds (SUB) which are usually processed outside of the linear programming matrix (Orchard–Hays, 1968). Similarly, inequalities (22) and (23) can be considered as the so–called variable upper bounds (VUB) and processed outside of the matrix due to a special technique. Basic rules of the technique for SUB & VUB processing are developed in Section 3.4.

The main group of equality constraints (14) represents typical network relations. Similarly, the equalities (15) and (16) include only variables with unit coefficients. All the rows (14) – (16) 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 3.5.

Thus only a small number of inequalities (17) – (19) has to be considered as typical rows of linear program. While taking advantage of this fact, the TRANSLOC solver can process transhipment problems of quite large dimensions.

3.4 Implicit representation of VUB & SUB constraints

The single–objective program (13) – (26) 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 $0 \leq x_j \leq c_j$ for some variables x_j and constants c_j, such as conditions (20) – (21), (26) with respect to variables v_p, and continuous form of (24). The second one includes the so–called variable upper bounds (VUB), i.e., inequalities of the form $x_j \leq c_j x_k$ for some variables x_j, x_k and constants c_j, such as conditions (22).

SUB constraints are usually implicitly represented in commercial simplex codes (see e.g. Orchard–Hays, 1968). Schrage (1975) proposed some technique for implicit representation of VUB constraints. The technique was further developed and led to effective implementations (see e.g. Todd, 1982).

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 3.5). 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 $m \times n$ 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 $N = \{1, 2, \ldots, n\}$ 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

$$\begin{array}{rll} \max & fx & \\ \textit{subject to} & Ax = b & \\ & x_j \le c_j x_k & \text{for all } k \in K \text{ and } j \in J(k) \\ & x_k \le c_k & \text{for all } k \in K \\ & x \ge 0 & \end{array}$$

Let s_j be a slack variable for the variable upper bound $x_j \le c_j x_k$, so that

$$x_j + s_j = c_j x_k, \qquad x_j \ge 0, \qquad s_j \ge 0$$

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 $m \times m$ 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

$$B_k = A_k + \sum_{j \in J(k) \cap JU} c_j A_j$$

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.,

$$z_i = f_i - f_B B^{-1} A_i, \qquad i \in J \cup K$$

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

$$d_j = z_j \qquad \text{for } j \in J$$

$$d_k = z_k + \sum_{j \in J(k) \cap JU} c_j z_j, \quad \text{for} \quad k \in K$$

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 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 fulfils:

$$\begin{array}{lll} d_j < 0 & \text{and} & j \in JL, \\ d_j > 0 & \text{and} & j \in JU, \\ d_k < 0 & \text{and} & k \in KL, \\ d_k > 0 & \text{and} & k \in KU. \end{array}$$

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 change 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.

Let x_s $(s \in J \quad \text{or} \quad s \in K)$ be a variable chosen for enter the basis. Considering 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 we get six formulae for upper bounds on the parameter Θ and six corresponding formulae for determination of the outgoing variable (for details see Ogryczak et al., 1987). Crossing these formulae with four types of incoming variables we get 19 types (5 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).

3.5 The simplex SON algorithm

The simplex special ordered network (SON) procedure was developed by Glover & Klingman (1981, 1985). It is a partitioning method for solving LP problems with embedded network structure. Every problem of this type is characterized by a full row rank matrix A partitioned as follows:

$$A = \begin{pmatrix} ANN & ANL \\ ALN & ALL \end{pmatrix}$$

where $ANN(m \times n)$ denotes the matrix corresponding to a pure network problem and the other submatrices $ANL(m \times p)$, $ALN(q \times n)$, $ALL(q \times p)$ consist of any real elements.

The matrix A of the auxiliary LP problem discussed in Section 3.2 has obviously this form. The matrix ANN is an incidence matrix corresponding to the transportation

network studied in Section 3.1. Therefore each constraint represented by a row of ANN corresponds to a node of the network and will be referred to as node constraint. Moreover, each variable represented by a column of ANN corresponds to an arc of the network and will be referred to as arc variable. There are two classes of the ANN columns: columns containing exactly two non-zero entries in ANN (one $+1$ and one -1) called ordinary arcs and columns containing exactly one non-zero entry in ANN ($+1$ or -1) called slack arcs. 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. If a column has exactly one nonzero element pointing one of the arc endpoints then an artificial node outside the network can be meant as the second arc endpoint.

The SUB and VUB simplex algorithms use a basis B which is composed of $m + q$ linearly independent columns selected from the matrix A. Any basis B may be partitioned as follows:

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

where B_{11} is a nonsingular submatrix of ANN. It appears to be better for the effectiveness of the algorithm if rank of B_{11} is as large as possible.

Let $x_B = (x_{B_1}, x_{B_2})$ denote the basic part of the decision variable vector x, where x_{B_1}, x_{B_2} correspond to the B_{11} and B_{12} submatrices, respectively. Thus the basic variables x_{B_1} are exclusively arc variables. The basic variables x_{B_2} may also contain arc variables. Similarly, the rows of B_{11} are exclusively node rows but the matrix (B_{21}, B_{22}) may also contain node rows.

The basis inverse B^{-1} may be written as follows

$$B^{-1} = \begin{pmatrix} B_{11}^{-1} + B_{11}^{-1}B_{12}V^{-1}B_{21}B_{11}^{-1} & -B_{11}^{-1}B_{12}V^{-1} \\ -V^{-1}B_{21}B_{11}^{-1} & V^{-1} \end{pmatrix}$$

where $V = B_{22} - B_{21}B_{11}^{-1}B_{12}$.

Define the so called master basis tree (MBT) associated with a given basis. The set of nodes of the tree contains all the nodes of our LP/embedded network problem plus an external node called the master root. Thus MBT always contains $m + 1$ nodes

$$N = \{0, 1, \ldots, m\}$$

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). Each ER is connected to the master root by an externalized arc (EA).

All of the ordinary arcs in B_{11} belong to MBT. There may be two types of slack arcs associated with B_{11}. If a slack arc in B_{11} is a slack arc of ANN then the arc is replaced by an arc between the master root and its unique node. If a slack arc in B_{11} is an ordinary arc in ANN, it is replaced by an arc between its nodes in ANN (one of these endpoints is an ER node).

The arcs in the master basis tree have a natural orientation defined as follows: if an 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.

The master basis tree is represented by the following node functions.

1. PRED
   The values of the function are defined as follows:
   $PRED[i]$ = the 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, i.e., 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. LAST
   The value $LAST[i]$ specifies the node in the subtree $T(i)$ that is the last node of this subtree in THREAD order.

6. CONF
   Each node i in MBT represents the predecessor arc of the node. If the arc is conformable then $CONF[i] = +1$, otherwise $CONF[i] = -1$.

Let $P = \begin{pmatrix} P_1 \\ P_2 \end{pmatrix}$ 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 $\alpha = \begin{pmatrix} \alpha_{B1} \\ \alpha_{B2} \end{pmatrix}$ 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

$$\begin{aligned} \alpha_{B2} &= V^{-1}(-B_{21}B_{11}^{-1}P_1 + P_2) \\ \alpha_{B1} &= B_{11}^{-1}(P1 - B_{12}\alpha_{B2}) \end{aligned}$$

Suppose that the matrix $D = V^{-1}$ (i.e., the right down corner part of the matrix B^{-1}) is attained in the explicit form. Thus, the multiplication by the matrix in the former formula may be simply performed. Both the formulae include a multiplication $x = B_{11}^{-1}G$ with some vector G. This multiplication is equivalent to solving an upper triangular system $\tilde{B}_{11}\tilde{x} = \tilde{G}$, where the matrix $\tilde{B}_{11}$ consists of the rows and columns of B_{11} ordered according to the corresponding nodes and arcs on the THREAD line of MBT.

Each column of $\tilde{B}_{11}$ has at most two nonzero elements. One of them is located at the diagonal and corresponds to a node v while the second one (if exist) is located above the diagonal and corresponds to the predecessor of v. Hence, if the THREAD line is passed backward and the node v is came across then the value of the variable represented by v is computed and simultaneously the value of the variable represented by the predecessor of v is modified. Thus a single pass through the master basis tree along the RETHREAD line is sufficient for computing the x solution. The cost of such a procedure is proportional to the number of nodes in MBT.

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:

$$\begin{aligned} w_2 &= (c_{B2} - c_{B1} B_{11}^{-1} B_{12}) V^{-1} \\ w_1 &= (c_{B1} - w_2 B_{21}) B_{11}^{-1} \end{aligned}$$

The multiplication by the matrix V^{-1} may be directly computed since the matrix is assumed to be kept in the explicit form. Further, both the last formulae include multiplications of the form $w = H B_{11}^{-1}$ which can be effectively executed using the master basis tree structure for B_{11}, similarly as while computing the primal solution x.

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.

At each iteration the matrix D is transformed by elimination using a given pivot row. The following cases appear when the elimination is performed:

- the pivot row is within the rows of D;
- the pivot row is outside of D;
- a row and a column of D are dropped;
- a new row and a new column are added to D;
- a column (row) of D is replaced by another column (row) from outside of D.

Combining the elimination cases with the transition rules for the incoming, outgoing and transfer variables we get seven types of basis exchange steps. When x_{B1} is maximal relative to x_{B2}, exactly one of the seven types of basis exchange steps will occur and their updating prescriptions will maintain x_{B1} maximal.

The main features of the discussed approach are cheap 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.

4 A tutorial

4.1 The tutorial problem

To illustrate the interactive procedure and the system capabilities we present in this chapter using of DINAS to analyse a small testing example. As the test problem we use an artificial part of the real-life model for the health service districts reorganization.

The problem of health service districts reorganization connected with location of new health–care centers can be formulated as follows. The region under consideration is assumed to consist of some number of geographically defined subareas or census blocks with known distribution of the population. A number of health–care centers is available in the region but their capabilities to offering health services is not sufficient. Therefore some new facilities are located. The problem depends on determination of the locations and capacities of some new centers as well as on assignment of individuals to the centers (new and old). The proposed solution should be optimal with respect to a few objective functions and simultaneously it must be accepted by the competent decision maker.

To set the stage, we consider as a region a part of city as it is shown in Fig. 1. The five major highways divide the region into 12 subareas. For each of these areas the demand on health–care services is identified in thousands of visits per year. These quantities are included in Table 1.

Area	*Demand*
Ribes	24.5
Larix	25.0
Robur	21.0
Arnika	20.5
Rumex	19.0
Pinus	20.0
Acer	16.0
Bobrek	22.0
Picea	15.0
Litwor	20.5
Betula	13.0
Erica	23.5

Table 1: Demands on health services

Within the region there are two health–care centers offering services: Pond and Hill. They can offer 100 and 90 thousands of visits per year, respectively. Thus the total supply of services amounts 190 while the total demand on services in the region is 240. Therefore some new health–care centers should be located within the region.

There are considered four potential locations for the new centers: Ice, Fiord, Bush and Oasis. The locations are divided into two subsets associated with the corresponding two subregions:

$$North = \{Ice, Fiord\},$$

$$South = \{Bush, Oasis\}.$$

The distance between two potential locations in the same subregion are relatively small whereas each of them can meet the demands on health services. Therefore the locations belonging to the same subregion are considered as exclusive alternatives, i.e., no more than one location from the subregion can be used. Moreover, different designed capacities

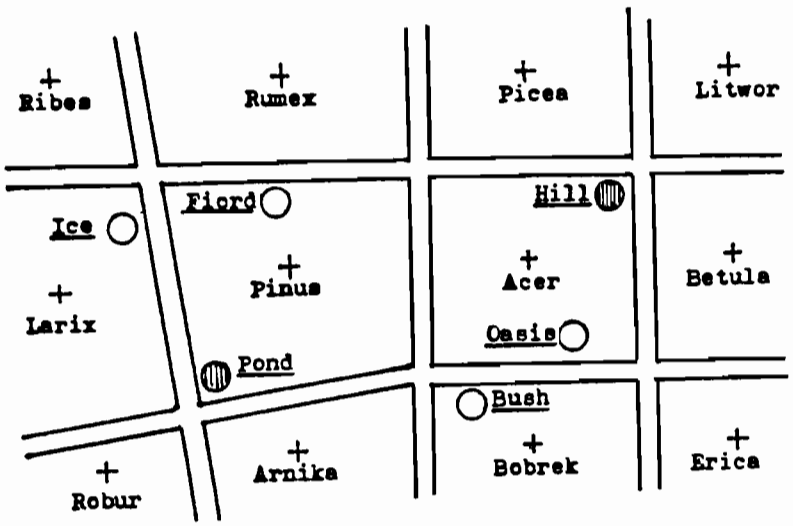


Figure 1: The region under consideration

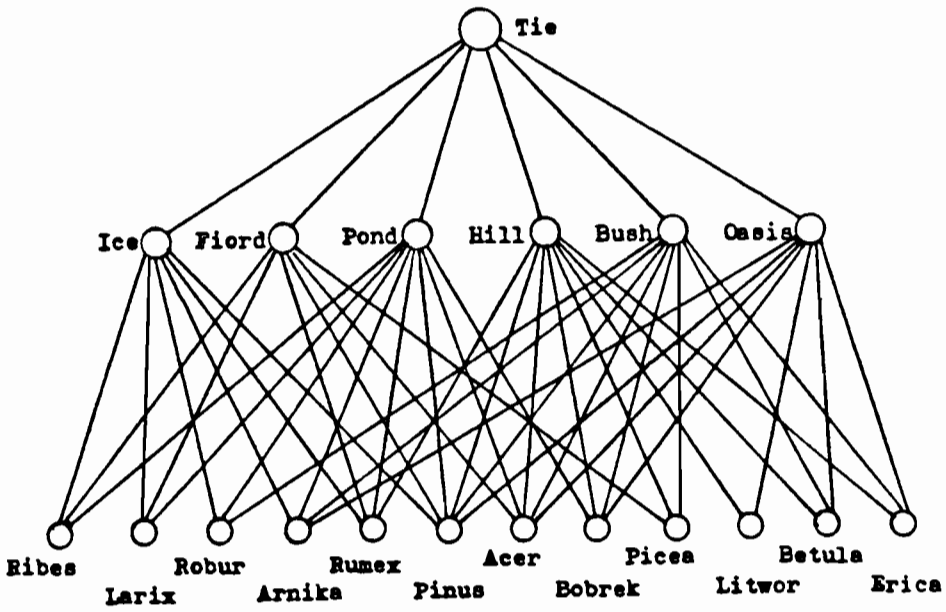


Figure 2: A scheme of the network

of health centers are associated with several locations. Table 2 lists capacities of the designed potential health–care centers.

Location	*Capacity*	*Subregion*
Ice	50	North
Fiord	60	North
Bush	50	South
Oasis	60	South

Table 2: Potential health–care centers

One must decide which potential health–care centers have to be built so as to meet the total demand on health services. The decision should be optimal with respect to the following criteria:

- minimization of the average distance per visit;
- maximization of the overall proximity to centers;
- minimization of the investment cost;
- maximization of the population satisfaction.

The first two criteria are connected with distances between health–care centers and areas assigned to them. Taking into account the urban morphology and the transportation network, it was accepted that the city–block metric was the best approximation to real distances. Therefore we define the distance between an individual and the health center as the rectangular distance between the centre of the corresponding area and the location of the health–care center. Certain connections between the areas and the health centers are eliminated as unacceptable due to too long distances or other troubles with transport. The distances between several areas and all the health centers are given in Table 3. The unacceptable connections are denoted by putting asterisks ($\star$) as a distance.

	Pond	*Hill*	*Ice*	*Fiord*	*Bush*	*Oasis*
Ribes	4.14	$\star\star\star$	2.34	3.03	$\star\star\star$	$\star\star\star$
Larix	2.61	$\star\star\star$	1.35	3.63	$\star\star\star$	$\star\star\star$
Robur	2.19	$\star\star\star$	3.96	$\star\star\star$	4.38	$\star\star\star$
Arnika	1.74	$\star\star\star$	4.08	$\star\star\star$	2.70	3.51
Rumex	4.53	4.32	2.85	1.35	$\star\star\star$	$\star\star\star$
Pinus	2.28	4.14	1.80	0.81	2.94	4.14
Acer	4.02	1.95	$\star\star\star$	3.27	1.77	1.56
Bobrek	3.84	4.17	$\star\star\star$	$\star\star\star$	1.35	1.65
Picea	$\star\star\star$	2.01	$\star\star\star$	3.30	4.71	$\star\star\star$
Litwor	$\star\star\star$	2.22	$\star\star\star$	$\star\star\star$	$\star\star\star$	4.65
Betula	$\star\star\star$	1.95	$\star\star\star$	$\star\star\star$	4.02	3.06
Erica	$\star\star\star$	3.72	$\star\star\star$	$\star\star\star$	3.42	2.46

Table 3: Distance coefficients

The overall proximity to the health care services is defined as a sum of all the individual proximity coefficients. The individual proximity is assumed to be inversely proportional to square of the distance to the health–care center. More precisely, the individual proximity coefficients are defined according to the following formula (compare Abernathy and Hershey, 1972):

$$p_{ac} = 1/(d_{ac} + \varepsilon)^2$$

where d_{ac} denotes the distance between the corresponding area a and the health–care center c, and ε is an arbitrarily small positive number. The proximity coefficients for the whole region under consideration are given in Table 4.

	Pond	*Hill*	*Ice*	*Fiord*	*Bush*	*Oasis*
Ribes	5.83	⋆⋆⋆	18.26	10.89	⋆⋆⋆	⋆⋆⋆
Larix	14.68	⋆⋆⋆	54.87	7.59	⋆⋆⋆	⋆⋆⋆
Robur	20.85	⋆⋆⋆	6.38	⋆⋆⋆	5.21	⋆⋆⋆
Arnika	33.03	⋆⋆⋆	6.01	⋆⋆⋆	13.72	8.12
Rumex	4.87	5.36	12.31	54.87	⋆⋆⋆	⋆⋆⋆
Pinus	19.24	5.83	30.86	152.42	11.57	5.83
Acer	6.19	26.30	⋆⋆⋆	9.35	31.92	41.09
Bobrek	6.78	5.75	⋆⋆⋆	⋆⋆⋆	54.87	36.73
Picea	⋆⋆⋆	24.75	⋆⋆⋆	9.18	4.51	⋆⋆⋆
Litwor	⋆⋆⋆	20.29	⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	4.62
Betula	⋆⋆⋆	26.30	⋆⋆⋆	⋆⋆⋆	6.19	10.68
Erica	⋆⋆⋆	7.23	⋆⋆⋆	⋆⋆⋆	8.55	16.52

Table 4: Proximity coefficients

The investment cost and the population satisfaction level are assumed to be a sum of fixed costs and a sum of fixed satisfaction levels connected with several possible locations, respectively. In our example the fixed coefficients take values given in Table 5.

	Ice	*Fiord*	*Bush*	*Oasis*
Investment	200	212	186	201
Satisfaction	176	87	100	192

Table 5: Investment and satisfaction coefficients

In the next section we show how the above problem can be formulated as a multiobjective transshipment problem with facility location, i.e., in the form which can be processed by the DINAS system.

4.2 The network model

The problem of health service districts reorganization connected with location of new health–care centers stated in the previous section can be easy formulated as a multiobjective transshipment problem with facility location. The areas and existing health–care centers are, certainly, fixed nodes of the network under consideration. Similarly, all the potential locations of new health centers are treated as potential nodes. Arcs represent all the possible assignments of patients to the health–care centers, i.e., arcs are associated with all the nonempty cells in Table 3 or 4. A flow along the arc from a center c to an area a expresses a number of visits in the area a serviced by the center c. In order to balance the problem in terms of supply and demand an artificial node Tie with supply equal to the overall demand is introduced. There are also defined additional arcs from the artificial node to each health–care center (existing or potential). Capacity of the existing health–care centers (Pond and Hill) are then represented as capacities of the arcs from Tie to the corresponding fixed nodes. A scheme of the network is presented in Fig. 2.

Now we can define several groups of data of the multiobjective transshipment problem with facility location. As we have mentioned in Section 1 the fixed node is characterized only by the balance, i.e., difference between the corresponding supply and demand. Table 6 lists supplies, demands and balances for all the fixed nodes in our model. Note that the sum of supplies is equal to the sum of demands and thereby the sum of balances is equal to zero.

Node	*Supply*	*Demand*	*Balance*
Ribes	0	24.5	-24.5
Larix	0	25.0	-25.0
Robur	0	21.0	-21.0
Arnika	0	20.5	-20.5
Rumex	0	19.0	-19.0
Pinus	0	20.0	-20.0
Acer	0	16.0	-16.0
Bobrek	0	22.0	-22.0
Picea	0	15.0	-15.0
Litwor	0	20.5	-20.5
Betula	0	13.0	-13.0
Erica	0	23.5	-23.5
Pond	0	0	0
Hill	0	0	0
Tie	240	0	240

Table 6: Fixed nodes

In the transshipment problem with facility location objective functions are considered as sums of linear functions of flows along several arcs and fixed costs connected with the used locations. In our model objective functions can be divided into two groups. Functions Investment (cost) and Satisfaction (level) are independent of the assignment decisions and thereby they have not coefficients connected with flows along arcs (i.e., these coefficients are equal to 0). On the other hand, functions (average) Distance and (overall) Proximity depend only on assignment decisions and they have not contain fixed terms connected with locational decisions. Fixed coefficients of the functions Investment and Satisfaction can be directly taken from Table 5. Similarly, the linear coefficients of the function Proximity are given in Table 4. The linear coefficients of the function Distance are defined as quotients of the corresponding distances by the sum of demands, i.e., as $d_{ac}/240$.

There are four potential nodes which represent the potential locations of the health-care centers, i.e., Ice, Fiord, Bush, Oasis. The data connected with the potential nodes are listed in Table 7. Here and thereafter the objective functions are denoted by abbreviations of the corresponding names.

As we have already mentioned the locations belonging to the same subregion are considered as exclusive alternatives, i.e., no more than one location from the subregion can be used. Therefore we introduce into the network model selections which represent such a type of requirements. In our model there are two selections associated with to subregions: North and South. Both the selections have the lower numbers equal to 0 and the upper numbers equal to 1. It guarantees that at most one potential node in each selection is active. The complete data connected with selections are given in Table 8.

The last group of data is connected with the arcs. The arcs are characterized by their capacities and objective functions coefficients. The cost coefficients have been already

Node	Capacity	Fixed costs			
		Invest	Satisf	Dist	Prox
Ice	50	200	176	0	0
Fiord	60	212	87	0	0
Bush	50	186	100	0	0
Oasis	60	201	192	0	0

Table 7: Potential nodes

Selection	Alternative nodes	Lower number	Upper number
North	Ice, Fiord	0	1
South	Bush, Oasis	0	1

Table 8: Selections

discussed while consideration of the objective functions. Capacities of the arcs from the artificial node Tie to the nodes representing health–care centers (Pond, Hill, Ice, Fiord, Bush, Oasis) express capacities of the corresponding centers. The arcs connecting the nodes representing health–care centers with the nodes representing the areas have essentially unlimited capacities. However, in practice, flows along these arcs are also bounded by capacities of the corresponding health–care centers and we use them as arcs capacities. All the data connected with arcs are listed in Table 9.

4.3 Getting started

To perform interactive analysis of the multiobjective problem with DINAS you need not to be familiar with neither computer techniques nor mathematical programming. DINAS is a menu–driven system with very simply commands. It is also equipped with the HELP file.

DINAS is started simply by typing command DINAS. Make sure that the drive containing the DINAS diskette is the default drive or the corresponding directory on the hard disk is the default directory. Type DINAS and press <ENTER>. The system cover is then visible on the display and there is a short pause while DOS finds and loads the execution program. When the loading procedure has already finished the blinking message:

Press any key to continue!

appears at the bottom of the display. Pressing <ENTER> or any other key, you will see the Main Menu screen with the system banner (see Fig. 3). From this moment the HELP file is still available. Press the <F1> function key to recognize it. You will get then full screen information about the Main Menu. Using the <PGDN> or <PGUP> keys you can select several screens which provide you with short characteristics of the menu commands. The same information is listed in Appendix B of this manual.

Operations available in the Main Menu are partitioned into three groups and corresponding three branches of the menu: PROCESS, SOLUTION and ANALYSIS. Command selection is performed with using a reverse image pointer which is controlled by the arrow keys. Namely, using the <LEFT> or <RIGHT> arrow you select a proper menu branch, and using the <UP> or <DOWN> arrow you select a proper item within the branch. Try to go across the menu. Note that only the selected branch is visible in an expanded form with

From	*To*	*Capacity*	*Invest*	*Satisf*	*Dist*	*Prox*
Tie	Pond	100	0	0	0	0.
Tie	Hill	90	0	0	0	0.
Tie	Ice	50	0	0	0	0.
Tie	Fiord	60	0	0	0	0.
Tie	Bush	50	0	0	0	0.
Tie	Oasis	60	0	0	0	0.
Pond	Ribes	100	0	0	.01725	5.83
Pond	Larix	100	0	0	.010875	14.68
Pond	Robur	100	0	0	.009125	20.85
Pond	Arnika	100	0	0	.00725	33.03
Pond	Rumex	100	0	0	.018875	4.87
Pond	Pinus	100	0	0	.0095	19.24
Pond	Acer	100	0	0	.01675	6.19
Pond	Bobrek	100	0	0	.016	6.78
Hill	Rumex	90	0	0	.018	5.36
Hill	Pinus	90	0	0	.01725	5.83
Hill	Acer	90	0	0	.008125	26.30
Hill	Bobrek	90	0	0	.017375	5.75
Hill	Picea	90	0	0	.008375	24.75
Hill	Litwor	90	0	0	.00925	20.29
Hill	Betula	90	0	0	.008125	26.30
Hill	Erica	90	0	0	.0155	7.23
Ice	Ribes	50	0	0	.00975	18.26
Ice	Larix	50	0	0	.005625	54.87
Ice	Robur	50	0	0	.0165	6.38
Ice	Arnika	50	0	0	.017	6.01
Ice	Rumex	50	0	0	.011875	12.31
Ice	Pinus	50	0	0	.0075	30.86
Fiord	Ribes	60	0	0	.012625	10.89
Fiord	Larix	60	0	0	.015125	7.59
Fiord	Rumex	60	0	0	.005625	54.87
Fiord	Pinus	60	0	0	.003375	152.42
Fiord	Acer	60	0	0	.013625	9.35
Fiord	Picea	60	0	0	.013750	9.18
Bush	Robur	50	0	0	.01825	5.21
Bush	Arnika	50	0	0	.01125	13.72
Bush	Pinus	50	0	0	.01225	11.57
Bush	Acer	50	0	0	.007375	31.92
Bush	Bobrek	50	0	0	.005625	54.87
Bush	Picea	50	0	0	.019625	4.51
Bush	Betula	50	0	0	.01675	6.19
Bush	Erica	50	0	0	.01425	8.55
Oasis	Arnika	60	0	0	.014625	8.12
Oasis	Pinus	60	0	0	.01725	5.83
Oasis	Acer	60	0	0	.0065	41.09
Oasis	Bobrek	60	0	0	.006875	36.73
Oasis	Litwor	60	0	0	.019375	4.62
Oasis	Betula	60	0	0	.01275	10.68
Oasis	Erica	60	0	0	.01025	16.52

Table 9: Arcs

the commands list and the pointer position within a list is automatically restored during the next selection of the same branch. For execution of the selected command you have to press <ENTER>. After a command has been executed you need, usually, to press the <ESC> key in order to return control to the Main Menu.

At the beginning the PROCESS branch is initialized. The PROCESS branch contains basic operations connected with processing of the multiobjective problem and generation of several efficient solutions. There are included problem definition operations such as calling the Network Editor for input or modification of the problem (PROBLEM) and converting of the edited problem with error checking (CONVERT). Further, in this branch the basic optimization operations are available: the computation of the pay–off matrix with the utopia and nadir vectors (PAY-OFF) and the generation of efficient solutions depending on the edited aspiration and reservation levels (EFFICIENT). As the last command in this branch is placed the QUIT operation which allows you to finish work with the system.

The SOLUTION and ANALYSIS branches collect additional operations on the efficient solutions. So, at the beginning their commands are not available. If you try to execute such a command you will get the error message:

```
There are no solutions !
SELECT ANOTHER COMMAND !
```

You have then to press the <ESC> key to return control to the Main Menu and select a proper command.

Before the multiobjective analysis is launched, a file must be constructed describing the problem to be analysed. This file is called the network file. It is a special binary file generated by the Network Editor. The Network Editor is called by execution of the PROBLEM command. Thus the PROBLEM command is usually selected as the initial step.

In the next section we describe in details the process of the network file preparation for the tutorial problem. However, if you want to learn only the Interactive System without uphill data input you can skip this stage and use our network file. On the distribution diskette there is available a network file DEMO which contains the data for the tutorial problem.

The network file created and saved during the execution of the PROBLEM command is not automatically assigned to the Interactive System. You may edit many various network files during a single editor calling. To assign a specific network file to the system you have to execute the CONVERT command. The execution of the CONVERT command makes the specified problem (network file) available for the system. Thus in both the cases: after the preparation of a network file by yourself as well as while using the DEMO file you have to execute the CONVERT command.

After selecting the CONVERT command you will be asked for a specification of the problem name. For this purpose a special 38–character field is displayed preceded by the message:

```
GIVE Problem Name:
```

You can type then DEMO or the name of your network file, respectively. If you have made a mistake while typing you can correct it by using the <LEFT> or <RIGHT> arrows for cursor movement and the <DEL> key to cancel improper characters. The general field editing rules are summarized in Appendix A.

Next, the specified problem data will be converted with error checking. Any recognized data error will be reported on the display. In case of using the DEMO file no errors should be reported. After a successful conversion the problem statistics will be presented (see

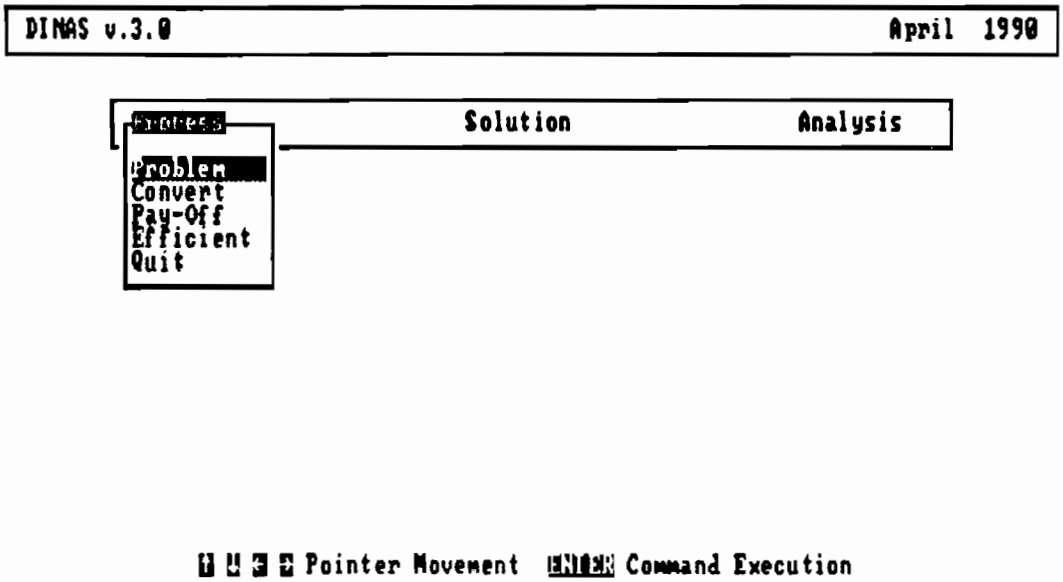


Figure 3: Main Menu screen

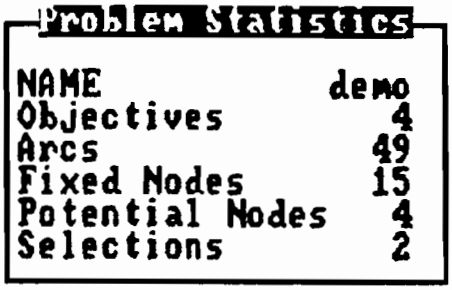


Figure 4: Problem statistics

Fig. 4). You may read there that the problem is built of 4 objectives, 49 arcs, 15 fixed nodes and 4 potential nodes organized into 2 selections. In order to return control to the Main Menu you have to press the <ESC> key.

4.4 The network file preparation

Let you try to prepare the network file for the tutorial problem. The file has to contain all the information defining the problem, such as:

- the names, types and activities for the objectives;
- the names and balances for the fixed nodes;
- the names, capacities, selections and objective coefficients for the potential nodes;
- the bounds for the selections;
- the names (optionally), capacities and objective coefficients for the arcs.

The data need not to be edited sequentially after the fashion of the given data tables (see Tables 6 to 9) but rather following the scheme of the network by visiting nodes and arcs according to the network structure.

To create the network file you have to select the PROBLEM command from the PROCESS branch of the DINAS Main Menu. Then the Network Editor (called EDINET) is loaded and the Editor Menu screen with the editor banner appears after a little while (see Figure 5). From this moment the editor HELP file is still available by pressing the <F1> key. Press the <F1> key to recognize the HELP file.

The editor commands are partitioned into three branches: FILE, PRINT NETWORK and EDIT NETWORK. The FILE branch collects commands associated with file operations and leaving the editor. The PRINT NETWORK branch consists of a single command which generates various reports. The EDIT NETWORK branch collects the editor commands which enable a problem data input or modification. Similarly as in the Main Menu a command is selected using a reverse image pointer which is controlled by the arrow keys.

The head line of the menu contains the names of the branches. The first branch FILE is active so its three commands are listed below. The cursor points out the LOAD command. Press <ENTER> to execute the LOAD command. Then the file name line appears on the display:

```
File:
```

and you are asked to define a file name. Type any legal new file name, e.g., Health, and press <ENTER>. Then the message:

```
Some problems with data file - press ENTER
```

will be issued since the file Health does not exist as yet. Press <ENTER> again. The new network file Health.net has been just opened and the Editor Menu is again activated.

Select the EDIT NETWORK branch from the menu using the <RIGHT> arrow. Then the display with the listed EDIT NETWORK commands is visible (see Figure 6).

It appears to be most effective if you execute at first the OBJECTIVES command. Having defined the objectives you will be able to edit the objective coefficients while visiting several nodes and arcs since the objective names will be automatically introduced into the corresponding windows.

Thus execute the OBJECTIVES command. Then the OBJECTIVES window is displayed. The cursor points out the field of the first objective name. Type the name of the objective

Invest. Next, you should define the TYPE of the objective (MIN for minimization or MAX for maximization) and its ACTIVITY (YES if active, NO otherwise). Since there are highlighted the proper options in the fields TYPE and ACTIVITY (MIN and YES, respectively) press the <DOWN> arrow to finish inserting this objective.

Now, the cursor points out the second objective name. Type the name Satisf. Next, you must change the highlighted option in the TYPE field since the objective have to be maximized. To select the MAX option press <ENTER> and use the <RIGHT> arrow. At the end press the <DOWN> arrow to point out the name field of the next objective. The other objectives may be defined in a similar way. The complete objective window is presented in Figure 7.

After the definition of all the objectives is finished press <ESC> to return control to the Editor Menu.

In order to edit nodes and arcs of the network select the LIST NODES command and press <ESC>. The execution of this command causes that the LIST NODES window is displayed. The window is blank since any node has not been defined as yet. To define a node , e.g., Bush, enter the first character of the node name (the capital B). Then the CURRENT NODE name window appears (see Figure 8). The character B is displayed inside the name field as the first character of the name. Finish typing of the name and press <ENTER>. Then the window presented in Figure 9 appears. The cursor points out the default "fixed" option. Use the <RIGHT> arrow to select the "potential" option since the node Bush is a potential one, and press <ENTER>. The potential node window with the node Bush as CURRENT NODE is then displayed (see Figure 10a).

The cursor points out the capacity field inside the window. Edit the capacity of Bush: 50, and press the <DOWN> arrow. Edit the name South of the unique selection containing the node and press the <DOWN> arrow. Now, the block of objectives is displayed inside the window. The cursor points out the coefficient field of the objective Invest. Insert the coefficient 186 and press the <DOWN> arrow to point the next objective field. Insert the coefficient 100 corresponding to the objective Satisf. The editor treats the blank coefficient fields as zeroes. Thus you need not to edit the coefficients corresponding to the Dist and Prox objectives since both of them are equal to zero. The complete potential node window is presented in Figure 10b.

Press <ESC> to finish editing of the data characterizing the node Bush. Then two additional windows are displayed on the screen (see Fig. 11): the NODE FROM list window (on the left hand side) and the NODE TO list window (on the right hand side). The windows allow you to define the predecessors and successors to the node Bush. Press <CTRL>/<LEFT> arrow to activate the NODE FROM window. Type the name Tie of the single predecessor to the node Bush. After the first character T of the name is inserted, the NODE FROM name window appears. The character T is displayed inside the name field. Finish inserting of the name Tie and press <ENTER> to select the default option "fixed" since the node Tie is fixed. The fixed node window with the node Tie as NODE FROM is then displayed. The cursor points out the balance field inside the window. Edit the balance of Tie as 240, and press <ESC>.

At the down–left corner of the window the brief question:

ARC?

appears (see Figure 12). You are asked this way whether you wish to edit data characterizing the arc Tie–Bush.

Press <Y> to display the predecessor arc window. The cursor points out the arc name field. Edit an arc name, e.g., Tiebush, if you wish it. Note that the arc name need not to

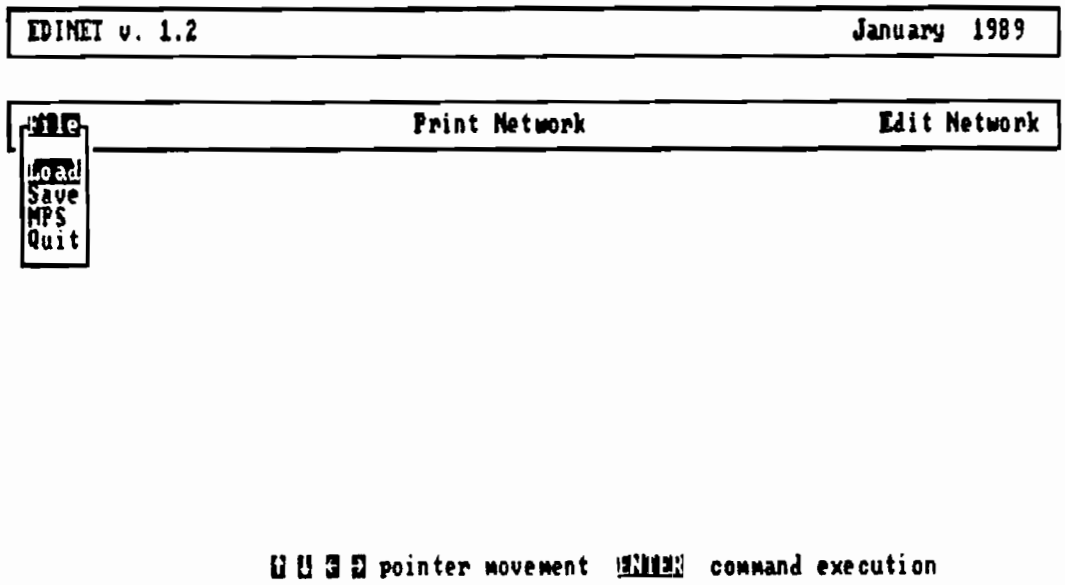


Figure 5: Editor Menu with the FILE branch specified

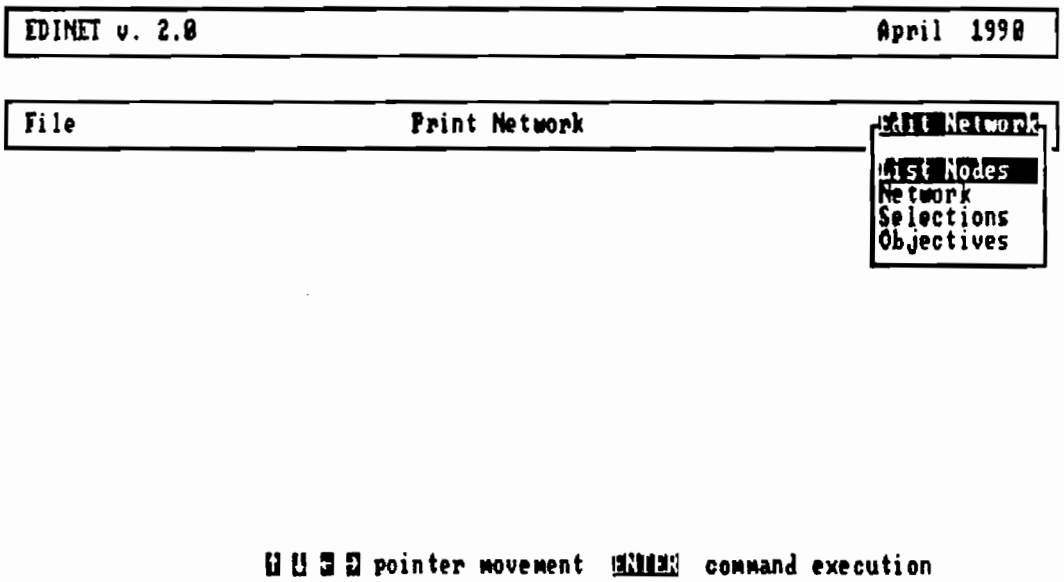


Figure 6: Editor Menu with the EDIT NETWORK branch specified

be inserted due to its optional character (press the <DOWN> arrow to avoid the arc name editing). After pressing <ENTER> (or <DOWN> arrow) the capacity field of the arc is pointed out. Edit capacity as 50, and press <ENTER>. The cursor moves to the objectives block and points out the Invest coefficient field. All the objective coefficients corresponding to the arc Tie–Bush have the zero value so you may to let their fields be blank. Thus the editing of this arc is finished. Press <ESC>.

After pressing <ESC> the NODE FROM list window appears. Activate the CURRENT NODE window using again the <ESC> key. In order to edit the data of successors to the current node Bush press <CTRL>/<RIGHT> arrow. The NODE TO list window (empty as yet) is then activated. Type the name of a successor to the node Bush, e.g., Robur. After inserting of the first character R the NODE TO name window is displayed. The character R is found inside the name field. Finish editing of the name and press <ENTER>. The cursor points out the "fixed" option. Press <ENTER> to confirm this option since Robur is a fixed node. The fixed node window with the node Robur as NODE TO is then displayed. The cursor points out the balance field inside the window. Edit the balance of Robur as -21, and press <ENTER>.

At the down–left corner of the window the question:

ARC?

is issued. Press <Y> to edit data of the arc Bush–Robur. Then you can see the ARC window corresponding to the arc Bush–Robur. The cursor points out the arc name field. Edit an arc name, e.g., Burob, the capacity of the arc (50) and the associated objective coefficients (Invest=0, Satisf=0, Dist=0.01825, Prox=5.21). This completed arc window is presented in Figure 13.

Press <CTRL>/<RIGHT> arrow and <ESC> to return to the NODE TO list window. The first line of the window contains the node name Robur and the arc name Burob. Repeat the similar procedure to edit data of the other successors to the node Bush, i.e., Arnika, Pinus, Acer, Bobrek, Picea, Betula, Erica. The NODE TO (likewise NODE FROM) list window may contain at most seven node names and associated arc names. Therefore after editing the eighth successor Erica you have to scroll the list with <DOWN/UP> arrow in order to look over all the successors since either the first or the eighth successor is not visible. The full NODE TO list window is presented in Figure 14.

Suppose that the data of successors to the node Bush are edited and the NODE TO list is displayed. To modify the data of a node or an arc you have at first to select the name of the corresponding successor (e.g., Robur). Press <ENTER> to display the NODE TO window and modify the data of the selected node Robur. If you want to modify the data of the arc Bush–Robur only, e.g., its name, press <CTRL>/<RIGHT> arrow. The corresponding arc window is then displayed. Change the name Burob into Buro (deleting the last character b) and press <CTRL>/<RIGHT> arrow to return to the NODE TO list window.

Use the <ALT>/<N> keys to call a network scheme of the edited part of the network. Use the <DOWN> and <UP> arrow keys to introduce a selected node onto the pointed out position (the highest central position). While changing the pointed out node, the window containing data of the node appears on the right hand side of the screen. Press <TAB> to activate the window corresponding to the pointed out node. Now, you can examine and modify data inside the window. Press <ESC> to return to the central position at the top of the network scheme.

Introduce the node Bush onto the pointed out position. Press <RIGHT> arrow to point out the node Robur. Change successors to Bush using <DOWN>, <UP> arrow keys. On the right hand side of the screen the window including data corresponding to the arc

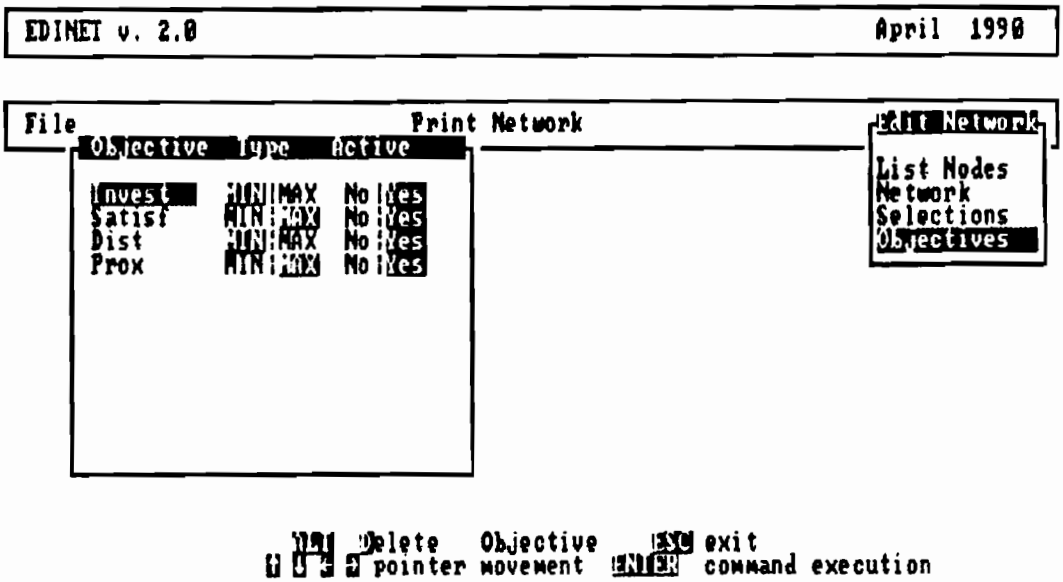


Figure 7: OBJECTIVES window

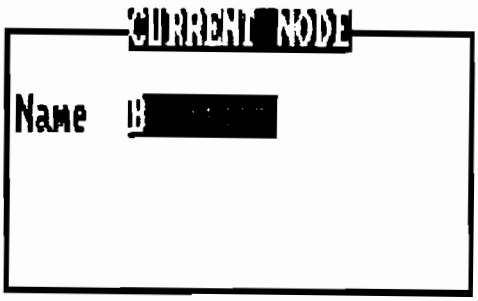


Figure 8: NODE name window

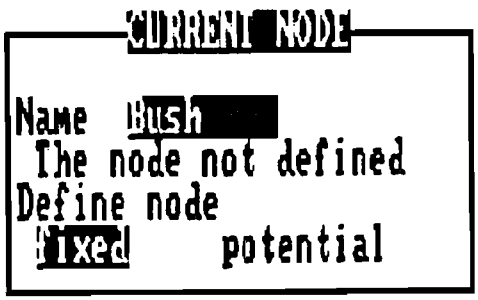


Figure 9: NODE type window

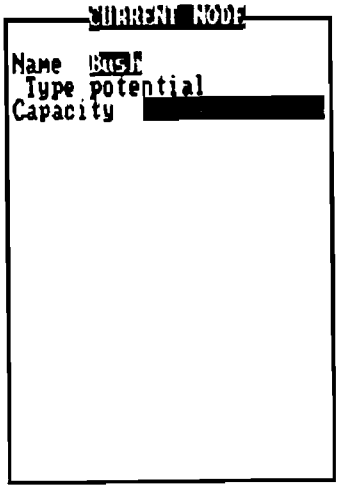


Figure 10a: Initial potential node window

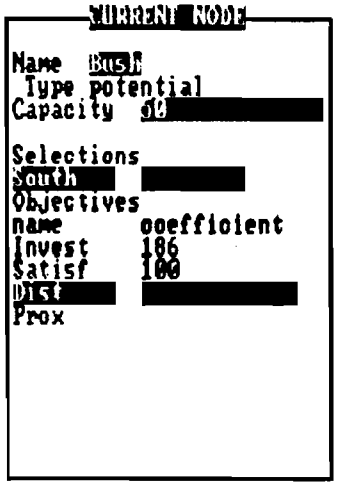


Figure 10b: Complete potential node window

Bush-Selected Node appears. The data are exchanged while selecting another successor. Select a successor to Bush and press the <TAB> key. Now, the arc window corresponding to the arc Bush-Selected Node is activated and you can examine and modify data inside the window. Press <ESC> to exit the arc window. To return to the node Bush occupying the central position at the top press the <LEFT> arrow key.

This compact part of the scheme illustrates the network part just edited (see Figure 15).

Press <LEFT> arrow to point the node Tie and then press <ENTER>. In result this node will occupy the central pointed out position. Use <ENTER> again to display the CURRENT/FROM/TO screen with the node Tie as the current one. The NODE FROM list window is blank since Tie has not any predecessor. Inside of the NODE TO list window there is only the single successor Bush.

Press <CTRL>/<RIGHT> arrow to activate the NODE TO list window. Type the name of another successor to the node Tie, say Hill, to edit data corresponding to this node and to the arc Tie–Hill. The editing may be performed similarly as it was mentioned in case of the node Robur. Repeat this procedure in order to introduce data corresponding to other successors to the node Tie (Pond, Oasis, Fiord, Ice).

After all these successors are introduced with their data press <ALT>/<L> to display the LIST NODES window (see Fig. 16). You may find the following node names inside the window: Acer, Arnika, Betula, Bobrek, Bush, Erica, Fiord, Hill, Ice, Oasis, Picea, Pinus, Pond, Robur, Tie, provided that the editing was performed correctly. Use arrow keys to select a node name from the nodes list. On the right hand side of the list the window containing data associated with the selected node appears (see Figure 16). The data inside the window are exchanged while selecting another node. Press <TAB> key to activate the node window associated with the selected node. Now, you can examine and modify data inside the window. Press <ESC> to return to the list level. Use arrow keys to select the node name Hill and press <ENTER>. This name with the associated balance appears inside the CURRENT NODE window since the node was defined before. The node Tie occupy the NODE FROM list window alone and the NODE TO list window is empty.

Activate the NODE TO list window pressing <CTRL>/<RIGHT> arrow. The successors to the node Hill are divided into two subsets: the subset of succeeding nodes which are not defined yet (Litwor, Rumex) and the subset of succeeding nodes which are already defined (Acer, Betula, Bobrek, Erica, Picea, Pinus). To define the nodes Litwor and Rumex repeat the procedure used in case of the node Robur. Defining of the other successors is easier since after editing of an existing node name, the NODE TO window with the data associated with this node immediately appears.

Type, say, the name Acer that is already defined. Pressing the first character A causes calling of the NODE TO name window. This character is found inside the name field. Finish editing of the name and press <ENTER>. Then the "fixed" NODE TO window will be automatically displayed with the node name Acer and the corresponding balance -16. After pressing <ESC> you are asked (ARC?) whether you want to define the arc Hill–Acer or not. Define the arc similarly as it was performed in case of the arc Bush–Robur. Define in such a way the other successors (Betula, Bobrek, Erica, Picea, Pinus) and return to the NODE TO list window.

Select the node name Pinus and use <ALT>/<C> keys. The CURRENT NODE window will be then activated with the current node Pinus. Activate the NODE FROM window by pressing <CTRL>/<LEFT> arrow. The window contains two node names: Bush and Hill. Define the other predecessors to the node Pinus: Ice, Fiord, Pond and Oasis, with their arcs.

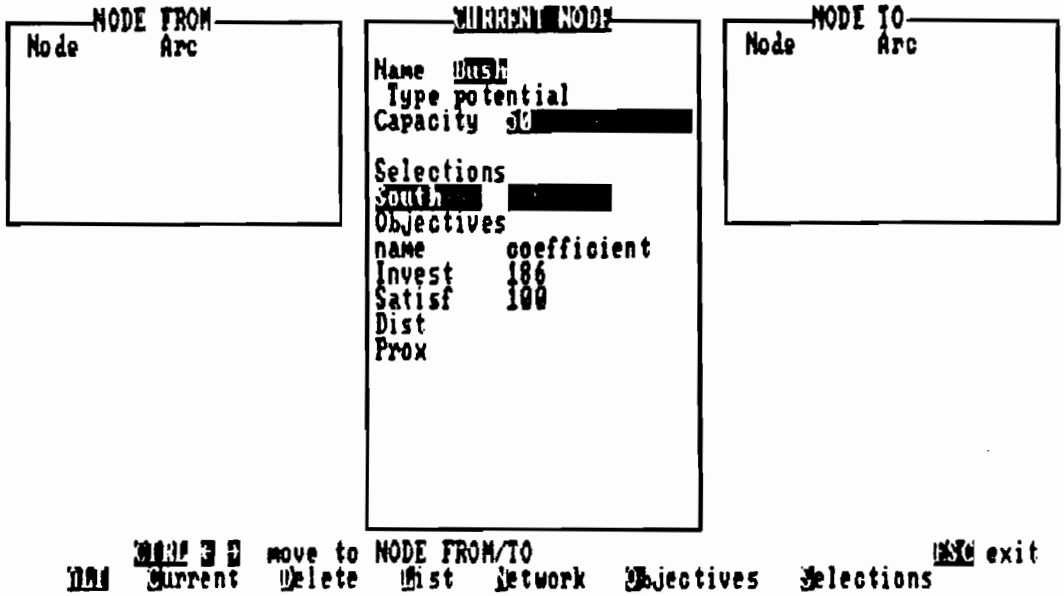


Figure 11: CURRENT NODE and NODE FROM/TO list windows

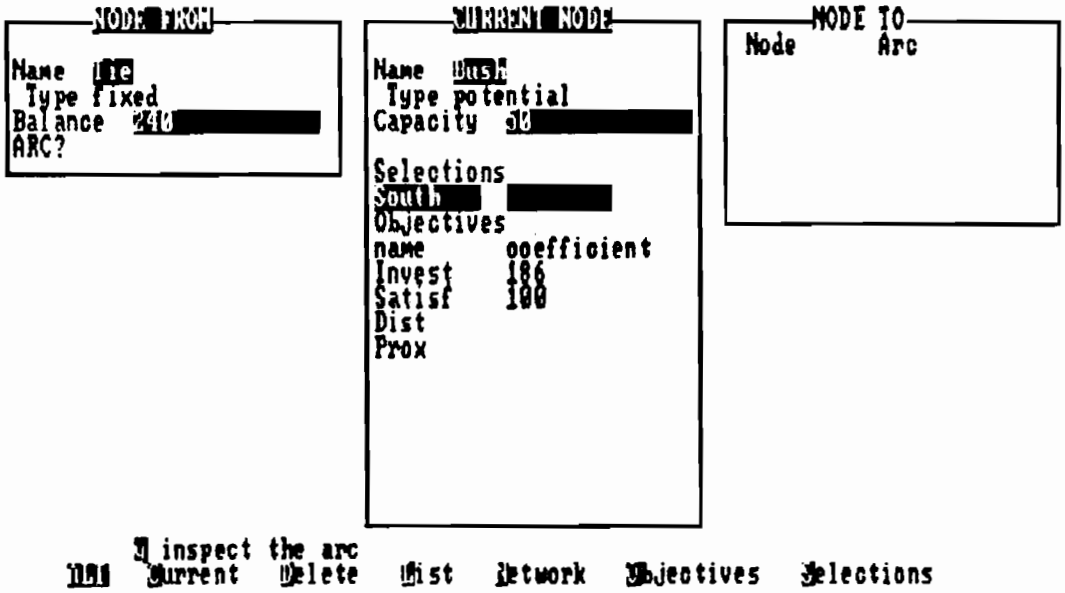


Figure 12: Fixed node window with the node Tie

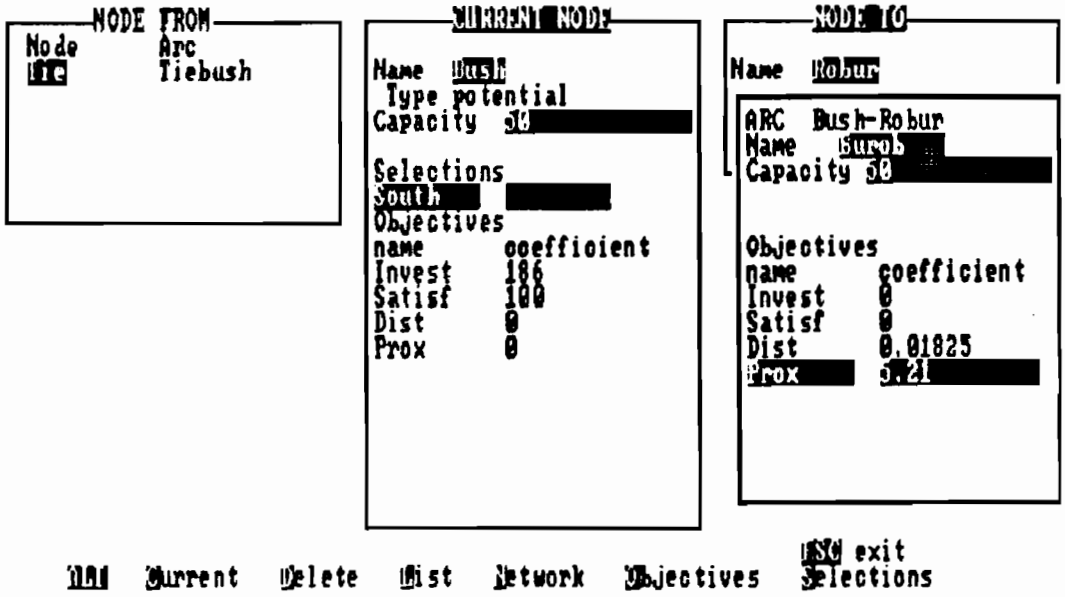


Figure 13: Arc window

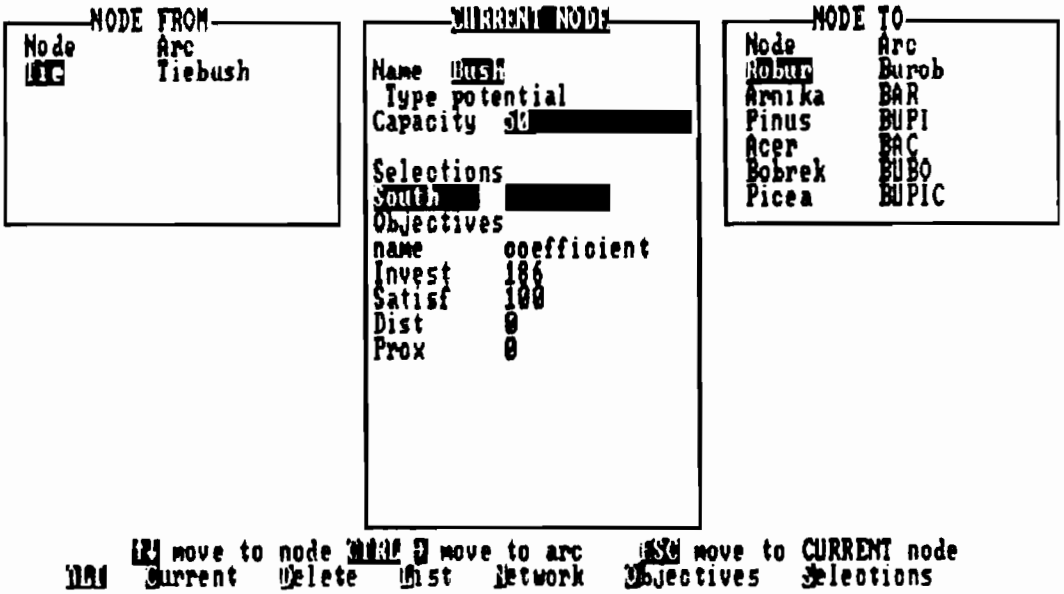


Figure 14: The full NODE TO list window

Similarly, select the node name Ice inside the NODE FROM window and press <ALT>/<C>. The node Ice becomes now the current node and the CURRENT NODE window is activated. Use <CTRL>/<RIGHT> arrow to activate the NODE TO window. Define successors to the node Ice.

Continue defining of the network and verify sometimes (using <ALT>/<L> and <ALT>/<N> commands) whether the introduced data are completed.

After editing of the network is finished press <ALT>/<S> to execute the SELECTIONS command. Then the SELECTIONS window will be issued (see Figure 17). The selections list inside the window contains two selections: North with the nodes Fiord and Ice, and South with the nodes Bush and Oasis. The North selection is pointed out. Press <ENTER> to display the BOUNDS window corresponding to the pointed selection. Edit the lower bound as 0 and the upper bound as 1 and press <ESC>. The selection list appears again. Use the <DOWN> arrow to introduce the South selection into the pointed out position and press <ENTER>. The BOUNDS window for editing of the South selection bounds is then displayed (see Figure 18). Edit the lower bound as 0 and the upper bound as 1 and press <ESC> to leave the BOUNDS window. At the end press <ESC> to leave the SELECTIONS window.

The input data of the network are completed. Now you have to save the edited data on a disk. For this purpose you have to press <ESC> a few times so as to return to the Editor Menu and execute the SAVE command. Then this message:

```
File: Health.net
```

will be issued. To save this network file in the default directory press <ENTER>.

You can also generate a MPS file corresponding to the edited data and save it on a disk. To do it you have to execute the MPS command. Then the message:

```
File: Health.mps
```

appears. Press <ENTER> to generate and save the MPS file. The file will be saved in the default directory.

The editing of the network file Health.net is finished. In order to leave the editor select the QUIT command and press <ENTER>.

4.5 Introductory multiobjective analysis

In this and subsequent sections we present an outline of the basic multiobjective analysis performed on our test problem. We do not discuss all the capabilities of the system which can be used in such an analysis.

Having defined and converted the problem as the first step of the multiobjective analysis you must perform the PAY-OFF command. It executes the optimization of each objective function separately. While performing of the optimization the computation process is reported by messages like this:

```
WAIT   Iterations: 57   Solutions:  0
```

where the numbers represent the number of simplex steps having performed and the number of feasible solutions having examined, respectively. You can suspend the computations at any moment (by pressing <ENTER> or any other key) and cancel them or continue after a break. Try it.

Finally, you get the so–called pay–off matrix presented in Fig 19. The pay–off matrix is a well–known device in multiobjective programming. It gives values of all the objective

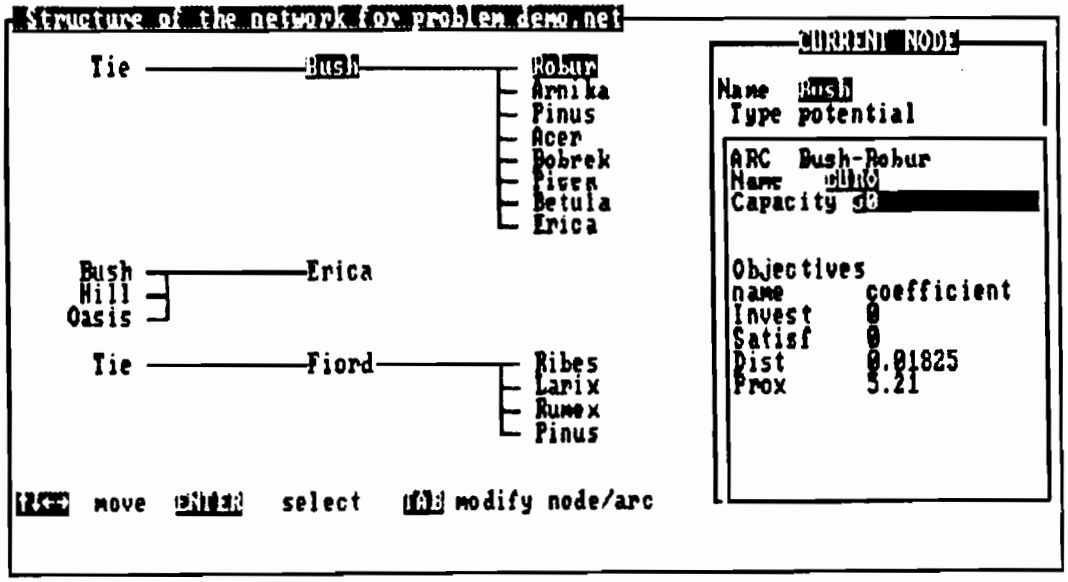


Figure 15: A part of the network scheme with the pointed out arc window

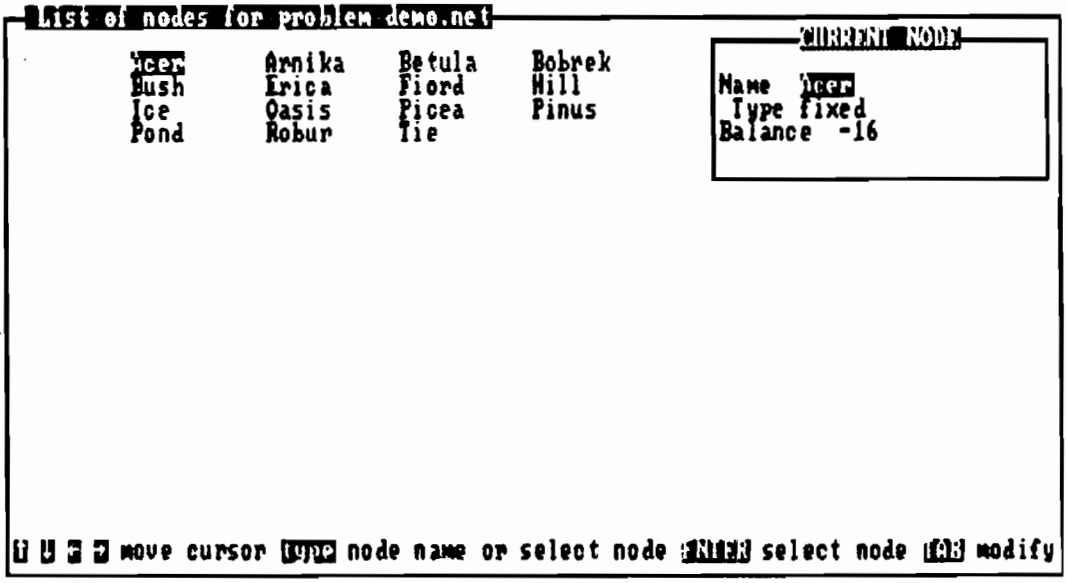


Figure 16: LIST NODES window with the pointed out node window

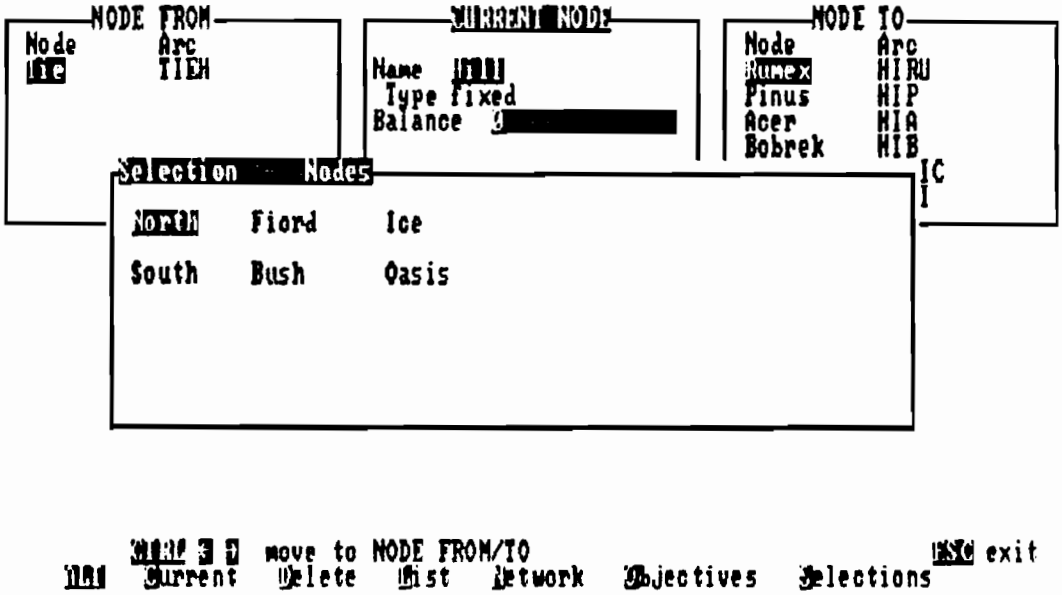


Figure 17: SELECTIONS window

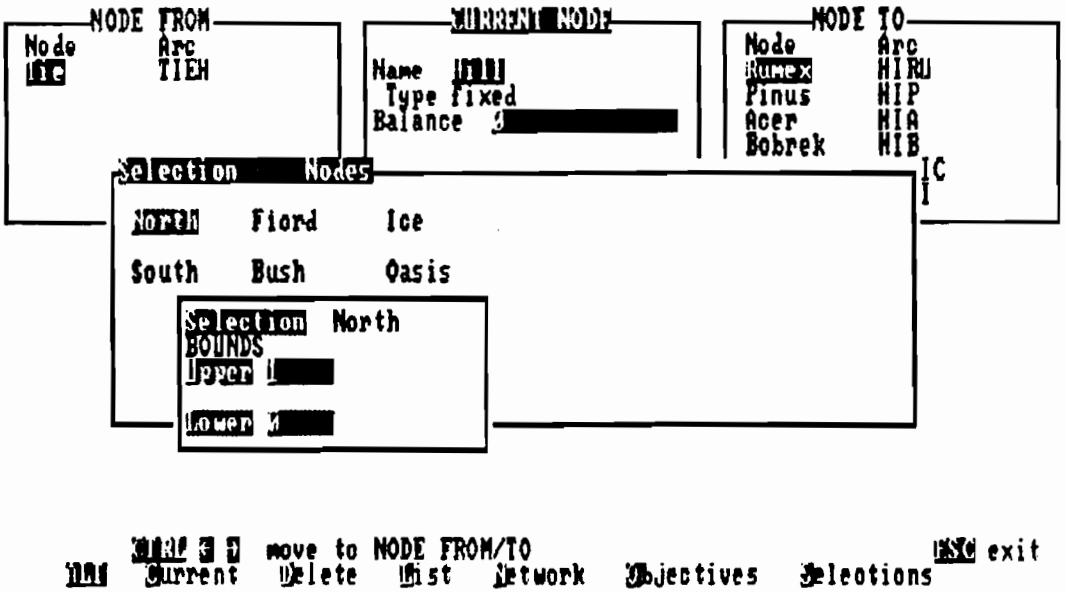


Figure 18: BOUNDS window

functions (columns) obtained while solving several single–objective problems (rows) and thereby it helps to understand the conflicts between different objectives.

Execution of the PAY-OFF command provides also you with two reference vectors: the utopia vector and the nadir vector (see Fig. 19). The utopia vector represents the best values of each objective considered separately, and the nadir vector expresses the worst values of each objective noticed during optimization of another objective function. The utopia vector is, usually, not attainable, i.e., there are no feasible solutions with such objective values.

While analysing tables in Fig. 19 you may find out that the objective values vary significantly depending on the selected optimization. Only for the average distance you may notice the relative variation less than 30% whereas for the other objectives it even overstep 100%. Moreover, you may recognize a strong conflict between the investment cost and all the other objectives. While minimizing the investment cost you get the worst values for all the other objectives. On the other hand, while optimizing another objective function you get doubled investment cost in comparison with its minimal value.

Coefficients of the nadir vector cannot be considered as the worst values of the objectives over the whole efficient (Pareto–optimal) set. They usually estimate these values but they express only the worst values of each objective noticed during optimization of another objective function. In further analysis we will show that these estimations can be sometimes overstep.

Due to the special regularization technique used while computation of the pay–off matrix (see Chapter 3) each generated single–objective optimal solution is also an efficient solution to the multiobjective problem. DINAS stores the efficient solutions in a special Solution Base. All the efficient solutions generated (or input from a file) during a session get consecutive numbers and are automatically put into the Solution Base.

DINAS is armed with many operations which help to manage the Solution Base. There are two kinds of operations connected with the Solution Base: operations on a single efficient solution, and operations on the whole Solution Base. Operations addressed to a single solution are connected with the Current Solution. The newest generated efficient solution is automatically assumed to be the Current Solution but any efficient solution from the Solution Base can be manually activated as the Current Solution.

So, you have already available in the Solution Base four efficient solutions connected with several rows of the pay–off matrix. Using different commands of DINAS we can examine in details these solutions.

The SOLUTION branch contains additional operations connected with the Current Solution. You can examine in details the Current Solution using the Network Editor (BROWSE) or analyse only short characteristics such as objective values and selected locations (SUMMARY). Values of the objective functions are presented in three ways: as a standard table, as bars in the aspiration/reservation scale and as bars in the utopia/nadir scale. The bars show percentage level of each objective value with respect to the corresponding scale. You may also save the Current Solution on a separate file in order to use it during next runs of the system with the same problem (SAVE). There is also available a special command to delete the Current Solution from the Solution Base if you find it as quite useless (DELETE).

The ANALYSIS branch collects commands connected with operations on the Solution Base. The main command COMPARE allows you to perform a comparison between all the efficient solutions included in the Solution Base or in some subset of the base. In the comparison only short characteristics of the solutions are used, i.e., objective values in the form of tables and bars as well as tables of selected locations. Moreover, some

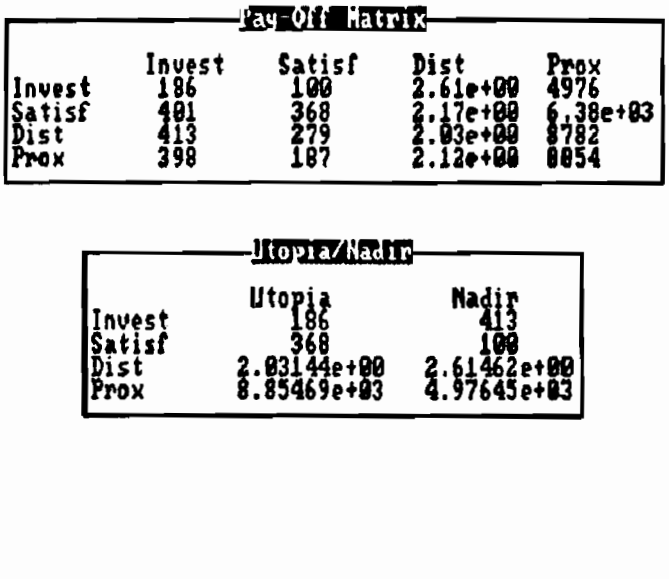


Figure 19: PAY-OFF screen

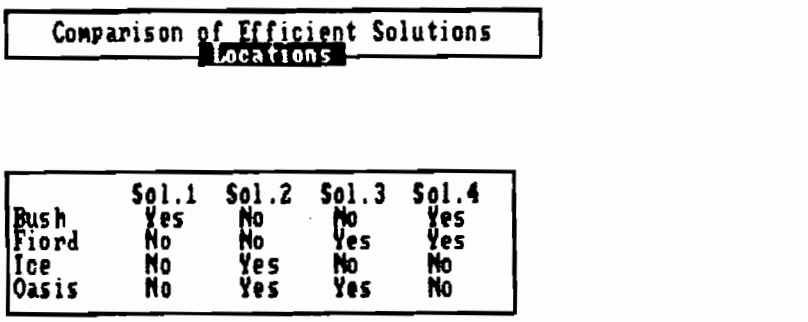


Figure 20: Locations for the pay–off solutions

commands included in this branch (PREVIOUS, NEXT and LAST) allow you to select any efficient solution from the Solution Base as the Current Solution. There exist also an opportunity to restore some efficient solution (saved earlier on a separate file) to the Solution Base (RESTORE).

In particular, you can recognize locations structure for several solutions. The simplest way to get this information depends on using of the COMPARE command from the ANALYSIS branch. Let you execute the COMPARE command. At first you are asked to select solutions included into the comparison. For this purpose a special (temporal) Comparison Base is used. The Comparison Base is initialized as containing all the efficient solutions from the Solution Base. You can remove some solutions from the Comparison Base and thereby select the remaining solutions for the comparison.

The process of the Comparison Base definition is organized as follows. At the top of the screen the message appears:

SELECT Efficient Solutions for Comparison

and simultaneously the initial Comparison Base is displayed. The Comparison Base is displayed as a table where each row corresponds to a single efficient solution while the columns correspond to several objective functions. In each row you can read the identification number of the efficient solution and objective values for several criteria. There is available a reverse–image pointer which allows you to select a solution by pointing out its identification number. The pointer is controlled by using the <UP> and <DOWN> arrows. If you have pressed the <DEL> key then the pointed out solution is removed from the Comparison Base and the corresponding row is deleted from the table on the screen. In such a way you can eliminate from the Comparison Base the solutions which you are not interested in during the comparison. You may finish the process of the Comparison Base definition at any moment by pressing the <ESC> key.

We are interested in comparison of all four efficient solutions. So, you should press the <ESC> key without using the <DEL> key. However, do not be afraid to use the <DEL> key. The Comparison Base is only temporal for a single execution of the COMPARE command. All the solutions remain still available in the Solution Base. You can next press twice the <ESC> key in order to return control to the Main Menu and repeat the COMPARE command with the full Comparison Base.

After setting up the Comparison Base by pressing the <ESC> key you can analyse simultaneously the four selected solutions in terms of the objective values and location decisions. The several COMPARE screens are controlled by the <PGDN> or <PGUP> keys which display screen by screen forward or backward, respectively. In order to get the so–called Locations screen you have to press the <PGUP> key or triple press the <PGDN> key. This screen is presented in Fig. 20. You may notice that the first solution which minimizes the investment cost is based on only one new health–care center located at Bush. Each other solution use two new centers what explains their significantly higher investment costs. They are based on the following locations: Ice and Oasis, Fiord and Oasis, Bush and Fiord, respectively. After performing the comparison you have to press the <ESC> key to return control to the Main Menu.

4.6 Interactive analysis

Having computed the utopia vector we can start the interactive search for a satisfying efficient solution. As we have already mentioned DINAS utilizes aspiration and reservation levels to control the interactive analysis. More precisely, you specify acceptable values for

several objectives as the aspiration levels and necessary values as the reservation levels. All the operations connected with editing the aspiration and reservation levels as well as with computation of a new efficient solution are included in the EFFICIENT command.

After selecting the EFFICIENT command you are asked for aspiration and reservation levels editing. The message:

```
EDIT Aspiration/Reservation Levels
```

appears at the top of the screen whereas two tables are displayed at the centre (see Fig. 21). The left table contains the objective values for the Current Solution (central column) in comparison with the utopia (left column) and nadir (right column) vectors. At this moment the fourth efficient solution is the Current Solution. The right table is a small spreadsheet for aspiration and reservation levels edition. You can select several fields and edit the corresponding coefficients. A twelve–character (reverse image) field pointer is controlled by the <ENTER> key which move the pointer entry by entry. Moreover, you can use the <UP> or <DOWN> arrows for a direct vertical movement and, similarly, the <CTRL>/<LEFT> or <CTRL>/<RIGHT> keys for a direct horizontal movement. Try to move the pointer. At the beginning of the interactive analysis the aspirations and reservation levels are initialized as equal to the utopia and nadir vectors, respectively.

You can finish the edition phase at any moment by pressing the <ESC> key. The system will check then the introduced coefficients. To be accepted by the system the edited coefficients have to satisfy the following relations:

- for each minimized objective (i.e., for Invest and Dist):
  - the aspiration and reservation levels must be no less than the corresponding utopia value and no greater than the corresponding nadir coefficient;
  - the aspiration level must be no greater than the corresponding reservation level;
- for each maximized objective (i.e., for Satisf and Prox):
  - the aspiration and reservation levels must be no greater than the corresponding utopia value and no less than the corresponding nadir coefficient;
  - the aspiration level must be no less than the corresponding reservation level.

In case of detecting some inconsistencies or errors in the coefficients you will be obliged to correct them.

At the beginning of the interactive analysis the so–called neutral solution is usually computed. For this purpose you should accept the utopia vector as the aspiration levels and the nadir vector as the reservation levels, by pressing the <ESC> key. Then you will be asked whether you will be interested in calculation of a new efficient solution with respect to the edited aspiration and reservation levels:

```
Do you wish to optimize? (y/n)
```

This question may seem to be unreasonable since usually you are interested in calculation of a new solution. However, a change of aspiration and reservation levels can be also very useful while performing some solutions comparisons and in such a situation you can use the EFFICIENT command only for the edition phase.

When you have decided to compute a new efficient solution the solver is invoked. Similarly as in the PAY-OFF command, the optimization process is reported by messages:

```
WAIT    Iterations:#   Solutions:#
```

and you can suspend computations at any moment (by pressing any key) and cancel or continue them after a break. After having finished the computations the new efficient solution is displayed in the left table as the Current Solution (central column) and the solution identification number appears at the top of the screen. From this moment you have available the so–called SUMMARY screens (by pressing the <PGUP> or <PGDN> key) for examination of the Current Solution in comparison with the previous one. There are four SUMMARY screens:

- Values;
- A/R Bars;
- U/N Bars;
- Locations.

At the top of each screen a header with the solution identification number and the screen name is displayed:

```
Efficient Solution No 5
```

The Values screen (see Fig. 22) provides you with numerical information about the objective values of the Current Solution. There is presented a table built of six columns. Each row of the table corresponds to one objective function. In several columns you can read the Utopia, Aspiration, Current, Previous, Reservation and Nadir values for the corresponding objective. From this table you may read that the investment cost for the neutral solution is rather high (Invest=386) whereas the other objectives get middling values (Satisf=276,, Dist=2.26, Prox=6457).

The A/R Bars screen (see Fig. 23) provides you with graphical presentation of the current objective values with respect to the given aspiration and reservation levels. Namely, for each objective separately the difference between the aspiration and reservation levels is taken as a unit, and the percentage degradation of the current objective value from the aspiration level is then calculated. The results are presented in form of the horizontal bars with accuracy of 1%.

More precisely, the aspiration and reservation levels as the basis of the analysis are fixed on the screen. Independently of the optimization type (minimization or maximization) the aspiration level is fixed on the left and the reservation level on the right. For each objective a dim bar shows the whole domain from the utopia value to the nadir value whereas the brighter segment covers the gap between the utopia value and the current value. So, the right side of the bright bar points out the current value. Moreover you are also provided with the objective value of the previous solution by a special sign below the bar. The value of the previous solution is pointed out by the top of the hat "^" (see Invest in Fig. 23) or by the right side of the small square "□" (see e.g., Satisf in Fig. 23).

The U/N Bars screen (see Fig. 24) provides you with a graphical presentation of the current objective values with respect to the utopia and nadir values. This screen is very similar to the A/R Bars. The main difference depend on using of the different scale for the bars. Namely in U/N Bars, the difference between the utopia and nadir values is taken as a unit and the percentage degradation of the current objective value from the utopia value is then calculated. Similarly as in A/R Bars the results are presented in form of the horizontal bars with accuracy of 1% and the same scheme for the current and previous values illustration is used. In our case the utopia and nadir values were taken

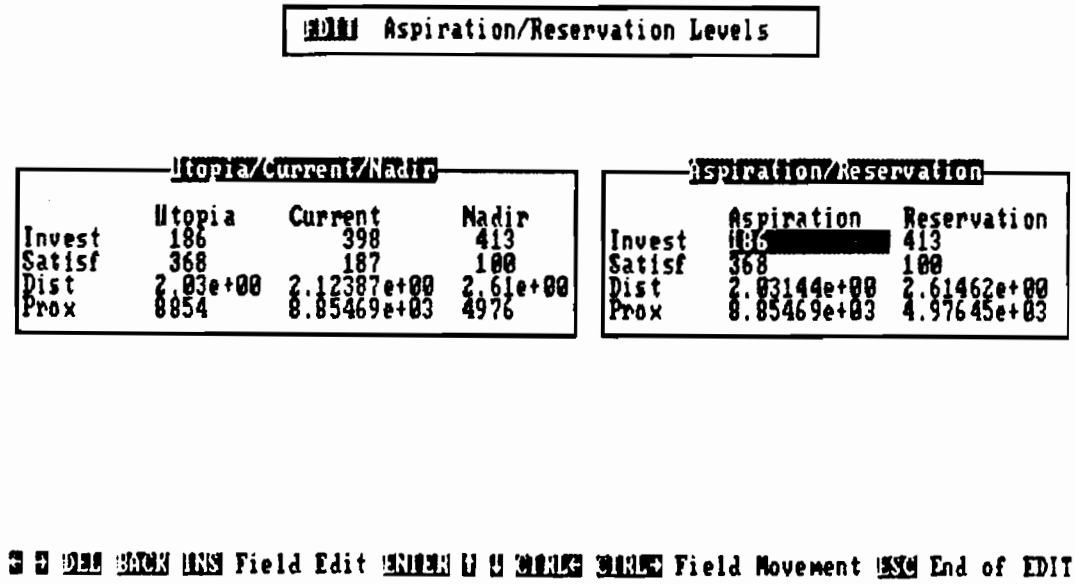


Figure 21: Aspiration/Reservation levels editing

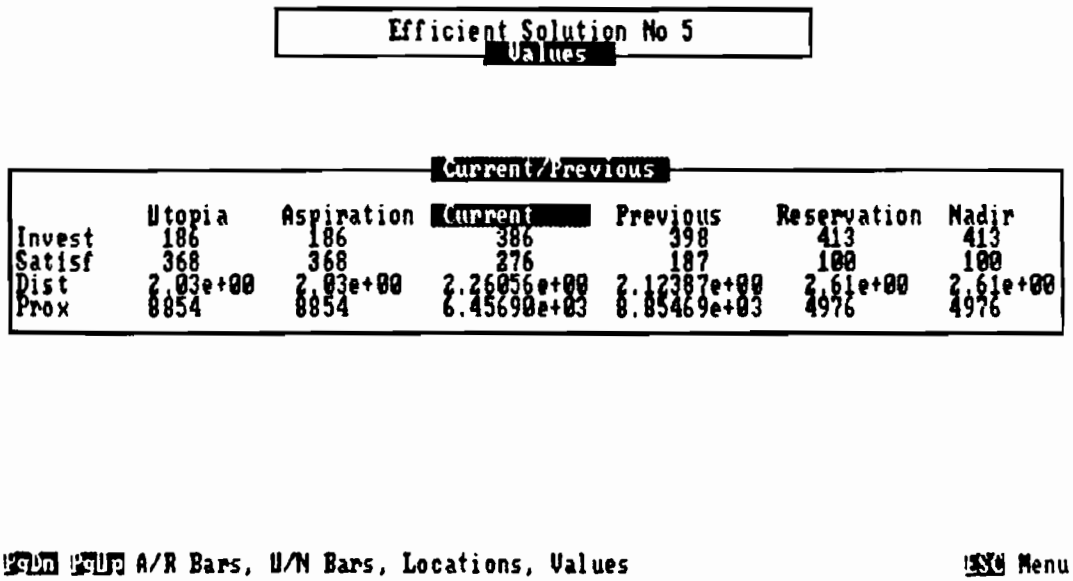


Figure 22: Values screen for the neutral solution

as the aspiration and reservation levels, respectively, and thereby both the bars screens have the same look–like. However, in general it is not true (see Fig. 26).

The Locations screen (see Fig. 25) provides you with the location decisions on which the Current Solution is based. There is displayed a very simple table built of two columns. The first column is associated with the Current Solution whereas the second one represents the previous solution. Each row of the table corresponds to one potential node. You can read from the table whether a potential node is active in the solution (denoted by Yes) or is not active (denoted by No). From the Locations screen you may read that the neutral solution is based on location of two new health–care centers at Bush and at Ice.

Apart from the solution connected with the minimization of the investment cost all the other solution are based on location of two new health–care centers what implies a high investment cost. Therefore we try to find an efficient solution with small investment cost (one new center) and relatively good values of the other objectives. For this purpose we suggest you to define the aspiration and reservation levels as it is given in Table 10.

	Invest	*Satisf*	*Dist*	*Prox*
aspiration	186	300	2.08	8500
reservation	250	200	2.50	7000

Table 10: Aspiration/reservation levels for solution no 6

In effect, you get the sixth efficient solution based on location of one new health–care center at Oasis. The investment cost is small (Invest=201), the satisfaction level has middling value (Satisf=192) while the average distance is very large (Dist=2.58) and the overall proximity is even less than the corresponding coefficient of the nadir vector (Prox=4933). Note that the system automatically corrects the nadir vector by putting the new worst value as the proper coefficient.

To avoid too small values of the overall proximity in the next solution we modify the reservation level for this objective putting 8000 as the new value. After repeating the computation we get the seventh efficient solution based on the sole new health–care center located at Ice. Due to the very convenient form of solution presentation in DINAS we can easy examine performances (in terms of objective values) of the new solution in comparison with the previous one (see Fig. 26). The overall proximity, the average distance and the investment cost are slightly better (Prox=5287, Dist=2.53 and Invest=200) while the overall satisfaction level is a few percent worse (Satisf=176).

After analysis of two last efficient solutions we make a supposition that it is necessary to relax requirements on the satisfaction level for making possibility to find an efficient solution with good values of the average distance and the overall proximity under small the investment cost. So, we change the reservation level associated with the function Satisf on 100. The system confirms our supposition. We get the eighth efficient solution based on the sole new health–care center located at Fiord. The solution guarantees quite large overall proximity (Prox=7691) and relatively small average distance (Dist=2.38) under small investment cost (Invest=212). On the other hand, the satisfaction level has a value even less than the corresponding coefficient of the nadir vector (Satisf=87). Despite of the latter the solutions seems to be very interesting among the other efficient solutions based on location of a sole health–care center.

Further search for a satisfying efficient solution based on only one new health–care center have finished without success. Namely, for different values of the aspiration and

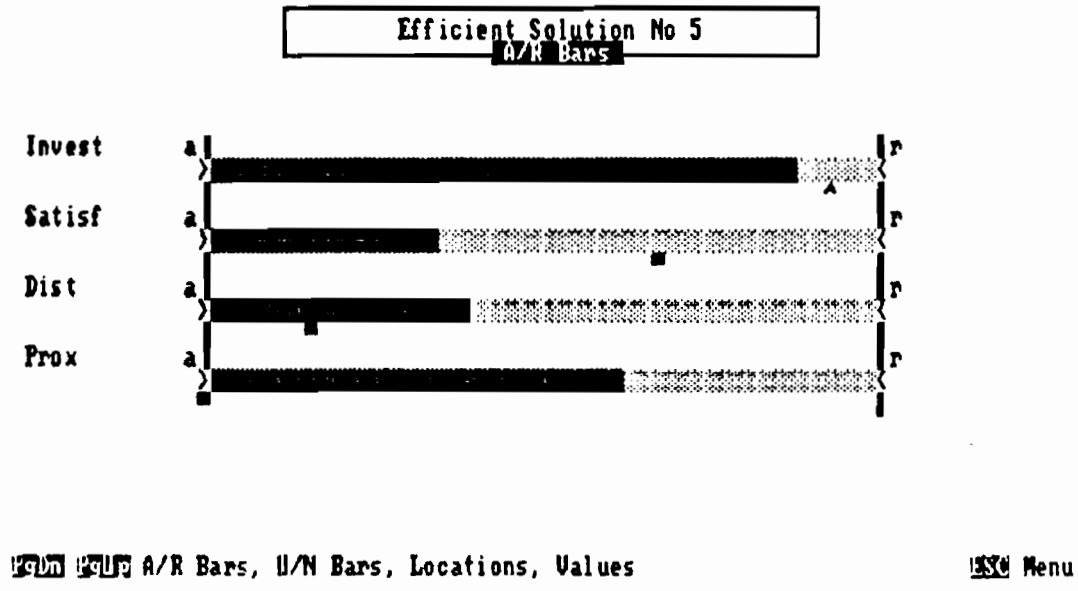


Figure 23: A/R Bars screen for the neutral solution

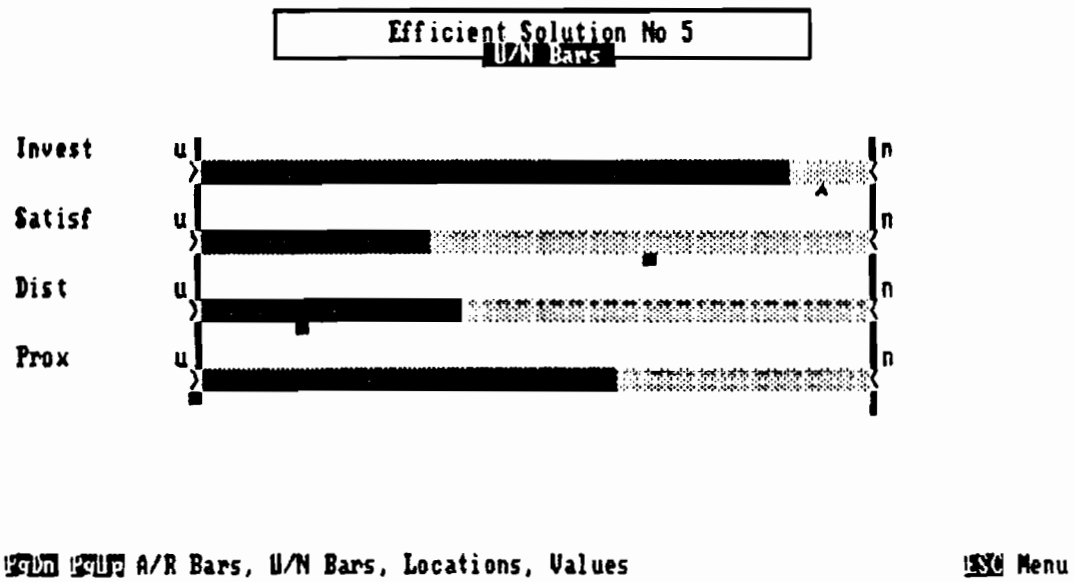


Figure 24: U/R Bars screen for the neutral solution

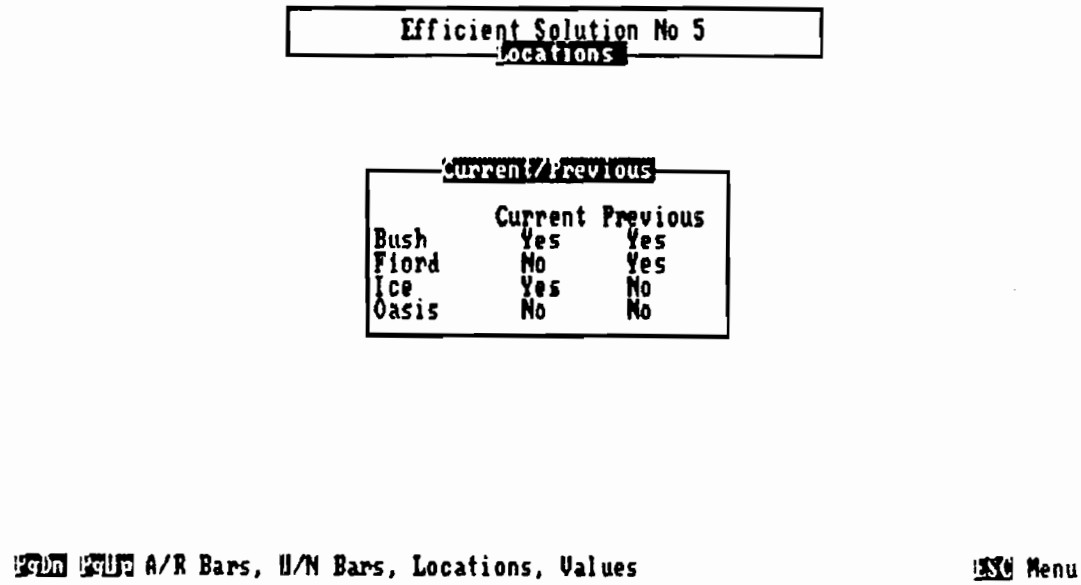


Figure 25: Locations screen for the neutral solution

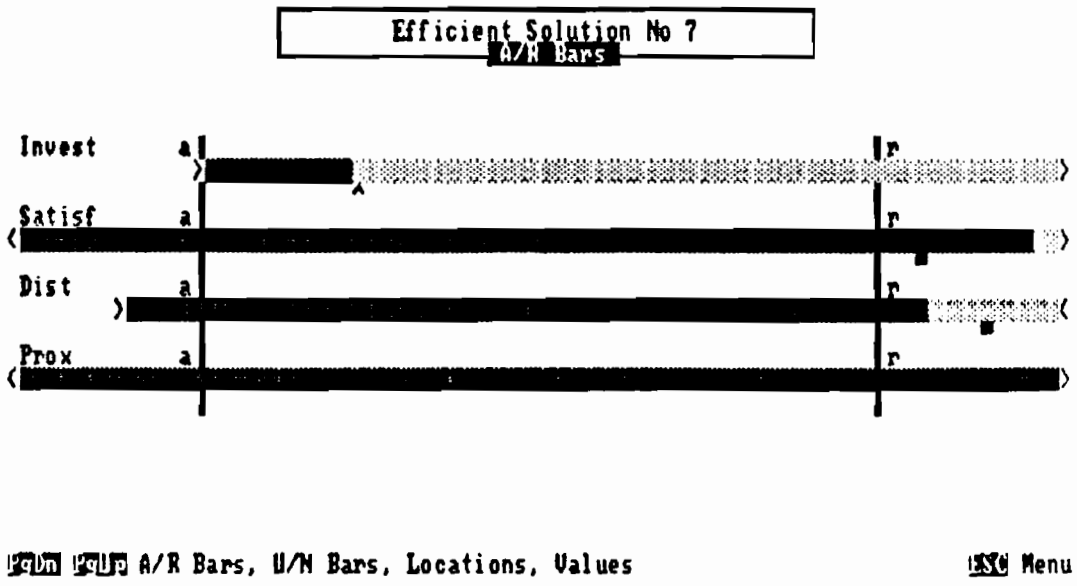


Figure 26: A/R Bars screen for the seventh solution

reservation levels the same efficient solutions have been generated. We suggest you to remove such solutions from the Solution Base immediately after the generation. For this purpose you can use the DELETE command from the SOLUTION branch. During the execution of the DELETE command the identification number of the Current Solution is displayed and you are asked to confirm your decision or cancel it. The corresponding message takes the form:

```
Do you wish to delete Efficient Solution No 9  (y/n)?
```

If you have confirmed the command then the Current Solution is deleted from the Solution Base whereas the previous solution is activated as the Current Solution and presented with the SUMMARY screens.

To complete the analysis you should try to examine another efficient solutions. For this purpose we suggest you to relax requirements on the investment cost. Among others, while using the aspiration and reservation levels given in Table 11 we get the eleventh efficient solution (the ninth and tenth solutions were deleted as duplications of the previous ones. It is based on the same location of new centers as the third solution (Fiord and Oasis) and thereby it gives the same investment cost (Invest=413) and satisfaction level (Satisf=279). However, the average distance and the overall proximity differs slightly (Dist=2.04 and Prox=8791).

	Invest	*Satisf*	*Dist*	*Prox*
aspiration	200	300	2.10	8800
reservation	400	200	2.50	8400

Table 11: Aspiration/reservation levels for solution no 11

Finally, you may examine all the generated efficient solutions using the COMPARE command. The solutions are listed in Figures 27 and 28. A careful analysis of these solutions leads us to the following conclusion. The investment cost cannot be regarded as a typical objective function since its values depend rather on the number of new health–care centers than on their locations. It only partitions all the efficient solutions into two groups: solutions based on a sole new health–care center and solutions based on location of two new centers. Therefore it is necessary to look for a good solution based on a sole new center which can be expanded to a better solution by adding the second new center. In our opinion, the first new health–care center should be located at Fiord (Solution 8). It is the only one efficient solution (based on a sole new center) which gives acceptable values of the average distance and the overall proximity (see Fig. 29). This solution gives also the worst value of the satisfaction level. However, further development of this solution by adding the new health–care center at Oasis (Solution 11) leads to a quite high value of the satisfaction level and makes further significant improvements with respect to the average distance and the overall proximity (see Fig. 30). Both the proposed solutions have the highest investment costs in the corresponding groups of solutions but variation of this objective among solutions of the same group is so small that it cannot be considered as a serious weakness.

Using the BROWSE command you can examine in details the selected solutions with the Network Editor. It turns out that in the eighth efficient solution the health–care center Hill is assigned to areas: Acer, Picea, Litwor, Betula and Erica. The health–care center Pond services areas: Bobrek, Arnika, Robur, Larix and a small part of Ribes. The

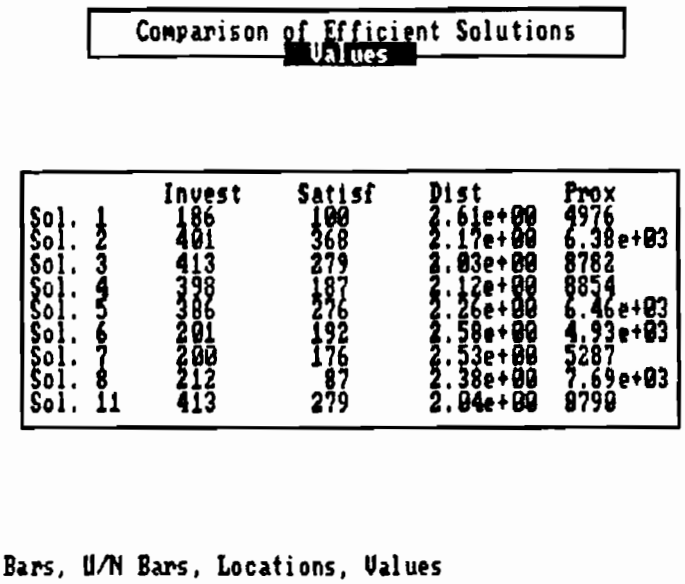


Figure 27: Values comparison

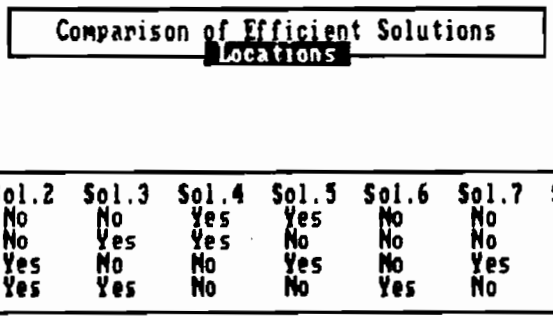


Figure 28: Locations comparison

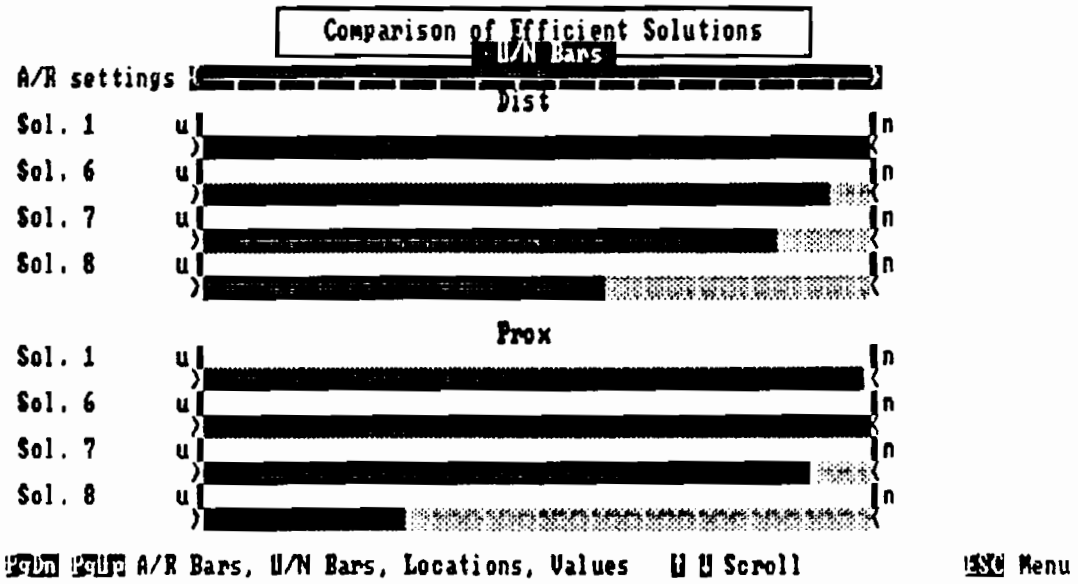


Figure 29: U/N Bars for solutions and the U/N scale for aspiration/reservation settings

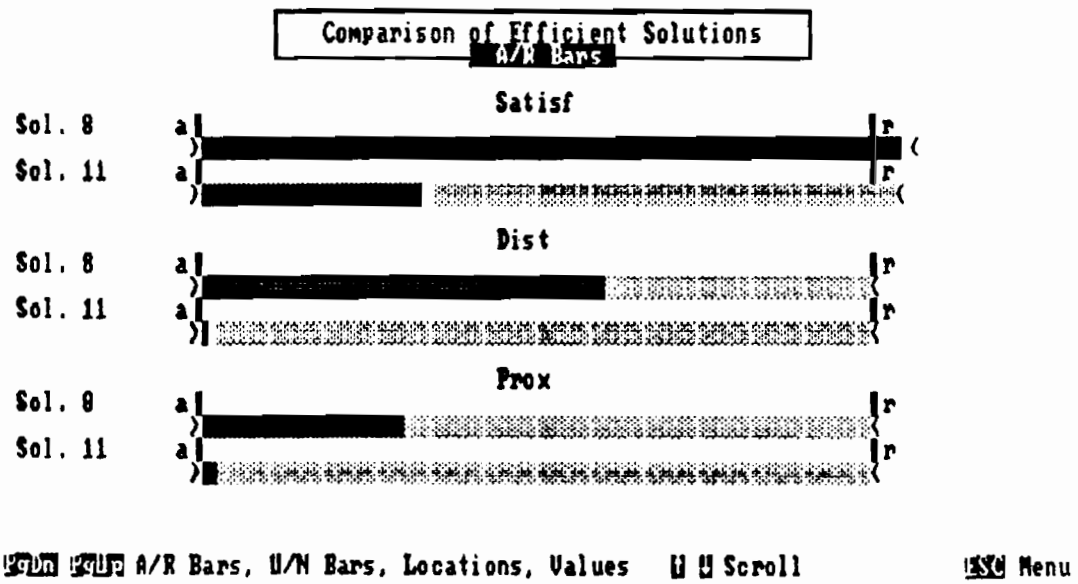


Figure 30: A/R Bars for two final solutions

new health–care center Fiord services only areas Pinus, Rumex and the main part of Ribes. Capacity of the new center Fiord is completely used whereas in the old centers we have noticed some small free capacities. It suggests that the old health–care centers have nonoptimal location with respect to the considered objective functions. The eleventh efficient solution confirms this observation. The additional new health–care center at Oasis takes the area Acer and the main part of Erica from the region of Hill and the area Bobrek from the region of Pond. So, in this solution both the new health–care centers use the whole their capacities whereas the old centers use only 50–70% of their capacities. In effect, all the health–care centers have well balanced charges.

Having performed the analysis you are usually interested in printing the selected solutions. For this purpose you should also use the **BROWSE** command. While executing the **BROWSE** command and selecting the **PRINT NETWORK** command from the Editor Menu the report on the Current Solution will be generated. In order to get the report on another efficient solution you have to use the **PREVIOUS** or **NEXT** command to change the Current Solution. The efficient solutions within the Solution Base are ordered according to the increasing identification numbers. The Next command causes that the next one after the Current Solution becomes a new Current Solution. On the other hand, the **PREVIOUS** command makes the solution that precedes the Current Solution to be a new Current Solution.

To leave the system you should select the **QUIT** command. **DINAS** requires a confirmation of your decision to leave the system and makes an opportunity to cancel it. Namely, you have to answer the question:

```
Do you wish to leave the system  (y/n)?
```

If you press the **<Y>** key then the control will be returned to the **DOS** operating system.

5 The Interactive System

5.1 Main Menu

To perform interactive analysis of the multiobjective problem with **DINAS** you need not to be familiar with neither computer techniques nor mathematical programming. **DINAS** is a menu–driven system with very simply commands. It is also equipped with the **HELP** file.

DINAS is started simply by typing command **DINAS**. The system cover is then visible on the display and there is a short pause while **DOS** finds and loads the execution program. When the loading procedure has already finished the blinking message:

```
Press any key to continue!
```

appears at the bottom of the display. Pressing **<ENTER>** or any other key, you will see the Main Menu screen with the system banner. From this moment the **HELP** file is still available (by pressing function key **<F1>**).

Operations available in the Main Menu are partitioned into three groups and corresponding three branches of the menu (see Table 12): **PROCESS**, **SOLUTION** and **ANALYSIS**.

The **PROCESS** branch contains basic operations connected with processing the multiobjective problem and generation of several efficient solutions. There are included problem definition operations such as calling the Network Editor for input or modification of the

Process	*Solution*	*Analysis*
Problem	Summary	Compare
Convert	Browse	Previous
Pay-Off	Save	Next
Efficient	Delete	Last
Quit		Restore

Table 12: Main Menu Commands

problem (**PROBLEM**) and converting of the edited problem with error checking (**CONVERT**). Further, in this branch the basic optimization operations are available: computation of the pay–off matrix with the utopia and nadir vectors (**PAY-OFF**) and generation of efficient solutions depending on the edited aspiration and reservation levels (**EFFICIENT**). As the last command in this branch is placed the **QUIT** operation which allows you to finish work with the system.

DINAS stores the efficient solutions in a special Solution Base. All the efficient solutions generated (or input from a file) during a session get consecutive numbers and are automatically put into the Solution Base. However at most nine efficient solution can be stored in the Solution Base. When the tenth solution is put into the base the oldest solution is automatically dropped from the base. On the other hand any efficient solution can be saved on a separate file and restored during the same or a subsequent session with the problem.

DINAS is armed with many operations which help to manage the Solution Base. There are two kinds of operations connected with the Solution Base: operations on a single efficient solution, and operations on the whole Solution Base. Operations addressed to a single solution are connected with the Current Solution. The newest generated efficient solution is automatically assumed to be the Current Solution but any efficient solution from the Solution Base can be manually assigned as the Current Solution.

The **SOLUTION** branch contains additional operations connected with the Current Solution. You can examine in details the Current Solution using the Network Editor (**BROWSE**) or analyse only short characteristics such as objective values and selected locations (**SUMMARY**). Values of the objective functions are presented in three ways: as a standard table, as bars in the aspiration/reservation scale and as bars in the utopia/nadir scale. The bars show percentage level of each objective value with respect to the corresponding scale. You may also save the Current Solution on a separate file in order to use it during next runs of the system with the same problem (**SAVE**). There is also available a special command to delete the Current Solution from the Solution Base if you find it as quite useless (**DELETE**).

The **ANALYSIS** branch collects commands connected with operations on the Solution Base. The main command **COMPARE** allows you to perform a comparison between all the efficient solutions included in the Solution Base or in some subset of the base. In the comparison only short characteristics of the solutions are used, i.e., objective values in the form of tables and bars as well as tables of selected locations. Moreover, some commands included in this branch (**PREVIOUS**, **NEXT** and **LAST**) allow you to select any efficient solution from the Solution Base as the Current Solution. There exist also an opportunity to restore some efficient solution (saved earlier on a separate file) to the Solution Base (**RESTORE**).

Command selection is performed with using a reverse image pointer which is controlled by the arrow keys. Namely, using the <LEFT> or <RIGHT> arrow you select a proper menu branch, and using the <UP> or <DOWN> arrow you select a proper item within the branch. Note that the pointer position within a branch is automatically restored during the next selection of the same branch. For execution of the selected command you have to press <ENTER>. After a command has been executed you need, usually, to press the <ESC> key in order to return control to the Main Menu.

Command selection may be also controlled by use of a mouse. If the mouse is installed (see part C) you can use it to select (and execute) with the pointer a command (or highlighted field) from the Main Menu as well as from submenus (or windows).

Control keys used in DINAS can be alternatively substituted by mouse operations as follows:

<LEFT> arrow — moving mouse to the left;

<RIGHT> arrow — moving mouse to the right;

<UP> arrow — moving mouse up;

<DOWN> arrow — moving mouse down;

<ENTER> — clicking the left button;

<ESC> — clicking the right button.

The mouse makes the system more user friendly since the user does not need continuously look at the keyboard during the session with DINAS.

Troubleshooting:

1. The Main Menu screen does not appear.
   Make sure that all the system files are available in the default directory.

2. After pressing the <F1> the following message:
   No help file - dinas.hlp press any key to continue
   appears in the help window.
   Make sure that the file DINAS.HLP is available in the default directory.

3. You cannot move the pointer across the Main Menu. Pressing any arrow key makes only the error noise.
   The <NUMLOCK> toggle is likely on. Make sure that <NUMLOCK> is off.

5.2 PROCESS branch

5.2.1 PROBLEM and CONVERT commands

Before the multiobjective analysis is launched, a file must be constructed describing the problem to be analysed. This file is called the network file. It is a special binary file generated by the Network Editor. The Network Editor is called by execution of the PROBLEM command.

Thus the PROBLEM command is usually selected as the initial step. In such a situation the Network Editor is called with an empty network file. The Editor Menu and the corresponding banner will appear on the display. Using the editor commands (see Chapter 6

for details) you will be able then to create a new network file (a new problem) or load and edit some existing one.

The network file created and saved during the execution of the PROBLEM command is not automatically assigned to the Interactive System. You may edit many various network files during a single editor calling. To assign a specific network file to the system you have to execute the CONVERT command. The execution of the CONVERT command makes the specified problem (network file) available for the system.

After selecting the CONVERT command you will be asked for a specification of the problem name. For this purpose a special 38–character field is displayed preceded by the message:

```
GIVE Problem Name:
```

You can then edit the name according to the general field editing rules (see Appendix A). Next the specified problem data will be converted with error checking. Any recognized data error will be reported on the display. In case of a successful conversion the problem statistics will be reported, i.e., the table containing: name of the problem, numbers of objectives, arcs, fixed nodes, potential nodes and selections will be displayed.

If you have prepared a network file during some previous session and do not change the problem then you can, obviously, initialize the analysis by direct execution of the CONVERT command without using the PROBLEM command.

You can execute the PROBLEM and CONVERT commands many times during the same DINAS session. However, if some problem is actually assigned to the system then effects of the commands executions will differ slightly in comparison with the initial ones. While executing the CONVERT command the current problem statistics will be displayed and you will be asked for confirmation that the current problem have to be abandoned, i.e., the following question will appears:

```
Do you wish to abandon Current Problem (y/n)?
```

Depending on your answer the command execution will be canceled or the standard converting procedure will be continued.

While executing the PROBLEM command the actually assigned problem is automatically loaded into the editor and therefore the message:

```
Reading Network
```

appears before displaying the Editor Menu. It is assumed that you are interested in data examination or making some changes in the problem under analysis. Nevertheless, if you want you can load another problem using the editor commands.

Multiple usage of PROBLEM and CONVERT commands during a single DINAS session can be very useful for more complex multiobjective analyses. However, to avoid troubles you must then observe the following rules:

- do not save the problem if you have not modified the problem;
- execute CONVERT command if you have modified the problem.

Troubleshooting:

1. After executing of the PROBLEM command the Network Editor is not activated. The Network Editor cannot be loaded. Make sure that all the system files are in the default directory. Make sure that there are 640 KB RAM available in your PC.

2. While executing of the CONVERT command the following message appears:
   Some Problems with NETWORK FILE
   PRESS ANY KEY TO CONTINUE !
   The specified network file does not exist or is not available. Make sure that you use the proper name of the network file. Make sure that the network file is in the default directory or use the proper path.

3. While executing of the CONVERT command the following message appears:
   Some Problems with WORKING FILE
   PRESS ANY KEY TO CONTINUE !
   A working file for the Solution Base cannot be opened in the default directory. Make sure that there is a possibility to open a new file in the default directory (the disk is not write–protected, the door is closed, etc.).

4. While executing of the CONVERT command the following message appears:
   DISK FULL !!!
   PRESS ANY KEY TO CONTINUE !
   There is no sufficient space for the Solution Base on the default disk. Delete some useless (network and solution) files or put them on another disk (see Appendix D).

5. While executing of the CONVERT command the following message appears:
   DATA ERROR !
   error description
   Do you wish to continue ? (y/n)
   An error in the network file found (see Section 5.2.2 for details). If you press the **<Y>** key the conversion will be continued, if possible, to recognize another errors. If you press the **<N>** key the conversion will be terminated prematurely. In both the cases no problem will be assigned to the Interactive System.

6. While executing of the CONVERT command the following message appears:
   EXECUTION INTERRUPTED !
   Do you wish to continue ? (y/n)
   You have pressed some key and in result the command execution is suspended. You can cancel the command execution by pressing the **<N>** key or continue it by pressing the **<Y>** key.

5.2.2 Data errors

There are many various restrictions on the problem data. They can be partitioned into three groups: size limits, restriction on the network structure, restriction on several data coefficients. Some of the restrictions (especially from the last group) are supported by the Network Editor in such a way that you cannot input a wrong coefficient. However many of the restrictions cannot be verified before the problem definition is finished. Therefore during the execution of the CONVERT command all the problem data are carefully examined with respect to the restrictions. In this section you will find detail description of the data restrictions and corresponding error messages issued by the CONVERT command.

For the basic version of the DINAS system the following size limits are valid:

number of active objectives - 7
number of arcs - 300
number of fixed nodes - 100

number of potential nodes - 15
number of selections - 8

However the given numbers define only maximal size of the problem with respect to a single quantity. We do not guarantee a possibility to solve the problem with all the quantities equal to their maximal sizes simultaneously. Namely, the system allocates dynamically necessary data structures and there exist an internal limit on the total size of the data structures. There are two restrictions on the network structure:

- the network cannot contains any simple loop;
- the network cannot contain any arc which joins directly two potential nodes.

Both the restriction are supported by the Network Editor (see Chapter 6) in such a way that you cannot input a wrong arc.

The several data coefficients have to fulfill the following requirements:

- the balances should be real numbers and the sum of all the balances have to be equal to 0;
- the objective coefficients should be real numbers;
- the arc capacities should be nonnegative real numbers;
- the potential node capacities should be positive real numbers;
- the selection bounds should be nonnegative integer numbers and for a given selection its lower bond cannot exceed its upper bound.

A quantity ρ is accepted by DINAS as a proper real number if it satisfies the following inequality:

$$-1.0e30 \leq r \leq 1.0e30.$$

Troubleshooting:

1. While executing of the CONVERT command a DATA ERROR message appears like these:

   ```
   Too many OBJECTIVES !
   Too many ARCS !
   Too many FIXED NODES !
   Too many POTENTIAL NODES !
   Too many SELECTIONS !
   ```

   The current number of objectives, arcs, fixed nodes, potential nodes or selections, respectively, exceeds the corresponding size limit. Verify whether your problem really need such a large number of given objects. If there is no possibility to rebuilt the model, it cannot be solved with this version of the system.

2. While executing of the CONVERT command a DATA ERROR message appears like these:

   ```
   There are NO OBJECTIVES !
   There are NO ARCS !
   There are NO FIXED NODES !
   ```

   An incomplete problem formulation. Correct your problem.

3. While executing of the CONVERT command a DATA ERROR message appears like these:

   Arc *name* INVALID CAPACITY !

Node *name* **INVALID CAPACITY !**

An invalid capacity for the given arc or potential node. Correct the corresponding data (remember that the capacities for arcs must be nonnegative and capacities for potential nodes must positive).

4. While executing of the CONVERT command a DATA ERROR message appears like these:

   Arc *name* **INVALID COEFFICIENT !**

   Node *name* **INVALID COEFFICIENT !**

   An invalid cost coefficient for the given arc or potential node. Correct the corresponding data.

5. While executing of the CONVERT command a DATA ERROR message appears like this:

   Node *name* **INVALID BALANCE !**

   An invalid balance for the given fixed node. Correct the corresponding data.

6. While executing of the CONVERT command a DATA ERROR message appears like this:

   Selection *name* **INVALID BOUNDS !**

   Invalid lower and/or upper bounds for the given selection. Correct the corresponding data (remember that the lower bound cannot exceed the upper bound).

7. While executing of the CONVERT command DATA ERROR messages of type (3) to (6) permanently appear for all the data.

   Probably you use a version of the system which takes advantage of the Numeric Data Processor whereas your computer is not armed with such a hardware. Change the version of the system.

8. While executing of the CONVERT command a DATA ERROR message appears like this:

   Arc *name* **INVALID CONNECTION !**

   An illegal arc which connects two potential nodes. Correct the network structure. In order to define a connection of two potential nodes you can use an artificial fixed node.

9. While executing of the CONVERT command a DATA ERROR message appears like this:

   Problem NOT BALANCED !

   The sum of all the balances is not equal to zero. Verify all the balances whether they are correct. If your problem is unbalanced in fact you can redefine it as balanced one by introducing an artificial fixed node.

10. While executing of the CONVERT command a DATA ERROR message appears like this:

    INSUFFICIENT MEMORY !

    The problem is too large to solve with this version of the system. Try to decrease the size of the problem, especially numbers of objectives and potential nodes.

5.2.3 PAY-OFF command

When the problem has been defined and converted you must execute the PAY-OFF command as the first step of the multiobjective analysis . It performs optimization of each objective function separately. In effect, you get the so–called pay–off matrix. The pay–off matrix is a well–known device in MCDM. It is displayed as a table containing values of all the objective functions (columns) obtained while solving several single–objective problems

(rows) and thereby it helps to understand the conflicts between different objectives (see Fig 19).

The execution of the PAY-OFF command provides also you with two reference vectors: the utopia vector and the nadir vector. They are displayed as a two–column table below the pay–off matrix (see Fig. 19). The utopia vector represents the best values of each objective considered separately, and the nadir vector express the worst values of each objective noticed during optimization of another objective function. The utopia vector is, usually, not attainable, i.e., there are no feasible solutions with such objective values.

Coefficients of the nadir vector cannot be considered as the worst values of the objectives over the whole efficient (Pareto–optimal) set. They usually estimate these values but they express only the worst values of each objective noticed during optimization of another objective function. In further analysis these estimations can be sometimes overstep.

Due to a special regularization technique used while computating the pay–off matrix (see Chapter 3) each generated single–objective optimal solution is also an efficient solution to the multiobjective problem. So, after the calculation of the pay–off matrix you have already available in the Solution Base a number of efficient solutions connected with several rows of the pay–off matrix.

The pay–off matrix calculation is, usually, the most time–consuming operation of the multiobjective analysis. Therefore DINAS automatically saves the computed pay–off matrix on the network file. However, the corresponding efficient solutions are not saved. So, if you restart the analysis of the problem (without any changes in formulation) the PAY-OFF command causes only restoration of the pay–off matrix without generation any efficient solution in the Solution Base. Editing of the problem (exactly: using the SAVE command in the Network Editor) cancels the pay–off record on the network file.

While performing the optimization the computation process is reported by the messages:

```
WAIT    Iterations: #   Solutions: #
```

where the numbers represent the number of simplex steps having performed and the number of feasible solutions having examined, respectively. You can suspend the computations at any moment (by pressing <ENTER> or any other key) and cancel them or continue after a break.

Troubleshooting:

1. While executing of the PAY-OFF command the following message appears:

   ```
   Problem NOT AVAILABLE !
   SELECT CONVERT COMMAND !
   ```

   There is no problem assigned to the Interactive System. Press the <ESC> key to return control to the Main Menu and execute the CONVERT command.

2. While executing of the PAY-OFF command a message appears in the pay–off table, like this:

   ```
   INFEASIBLE PROBLEM !
   UNBOUNDED PROBLEM !
   ```

   During the optimization process the corresponding single–objective problem proved to be infeasible or unbounded respectively. You have to correct the problem formulation. Press the <ESC> key to return control to the Main Menu.

3. While executing of the PAY-OFF command the following message appears:

   EXECUTION INTERRUPTED !
   Do you wish to continue ? (y/n)

   You have pressed some key and in result the command execution is suspended. You can cancel the command execution by pressing the <N> key or continue it by pressing the <Y> key.

4. While executing of the PAY-OFF command the following message appears in the pay–off table:

   USER's BREAK !

   You have canceled the command execution (see point (3)). Press the <ESC> key to return control to the Main Menu.

5. While executing of the PAY-OFF command the following message appears in the pay–off table:

   RESTORING PROBLEMS !

   The system came across some problems while restoring the pay–off record from the network file. Press the <ESC> key to return control to the Main Menu and execute the PAY-OFF command again. The pay–off matrix will be then computed.

6. While executing of the PAY-OFF command the following message appears in the pay–off table:

   NUMERICAL PROBLEMS !

   The system came across some numerical problems and cannot overcome them. So, your problem cannot be solved.

5.2.4 EFFICIENT command

Having executed the PAY-OFF command you can start the interactive search for a satisfying efficient solution. DINAS utilizes aspiration and reservation levels to control the interactive analysis. More precisely, you specify acceptable values for several objectives as the aspiration levels, and necessary values as the reservation levels. All the operations connected with editing the aspiration and reservation levels as well as with computation of a new efficient solution are performed within the EFFICIENT command.

After selecting the EFFICIENT command you are asked for aspiration and reservation levels editing. The message:

EDIT Aspiration/reservation Levels

appears at the top of the screen whereas two tables are displayed at the centre (see Fig. 21). The left table contains the objective values for the Current Solution (central column) in comparison with the utopia (left column) and nadir (right column) vectors. The right table is a small spreadsheet for aspiration and reservation levels edition. You can select several fields and edit the corresponding coefficients. A twelve–character (reverse image) field pointer is controlled by the <ENTER> key which move the pointer entry by entry. Moreover, you can use the <UP> or <DOWN> arrows for a direct vertical movement and, similarly, the <CTRL>/<LEFT> or <CTRL>/<RIGHT> keys for a direct horizontal movement. At the beginning of the interactive analysis the aspirations and reservation levels are initialized as equal to the utopia and nadir vectors, respectively.

The aspiration and reservation levels may be also defined approximately by use of the COMPARE command while analyzing the group of utopia/nadir bars corresponding to

several Comparison Base solutions. This way is easy and strongly recommended since it allows to define aspirations/reservations without strenuous editing their numerical values. Specifying aspiration and reservation levels in this way is described in details in Section 5.4.1.

You can finish the edition phase at any moment by pressing the <ESC> key. The system will check then the introduced coefficients. To be accepted by the system the edited coefficients have to satisfy the following relations:

- for each minimized objective :
  - the aspiration and reservation levels must be no less than the corresponding utopia value and no greater than the corresponding nadir coefficient;
  - the aspiration level must be no greater than the corresponding reservation level;
- for each maximized objective :
  - the aspiration and reservation levels must be no greater than the corresponding utopia value and no less than the corresponding nadir coefficient;
  - the aspiration level must be no less than the corresponding reservation level.

In case of detecting some inconsistencies or errors in the coefficients you will be obliged to correct them.

After successful finishing of the editing phase you will be asked whether you will be interested in calculation of a new efficient solution with respect to the edited aspiration and reservation levels:

```
Do you wish to optimize? (y/n)
```

This question may seem to be unreasonable since usually you will be interested in calculation of a new solution. However, a change of aspiration and reservation levels can be also very useful while performing some solutions comparisons and in such a situation you can use the EFFICIENT command only for the editing phase.

When you have decided to compute a new efficient solution the solver is invoked.

Similarly as in the PAY-OFF command, the optimization process is reported by messages:

```
WAIT    Iterations: #   Solutions: #
```

and you can suspend computations at any moment (by pressing any key) and cancel or continue them after a break. After having finished computations the new efficient solution is displayed in the left table as the Current Solution (central column) and the solution identification number appears at the top of the screen. From this moment you have available four SUMMARY screens (by pressing the <PGUP> or <PGDN> key) for examination of the Current Solution in comparison with the previous one. After solution examination you leave the EFFICIENT command (by pressing the <ESC> key) to select another command from the SOLUTION or ANALYSIS branch, or to repeat the EFFICIENT command. You have to leave the EFFICIENT command even if you want to repeat it.

Troubleshooting:

1. While executing of the EFFICIENT command the following message appears:
   Problem NOT AVAILABLE !
   SELECT CONVERT COMMAND !
   There is no problem assigned to the Interactive System. Press the <ESC> key to return control to the Main Menu and execute the CONVERT command.

2. While executing of the EFFICIENT command the following message appears:
   Utopia & Nadir Vectors are NOT AVAILABLE !
   SELECT PAY-OFF COMMAND !
   The PAY-OFF command has to be executed before first using of the EFFICIENT command. Press the <ESC> key to return control to the Main Menu and execute the PAY-OFF command.

3. While executing of the EFFICIENT command the following message appears:
   There is ONLY ONE OBJECTIVE !
   SELECT PAY-OFF COMMAND !
   Your problem is a single–objective one. You will get an optimal solution by using the PAY-OFF command. Press the <ESC> key to return control to the Main Menu and execute the PAY-OFF command.

4. After editing of the aspiration and reservation levels the following message appears:
   Illegal Aspiration/Reservation Levels !
   PRESS ANY KEY TO CONTINUE !
   The edited aspiration and/or reservation levels do not satisfy necessary inequalities. The names of the objectives with illegal quantities are displayed in the reverse form. Press <ENTER> or any other key and edit the aspiration and reservation levels again.

5. While computing of a new efficient solution the following message appears:
   EXECUTION INTERRUPTED !
   Do you wish to continue ? (y/n)
   You have pressed some key and in result the command execution is suspended. You can cancel the command execution by pressing the <N> key or continue it by pressing the <Y> key.

6. While computing of a new efficient solution the following message appears:
   USER's BREAK !
   You have canceled the command execution (see point (5)). Press the <ESC> key to return control to the Main Menu.

7. While computing of a new efficient solution the following message appears:
   NUMERICAL PROBLEMS !
   The system came across some numerical problems and cannot overcome them. So, your problem cannot be solved.

5.2.5 QUIT command

The QUIT command allows you to leave the DINAS system and return control to the DOS Operating System. Leaving the DINAS system makes the Solution Base be lost. Therefore before execution of the QUIT command you must be sure that you have saved all the solutions which are intended to be used in a further analysis. To prevent you from accidental loss of the efficient solutions while executing the QUIT command, DINAS

requires a confirmation of your decision to leave the system and makes an opportunity to cancel it. Namely, you have to answer the question:

Do you wish to leave the system (y/n)?

If you have pressed the **<N>** key then the control will return to the **DINAS** Main Menu.

5.3 SOLUTION branch

5.3.1 SUMMARY command

The **SUMMARY** command provides you with the most important but compressed characteristics of the Current Solution. It allows you to analyse in terms of the objective values and location decisions the Current Solution in comparison with the previous one. Execution of the **SUMMARY** command makes available four screens:

- Values;
- A/R Bars;
- U/N Bars;
- Locations.

At the top of each screen a header with the solution identification number and the screen name is displayed.

The Values screen (see Fig. 22) provides you with numerical information about the objective values of the Current Solution. There is presented a table built of six columns. Each row of the table corresponds to one objective function. In several columns you can read the Utopia, Aspiration, Current, Previous, Reservation and Nadir values for the corresponding objective.

The A/R Bars screen (see Fig. 23 or 26) provides you with graphical presentation of the current objective values with respect to the given aspiration and reservation levels. Namely, for each objective separately the difference between the aspiration and reservation levels is taken as a unit and the percentage degradation of the current objective value from the aspiration level is then calculated. The results are presented in form of the horizontal bars with accuracy of 1%.

More precisely, the aspiration and reservation levels as the basis of the analysis are fixed on the screen. Independently of the optimization type (minimization or maximization) the aspiration level is fixed on the left and the reservation level on the right. For each objective a dim bar shows the whole domain from the utopia value to the nadir value whereas the brighter segment covers the gap between the utopia value and the current value. So, the right side of the bright bar points out the current value. Moreover you are also provided with the objective value of the previous solution by a special sign below the bar. The value of the previous solution is pointed out by the top of the hat ($\hat{}$) or by the right side of the small square ($\square$).

You can define various aspiration and reservation levels during the execution of the **EFFICIENT** command. For a narrow distance between the aspiration and reservation levels in a comparison with the whole utopia/nadir domain you can get only part of the bar on the screen. To make an opportunity to recognize such a situation the bars are bordered by the signs > on the left side and < on the right side. If a bar has cut on the screen then you will perceive the sign < on the left side of the screen or the sign > on the right one.

The U/N Bars screen (see Fig. 24) provides you with a graphical presentation of the current objective values with respect to the utopia and nadir values. This screen is very

similar to the A/R Bars. The main difference depend on using of the different scale for the bars. Namely in U/N Bars, the difference between the utopia and nadir values is taken as a unit and the percentage degradation of the current objective value from the utopia value is then calculated. Similarly as in A/R Bars the results are presented in form of the horizontal bars with accuracy of 1% and the same scheme for the current and previous values illustration is used. In U/N Bars the utopia and nadir values are fixed on the screen. Thus the whole domain and thereby the whole bars are always visible.

The Locations screen (see Fig. 25) provides you with the location decisions on which the Current Solution is based. There is displayed a very simple table built of two columns. The first column is associated with the Current Solution whereas the second one represents the previous solution. Each row of the table corresponds to one potential node. You can read from the table whether a potential node is active in the solution (denoted by Yes) or is not active (denoted by No).

The **SUMMARY** screens are controlled by the **<PGDN>** or **<PGUP>** keys which display screen by screen forward or backward, respectively. To return control to the Main Menu you have to press the **<ESC>** key.

Apart from the direct selection in the Main Menu, the **SUMMARY** command is also executed by many other system commands. Namely, each command which implies a change of the Current Solution is finished with the activation of the **SUMMARY** screens for the new Current Solution.

Troubleshooting:

1. While executing of the **SUMMARY** command the following message appears:

   There are NO SOLUTIONS !
   SELECT ANOTHER COMMAND !

   The Solution Base is empty. Press the **<ESC>** key to return control to the Main Menu and execute another command (e.g., the **EFFICIENT** command for solution generation).

5.3.2 BROWSE command

The **BROWSE** command allows you to examine in details the Current Solution using the Network Editor. The execution of this command causes calling of the Network Editor in the view mode. The network file under consideration with linked Current Solution information is then automatically loaded into the editor. You can examine flows along several arcs and through several potential nodes, as well as activity of the potential nodes and objective values. There are no any automatic schemes of examination. You have to decide which characteristics or which part of the network you want to examine . Simply you can browse the Current Solution.

During the execution of the **BROWSE** command some editor file operations are not available as unreasonable. Namely, you cannot load any network file as well as you cannot save the problem. The execution of the **PRINT** command generates then the solution report instead of the standard network report (see 6.3 for details).

Troubleshooting:

1. While executing of the **BROWSE** command the following message appears:

   There are NO SOLUTIONS !

SELECT ANOTHER COMMAND !

The Solution Base is empty. Press the <ESC> key to return control to the Main Menu and execute another command (e.g., the EFFICIENT command for solution generation).

2. After executing of the BROWSE command the Network Editor is not activated.
   The Network Editor cannot be loaded. Make sure that all the system files are in the default directory. Make sure that there are 640 KB RAM available in your PC.

3. While executing of the BROWSE command the following message appears:
   Some problems with NETWORK FILE !
   PRESS ANY KEY TO CONTINUE !
   The network file was changed during the analysis and solutions from the Solution Base can be improper for the current problem. You should restart the analysis with the CONVERT and PAY-OFF commands. Press any key to return control to the Main Menu.

4. While executing of the BROWSE command the following message appears:
   Some Problems with WORKING FILE
   PRESS ANY KEY TO CONTINUE !
   A working file with the Solution Base cannot be found in the default directory. Make sure that you have not changed a diskette during the work and the door is closed. Press any key to return control to the Main Menu.

5.3.3 SAVE command

The SAVE command generates a special Solution File for the Current Solution. This file can be used later during the same DINAS session or even during one of subsequent sessions for restoration of the efficient solution. Thus the SAVE command allows you to make the Current Solution be save for further analysis. You can, certainly, save any efficient solution from the Solution Base having made it previously the current one with commands from the ANALYSIS branch (see Section 5.4.2).

There are two reasons for using of the SAVE command. The first one is connected with conducting a time–consuming analysis which cannot be finished during a single session. You should save then all the interesting efficient solutions before breaking off the analysis and leaving the DINAS system. The second reason is associated with performing a complex analysis where there are many (greater than nine) interesting solutions. The Solution Base is then too small and to prevent from loss you must save some solutions on separate files.

To avoid mistakes, while executing of the SAVE command the identification number of the Current Solution is displayed and you are asked to confirm your decision or cancel it. The corresponding message takes the form:

```
Do you wish to save Efficient Solution No # (y/n)?
```

Next, if you have decided to save the solution you are asked for a name for the solution file. For this purpose a special eight–character field is displayed preceded by the message:

```
GIVE  Solution File Name:
```

You can then edit the name according to the general field editing rules (see Appendix A). At the beginning the field is initialized with the name SOLUTION. During subsequent

executions in the field appears the latest used name of the solution file (in SAVE or RESTORE commands). During saving the solution the following message appears:

WAIT Saving Efficient Solution No # as *filename*

where *filename* represents the name of the solution file.

Warning: To avoid accidental loss of solution files you have to use different names for different efficient solutions.

The solution file is generated in the default directory as a binary file with the given name and extension EFF. For instance having given the name SOLUTION the system will generate the solution file SOLUTION.EFF.

To avoid various troubles with the saved files you should observe the following rules:

- after the session copy all the important solution files from the default directory;
- after the session delete from the default directory all the solution files which are useless;
- before the session copy all the necessary solution files to the default directory.

Troubleshooting:

1. While executing of the SAVE command the following message appears:
   There are NO SOLUTIONS !
   SELECT ANOTHER COMMAND !
   The Solution Base is empty. Press the <ESC> key to return control to the Main Menu and execute another command (e.g., the EFFICIENT command for solution generation).

2. While executing of the SAVE command the following message appears:
   Some problems with FILE: *name*
   The specified solution file cannot be opened in the default directory. Make sure that there is a possibility to open a new file in the default directory (the disk is not write–protected, the door is closed, etc.).

3. While executing of the SAVE command the following message appears:
   DISK FULL !!!
   PRESS ANY KEY TO CONTINUE !
   There is no sufficient space for the solution file on the default disk. Delete some useless (network and solution) files or put them on another disk (see Appendix D).

4. While executing of the SAVE command the following message appears:
   Some Problems with WORKING FILE
   PRESS ANY KEY TO CONTINUE !
   A working file with the Solution Base cannot be found in the default directory. Make sure that you have not changed a diskette during the work and the door is closed. Press any key to return control to the Main Menu.

5.3.4 DELETE command

The DELETE command is connected with the Current Solution but it help you to manage the Solution Base. In the Solution Base can be stored at most nine solutions and while the tenth solution is putting into the base the oldest solution is automatically deleted. To avoid loss of the interesting efficient solution you can delete manually another solution

before the generation of a new efficient solution. The DELETE command delete the Current Solution from the Solution Base. It is recommended to delete not promising or duplicated efficient solution immediately after the generation when it is the Current Solution. Nevertheless you can delete any solution from the Solution Base having used previously some commands from the ANALYSIS branch (see Section 5.4.2) for activation this solution as the Current Solution.

To prevent you from an accidental lost of the important efficient solution, while executing of the DELETE command the identification number of the Current Solution is displayed and you are asked to confirm your decision or cancel it. The corresponding message takes the form:

```
Do you wish to delete Efficient Solution No #  (y/n)?
```

If you have confirmed the command then the Current Solution is deleted from the Solution Base whereas the previous solution is activated as the current one and presented with the SUMMARY screens. If there is no previous solution the last solution will be selected as the Current Solution. For availability of the DELETE command the Solution Base must contain at least two solutions.

Deleting of the efficient solution from the Solution Base does not erase the corresponding solution file if the solution has been previously saved (see Section 5.3.3). So, you can restore later such a solution from the solution file to the Solution Base (see Section 5.4.3). However, while restoring the solution get a new identification number.

Troubleshooting:

1. While executing of the DELETE command the following message appears:
   ```
   There are NO SOLUTIONS !
   SELECT ANOTHER COMMAND !
   ```
   The Solution Base is empty. Press the <ESC> key to return control to the Main Menu and execute another command (e.g., the EFFICIENT command for solution generation).

2. While executing of the DELETE command the following message appears:
   ```
   There is ONLY ONE SOLUTION !
   SELECT ANOTHER COMMAND !
   ```
   The Solution Base contains only one solution and it cannot be deleted. Press the <ESC> key to return control to the Main Menu and execute another command (e.g., the EFFICIENT command for solution generation).

5.4 ANALYSIS branch

5.4.1 COMPARE command

The COMPARE command is the main command of the ANALYSIS branch. It allows you to compare any set of the efficient solution included in the Solution Base with respect to all the characteristics supported by the SUMMARY screens. So, with the COMPARE command you can get objective values, aspiration/reservation bars, utopia/nadir bars and location activities for all the selected solutions simultaneously.

At the beginning of the COMPARE command execution you have to select solutions included into the comparison. For this purpose a special (temporal) Comparison Base is

used. The Comparison Base is initialized as containing all the efficient solutions from the Solution Base. You can remove some solutions from the Comparison Base and thereby select the remaining solutions for the comparison.

The process of the Comparison Base definition is organized as follows. At the top of the screen the message appears:

`SELECT Efficient Solutions for Comparison`

and simultaneously the initial Comparison Base is displayed. The Comparison Base is displayed as a table where each row corresponds to a single efficient solution while the columns correspond to several objective functions. In each row you can read the identification number of the efficient solution and objective values for several criteria. There is available a reverse–image pointer which allows you to select a solution by pointing out its identification number. The pointer is controlled by using the **<UP>** and **<DOWN>** arrows. If you have pressed the **<DEL>** key then the pointed out solution is removed from the Comparison Base and the corresponding row is deleted from the table on the screen. In such a way you can eliminate from the Comparison Base the solutions which you are not interested in during the comparison. You may finish the process of the Comparison Base definition at any moment by pressing the **<ESC>** key.

Remark: Do not be afraid to use the **<DEL>** key. The Comparison Base is only temporal for a single execution of the `COMPARE` command. All the solutions remain still available in the Solution Base.

For making a comparison at least two items in the Comparison Base are needed. Therefore the Comparison Base definition is automatically finished prematurely if only two efficient solutions remain in the base. The `COMPARE` command is not available if there is less than two efficient solutions within the Solution Base.

After setting up the Comparison Base you can analyse simultaneously all the selected solutions in terms of the objective values and location decisions. Similarly as during the execution of the `SUMMARY` command, there are available four screens:

- Values,
- A/R Bars,
- U/N Bars,
- Locations.

At the top of each screen a header with the screen name is displayed.

The Values screen (see Fig. 27) provides you with numerical information about the objective values of several efficient solutions within the Comparison Base. There is presented a table where each row corresponds to a single efficient solution and the columns correspond to several objective functions. So, paying attention on a column you can compare the objective values of the specific criterion for given efficient solutions (rows).

The A/R Bars screen (see Fig. 30) provides you with a graphical comparison of the efficient solutions achievements with respect to the given aspiration and reservation levels. Namely, for each objective separately the group of aspiration/reservation bars corresponding to several Comparison Base items is presented. For detail description of a single aspiration/reservation bar structure see Section 5.3.1.

Similarly, the U/N Bars screen (see Fig. 29) provides you with a graphical comparison of the efficient solutions achievements with respect to the utopia and nadir values. Namely, for each objective separately the group of utopia/nadir bars corresponding to several Comparison Base items is presented. For detail description of a single aspiration/reservation bar structure see Section 5.3.1.

At both the bars screens the name of the objective is displayed over the bars group and each bar is marked (on the left) by the solution identification number. Usually, all the objective groups of bars cannot be displayed simultaneously due to lack of place on the screen. You can then scroll the screen using the <UP> or <DOWN> arrows. At the screen is displayed as many objective groups as possible (depending on the Comparison Base size) and using the narrow arrows you scroll one objective group up or down, respectively.

The Locations screen (see Fig. 20 or 28) provides you with comparison of the location decisions on which several efficient solutions are based. There is displayed a special table where each row corresponds to one potential node and the columns correspond to several items of the Comparison Base. So, paying attention on a row you may compare decisions connected with a specific potential node for given efficient solutions (columns).

Similarly as the standard SUMMARY screens, the COMPARE screens are controlled by the <PGDN> or <PGUP> keys which display screen by screen forward or backward, respectively. In order to return control to the Main Menu you have to press the <ESC> key.

As it was mentioned in Section 5.2.4 the aspiration and reservation levels may be defined approximately while analyzing the group of utopia/nadir bars corresponding to several Comparison Base items. We are going to describe shortly this procedure.

At the top of the U/N bars screen you can find an utopia/nadir scale (see Figure 29). Whole the scale represents 100% of the difference = nadir − utopia. Every interval between two neighboring marks on the scale represents about 6% of the difference. This division will help you to select satisfactory points on the scale.

There are two braces at the endpoints of the scale. The left brace represents the aspiration and shows utopia level at the beginning. After pressing <a> you can move the brace along the scale using <LEFT> and <RIGHT> arrows. Move the brace and leave it at a selected position on the scale. The position corresponds to a percentage degradation of the current objective value from the aspiration level. This position will be automatically recalculated onto the corresponding aspiration level and the result will be placed in the aspiration/reservation table.

Similarly you can approximately define a new reservation level. In this purpose press <r> and move the right brace (pointing out the nadir level at the beginning) to a selected position on the scale using control keys.

It appears to be very useful to base selection of braces positions on examination and analysis of the presented bars.

Note that the scale is 50 characters long. Hence, this method allows to define aspiration and reservation levels with accuracy of 2%.

In such a way the aspiration/reservation levels for the objective occupying position at the top of the screen have been defined. To specify aspiration and reservation levels for another objective you have to place the objective bars at the top of the screen using control keys, and repeat the procedure described above.

Troubleshooting:

1. While executing of the COMPARE command the following message appears:

   ```
   There are NO SOLUTIONS !
   SELECT ANOTHER COMMAND !
   ```

   The Solution Base is empty. Press the <ESC> key to return control to the Main Menu and execute another command (e.g., the EFFICIENT command for solution generation).

2. While executing of the COMPARE command the following message appears:
   There is ONLY ONE SOLUTION !
   SELECT ANOTHER COMMAND !
   The Solution Base contains only one solution and the comparison cannot be performed. Press the <ESC> key to return control to the Main Menu and execute another command (e.g., the SUMMARY command for solution examination).

5.4.2 NEXT, PREVIOUS and LAST commands

The commands grouped in the SOLUTION branch can operate only on the Current Solution. The NEXT, PREVIOUS and LAST commands allows you to select any efficient solution from the Solution Base as the Current Solution and thereby to use the SOLUTION branch commands on that solution.

The efficient solution within the Solution Base are ordered according to the increasing identification numbers. The NEXT command causes that the next one after the Current Solution becomes a new Current Solution. While executing the PREVIOUS command the solution that precedes the Current Solution in the Solution Base becomes a new Current Solution. The LAST command makes the last solution in the Solution Base being a new Current Solution. While executing any of the three commands the new Current Solution is presented using the SUMMARY screens (see Section 5.3.1 for details). So, in order to return control to the Main Menu you have to press the <ESC> key.

Troubleshooting:

1. While executing of the NEXT, PREVIOUS or LAST command the following message appears:
   There are NO SOLUTIONS !
   SELECT ANOTHER COMMAND !
   The Solution Base is empty. Press the <ESC> key to return control to the Main Menu and execute another command (e.g., the EFFICIENT command for solution generation).

2. While executing of the NEXT command the following message appears:
   There is NO NEXT SOLUTION !
   SELECT ANOTHER COMMAND !
   The last solution is the Current Solution. Press the <ESC> key to return control to the Main Menu.

3. While executing of the PREVIOUS command the following message appears:
   There is NO PREVIOUS SOLUTION !
   SELECT ANOTHER COMMAND !
   The oldest solution in the Solution Base is the Current Solution. Press the <ESC> key to return control to the Main Menu.

5.4.3 RESTORE command

The RESTORE command allows you to insert an efficient solution from a solution file to the Solution Base. The solution file have to be previously prepared by the SAVE command (during the same or some earlier session with the problem). Thus the RESTORE command together with the SAVE command extend capabilities of the DINAS system making an

opportunity for partitioning the analysis process into a few sessions as well as they allows you to conduct an analysis with greater number of efficient solutions than the Solution Base capacity (more than nine).

During the execution of the RESTORE command you are asked for a name of the solution file to be inserted. For this purpose a special eight–character field is displayed preceded by the message:

GIVE Solution File Name:

You can then edit the name according to the general field editing rules (see Appendix A for details).

You finish the edition by pressing the <ENTER> key whereas pressing the <ESC> key allows you to cancel the RESTORE command. If the system has found the proper solution file then the following message appears:

WAIT Restoring *filename* **as Efficient Solution No #**

and there is a short pause while the solution file is input and checked. If the restored solution file has found as a proper one then the inserted solution becomes the Current Solution and it is presented with the SUMMARY screens (see Section 5.3.1 for details). So, in order to leave the command you have to press the <ESC> key.

To avoid troubles while executing the RESTORE command you should observe the following rules:

- before the session copy all the necessary solution files to the default directory;
- make sure that the solution files correspond to the problem under analysis, i.e., the problem has not be changed after the solution files generation.

Troubleshooting:

1. While executing of the RESTORE command the following message appears:
   Problem NOT AVAILABLE !
   SELECT CONVERT COMMAND !
   There is no problem assigned to the Interactive System. Press the <ESC> key to return control to the Main Menu and execute the CONVERT command.

2. While executing of the RESTORE command the following message appears:
   Utopia & Nadir Vectors are NOT AVAILABLE !
   SELECT PAY-OFF COMMAND !
   The PAY-OFF command has to be executed before first using of the RESTORE command. Press the <ESC> key to return control to the Main Menu and execute the PAY-OFF command.

3. While executing of the RESTORE command the following message appears:
   Some problems with FILE: *name*
   PRESS ANY KEY TO CONTINUE !
   The specified solution file cannot be found in the default directory. Make sure that you have specified a proper file name and it is in the default directory.

4. While executing of the RESTORE command the following message appears:
   Illegal SOLUTION FILE !
   PRESS ANY KEY TO CONTINUE !
   The specified solution file does not contain a solution which corresponds to the

problem under consideration. Simply, it is not a solution file or it is a solution file to another problem. After pressing any key you can edit the name of the solution file to be restored. You can cancel the command execution by pressing the <ESC> key while editing the name.

6 Network Editor

6.1 The Editor Menu

DINAS is armed with the built–in Network Editor EDINET. EDINET is a full–screen editor which allows you to create a network file for the problem using IBM PC's keyboard. Two modes of using the editor are supported: the edit and the view mode. The edit mode allows you to display and edit data of the problem. In the view mode the data can be only examined but cannot be edited. This mode is used for an examination of the current efficient solution. The modes are activated due to selecting of appropriate commands from the Main Menu: PROBLEM for the edit mode and BROWSE for the view mode.

Using the EDINET editor is simple as it can be: when you have selected the PROBLEM or BROWSE command, the editor is activated automatically. The Editor Menu and the corresponding banner appears on the display.

FILE	PRINT NETWORK	EDIT NETWORK
Load		List nodes
Save		Network
MPS		Selections
Quit		Objectives

Table 13: The Editor Menu

Table 13 shows the commands available in the Editor Menu. A command is selected while moving the reverse image pointer through the menu by use of arrow keys, and executed by pressing the <ENTER> key. The HELP file is available at any time by pressing the <F1> function key.

The commands are partitioned into three branches: FILE, PRINT NETWORK and EDIT NETWORK. The FILE branch collects commands associated with file operations and leaving the editor (see Figure 5). The PRINT NETWORK branch consists of a single command which generates various reports. The EDIT NETWORK branch collects the editor commands which enable the creation and modification of the network file (see Figure 6).

Troubleshooting:

1. After pressing the <F1> key the message
   No help file - editor.hlp press any key to continue
   appears in the help window. Make sure that the file EDITOR.HLP is available in the default directory.

6.2 FILE branch

6.2.1 LOAD command

The LOAD command is used to select a network file to be edited, printed and saved. The LOAD command will issue the request for a network file name. A special field for the name is displayed preceded by the short message:

```
File:
```

and you may respond with any legal file name, e.g., **Filename.ext**, editing the name according to the general field editing rules (see Appendix A).

If you enter a file name without period and extension, the extension **NET** is automatically assumed and appended to the name.

If the specified file does not exist or cannot be opened, the following message is displayed:

```
Some problems with input file - PRESS enter
```

After pressing the <ENTER> key a new file is opened and is started as the current network file if it was not opened before.

If the EDINET editor is called by the BROWSE command (in the view mode), the LOAD command is not attainable to you; the current network file processed in the system is automatically loaded and assumed to be current in the editor.

If you work with a network file and you are going to load another file, then you are asked whether the current network file may be overridden.

Remarks: The network file name may be preceded by a path what allows you to edit files from various directories.

The LOAD command is not available while working in the view mode.

Troubleshooting:

1. The message

   Some problems with input file - PRESS enter

   is issued during the execution of the LOAD command since the specified network file cannot be loaded.

   The accident may occur for the following reasons

   - you want to create a new network file;
   - you made a mistake specifying an incorrect file name;
   - the drive is not ready for reading.

   In the first case simply press <ENTER> to open the new network file. In the second case specify the file once again using its correct name, and press <ENTER>. In the third case you have to make the drive ready for reading and press <ENTER>.

2. The message

   Can't read network file ! Press any key.

   is issued during the execution of the LOAD command. It may happen when a fatal disk read error takes place. Use another source disk to input the network file.

6.2.2 SAVE command

The SAVE command is used to save the current network file on a disk. The editing process is performed entirely in memory, and any associated disk file is not affected. Saving of the edited file on a disk is done explicitly with the SAVE command. After selecting the SAVE command you are asked for a name of the network file. For this purpose a special field appears preceded by the message:

```
File:
```

You can then edit the name according to the general field editing rules (see Appendix A for details). The field is initialized with the recently used name of the network file during the LOAD or SAVE command execution. Thus if you want to save the recently loaded file without any change of the name you can simply press <ENTER> to confirm the file name.
Warning: The SAVE command does not produces a BAK file. Therefore you should change the file name during the execution of the SAVE command, if you are interested in saving both the network files: the original one and the modified one.
Warning: During the execution of the PAY-OFF command from the DINAS Main Menu, the computed pay off matrix and the utopia and nadir vectors are automatically saved on the current network file. They can be used in next runs without renewed computation provided that the file has not been saved by the editor SAVE command. Using of the SAVE command destroys the pay–off record on the network file.
Remark: The SAVE command is not available while working in the view mode.

Troubleshooting:

1. The message
   ```
   Some problems with output file - PRESS enter
   ```
   is issued during the execution of the SAVE command since the specified network file cannot be saved. It happens when the drive is not ready for writing. Thus make the drive ready and press <ENTER> to finish the execution of the SAVE command.

2. The message
   ```
   Can't write network file!  Press any key.
   ```
   is issued during the execution of the SAVE command. It happens when either the disk space is exhausted and the network file cannot be saved as whole or a fatal disk write error takes place. To avoid these troubles change the target disk and press <ENTER>.

6.2.3 MPS command

The MPS command is used to generate a MPS file corresponding to the current network file. If the current network is not yet declered end editor is empty, the MPS command generates the MPS file consisting of identyfier lines only. The generated MPS file is saved on a disk explicitly with the MPS command. The form of the MPS file is described in details in Appendix E.

After selecting the MPS command you are asked for a name of the MPS file. For this purpose a special field appears preceded by the message:

```
File:
```

The field is initialized with the name of the current network file. The extension .mps is automatically added to the name. Thus if you want to generate and save the MPS file corresponding to the recently loaded Currfile.net you can simply press <ENTER> to confirm the name Currfile.mps.

Instead of the automatically initialized name you can edit in the field File: a new name of the MPS file according to the general field editing rules (see Appendix A for details).

Warning: The MPS command does not produce a BAK file. Therefore you should change the MPS file name during execution of the MPS command if you are interested in keeping MPS files corresponding to different versions of an orginal network file.

Warning: The MPS command may produce a MPS file in which two columns (or two rows) have identical names. Such an event is not allowed. Therefore before the generated MPS file is used as an input file you must prove whether all columns (rows) names are different and correct the names if it is needed.

Remark: The MPS command is not available while working in the view mode.

6.2.4 QUIT command

The QUIT command is used to leave the editor and return control to the Main Menu. The QUIT command does not save automatically the current network file. In order to save the edited data on a disk you have to use the SAVE command. To prevent you for accidental loss of the data, if a network file has been edited (since it was loaded) you are asked whether you want to save it before leaving the editor. Namely, the following question appears:

```
Abandon current network file ? (y/n)
```

If you have pressed the <Y> key then the control will return to the DINAS Main Menu. Pressing another key allows you to remain within the editor.

6.3 PRINT NETWORK branch

6.3.1 PRINT NETWORK command

The PRINT NETWORK command controls the generation of two kinds of reports: the network report or the solution report. The report kind is strictly connected with the editor mode. If the editor is called in the edit mode (PROBLEM command) then the network report is generated whereas the solution report is generated in the view mode (BROWSE command). Both the reports take the form of a set of tables. After the command activation, the following message is displayed:

```
Press ENTER to select printer or type file name
```

It allows you to specify whether the tables have to be printed directly or stored on a designated file. The report file is a standard text file and thereby it will be able to be displayed (typed) or printed with the standard DOS commands after leaving the DINAS system. The file name have to be complied with DOS requirements.

During printing the report a message appears:

```
Press any key to break printing
```

which inform you about an opportunity to interrupt printing at any time. If you press a key then the report generation is suspended and the following command is displayed:

```
Press ENTER to resume, ESC to stop printing
```

Then, if <ENTER> is pressed, printing will be continued, but if you press the <ESC> key then the command will be canceled and the Editor Menu will be reactivated.

6.3.2 The network report

The network report consists of 5 tables: OBJECTIVES, ARCS, FIXED NODES, POTENTIAL NODES and SELECTIONS. Each table is preceded by a brief header identifying the system (DINAS), problem (file name) and table. The header is directly followed by a pattern of a single record of the table. An example of each table is given below. The contents of the tables are described in the notes that follow each example.

The OBJECTIVES table:

```
DINAS
PROBLEM   demo.net
OBJECTIVE
  NAME       TYPE     ACTIVITY
Invest       MIN        YES
Satisf       MAX        YES
Dist         MIN        YES
Prox         MAX        NO
```

Contents: A list of the objectives with their types and activities.
Name: The name of the objective function.
Type: MIN if the objective function is minimized and MAX if it is maximized.
Activity: YES if the objective is active and NO otherwise.

The ARCS table:

```
DINAS
PROBLEM   demo.net
ARC
NAME          FROM  -  TO          OBJECTIVE
CAPACITY                           NAME      COEFFICIENT
BURO      Bush      -   Robur
50                                 Invest    0
                                   Satisf    0
                                   Dist      0.01825
                                   Prox      5.21
BAR       Bush      -   Arnika
60                                 Invest    0
                                   Satisf    0
                                   Dist      0.01125
                                   Prox     13.72
```

Contents: A list of the network arcs with their capacities and objective coefficients.
Name: The name of the arc.
Capacity: The arc capacity.
From: The name of the predecessor node of the arc.
To: The name of the successor node of the arc.
Objective: A list of the objectives with their coefficients corresponding to the arc.
Name: The name of the objective.
Coefficient: The objective coefficient corresponding to the arc.

The FIXED NODES table:

```
DINAS
PROBLEM   demo.net
FIXED NODES
  NAME        BALANCE          NAME        BALANCE
Acer          -16              Arnika      -20.5
Betula        -13              Bobrek      -22
Tie           240
```

Contents: A list of the fixed nodes and their balances, in alphabetic order.
Name: The name of the node.
Balance: The balance amount for the corresponding node.

The POTENTIAL NODES table:

```
DINAS
PROBLEM   demo.net
POTENTIAL NODES
                              SELECTIONS          OBJECTIVE
NAME          CAPACITY                            NAME     COEFFICIENT

Bush          50              South    Forest     Invest       126
                                                  Satisf       100
                                                  Dist           0
                                                  Prox           0
Fiord         60              North               Invest       220
                                                  Satisf        87
                                                  Dist           0
                                                  Prox           0
```

Contents: A list of the potential nodes with their capacities, selections which contain the node, and objective coefficients.
Name: The name of the node.
Capacity: The node capacity.
Selection: The names of (no more then two) selections including this node.
Objective: A list of the objectives with their coefficients corresponding to the node.
Name: The name of the objective.
Coefficient: The objective coefficient corresponding to the node.

The SELECTIONS table:

```
DINAS
PROBLEM   demo.net
SELECTIONS
  NAME         BOUND          NODES
             LOWER UPPER
North          0     1        Fiord     Ice
South          1     2        Bush      Oasis
```

Contents: The alphabetically ordered list of selections with their lists of included potential nodes and lower and upper bounds on numbers of the nodes to be selected.
Name: The name of the selection.
Lower: The lower bound on number of the nodes to be selected.
Upper: The upper bound on number of the nodes to be selected.
Nodes: A list of the potential nodes which belong to the selection.

6.3.3 The solution report

The solution report differs from the network report only in the tables OBJECTIVES, ARCS and POTENTIAL NODES. In these tables there are some additional entries which represent the current efficient solution. An example of each of the extended tables is given below.

The OBJECTIVES table:

```
DINAS
PROBLEM   demo.net
OBJECTIVE
  NAME       TYPE     ACTIVITY     VALUE
Invest       MIN        YES       386
Satisf       MAX        YES       276
Dist         MIN        YES       2.26056e+00
Prox         MAX        YES       6.45690e+03
```

Additional entries:
Value: The optimal (efficient) value of the corresponding objective function.

The ARCS table:

```
DINAS
PROBLEM   demo.net
ARC
NAME          FROM  -  TO          OBJECTIVE
CAPACITY       FLOW                NAME      COEFFICIENT
BURO      Bush      -   Robur
50             2.05000e+01        Invest    0
                                    Satisf    0
                                    Dist      0.01825
                                    Prox      5.21
BAR       Bush      -   Arnika
60             0                   Invest    0
                                    Satisf    0
                                    Dist      0.01125
                                    Prox     13.72
```

Additional entries:
Flow: The optimal (efficient) flow along the arc.

The POTENTIAL NODES table:

```
DINAS
PROBLEM   demo.net
POTENTIAL NODES
ACTIVITY    FLOW          SELECTIONS          OBJECTIVE
NAME        CAPACITY                          NAME     COEFFICIENT
YES         47.86
Bush        50          South    Forest       Invest       126
                                              Satisf       100
                                              Dist           0
                                              Prox           0
NO           0
Fiord       60          North                 Invest       220
                                              Satisf        87
                                              Dist           0
                                              Prox           0
```

Additional entries:
Activity: YES if the node has been located and NO otherwise.
Flow: The optimal (efficient) flow through the node.

6.4 EDIT NETWORK branch

The **DINAS** interactive procedure works with the network file containing whole the information defining the problem and the **EDIT NETWORK** branch of the Editor Menu enables to prepare this file. The main data of the problem can be divided into two groups:

- logical data defining the structure of a transportation network (nodes, arcs, selections);

- numerical data describing the nodes and arcs of the network (balances, capacities, objective coefficients).

The general concept of the Network Editor is to edit the data while defining the logical structure of the network. More precisely, the essence of the EDINET concept is a dynamic movement from some current node to its neighboring nodes, and vice versa, according to the network structure. The input data are inserted by a special mechanism of windows, while visiting several nodes. At any time only one of the windows representing different kinds of the data is active. The corresponding part of the data can be then inserted. While working with the editor you can activate several windows.

The EDIT NETWORK branch consists of he following commands: LIST NODES, NETWORK, SELECTIONS and OBJECTIVES. Each of these commands is executed explicitly by selecting the appropriate position in the EDIT NETWORK submenu and pressing the <ENTER> key. However, in order to prevent you from frequent returns to the menu during the editing process, a special implicit execution technique is also available. Namely, each of these commands can be activated during execution of another command by pressing the <ALT> key together with the first letter of the command name, i.e., by pressing <ALT>/<L>, <ALT>/<N>, <ALT>/<S> or <ALT>/<O>, respectively.

6.4.1 LIST NODES command

After activating the LIST NODES command a special window appears. An example of such a window is presented in Figure 16.

The message at the top of the window provides you with the window name and the name of the current network file. While starting with a new network file the window is empty. If not, the window includes all the network nodes (in the alphabetic order), which has been inserted so far. At any time a certain node of the nodes list is pointed out. The names of the potential nodes are highlighted (blue) and the pointed out node is blinking. Using arrow keys you can move the pointer along the list to select a node. The list is vertically scrolled, if it is too long to be presented directly.

On the right hand side of the LIST NODES window, the window with data corresponding to the pointed out node is presented. The data are exchanged while pointing out another node. The node window can be active or not. If the window is not active you can only examine the data inside the window. To activate the pointed out node window press <TAB>. Then you can modify the data according to the editing procedure described in Section 6.5.1. To return to the LIST NODES command level press <ESC>.

There are two ways to finish the LIST NODES command. By pressing the <ESC> key you simply leave the command and return control to the Editor Menu (in case of a direct execution) or to the place where the command was executed (by the <ALT>/<L> key). You can also finish the command by activating the EDIT windows (see Section 6.5) with a selected node. For this purpose you press <ENTER> what confirms the node selection given by the pointer or simply you type a name of the node (new or existing).

6.4.2 NETWORK command

The NETWORK command issues a special window which helps you to recognize the network structure. An example of the NETWORK window is presented in Figure 15. The window name and the name of the network file are given at the top of the window.

The NETWORK window presents a graphic scheme of the network that is actually edited. The basic concept of the scheme is to give a local view on the structure of the

network. The scheme consists of several (alphabetically ordered) central nodes connected to their predecessors (on the left hand side) and their successors (on the right hand side). As a rule, the pointer points out the central node at the top of the window. However, you can change the pointed out central node. After pressing the <DOWN> arrow, the next (in alphabetic order) central node is pointed out; the central nodes with their predecessors and successors are scrolled up and, in result, the next central node will occupy the first position. Similarly, to point out the preceding central node you have to press the <UP> arrow, then the central nodes are scrolled down and the preceding central node is shifted to the pointed out position.

If a central node is pointed out, the window containing data corresponding to the node are presented on the right hand side of the screen. The data are exchanged while pointing out another central node. The node window can be active or not. If the window is not active you can only examine the data inside the window. To activate the pointed out node window press <TAB>. Then you can modify the data according to the editing procedure described in Section 6.5.1. To return to the NETWORK command level press <ESC>.

You can also move the pointer to a predecessor or successor of the pointed out central node. To point out a predecessor of the central node you have to use the <LEFT> arrow. Then the first predecessor is pointed out and using the <DOWN> or <UP> arrows you can select another predecessor. Use the <RIGHT> arrow to return to the pointed out central node. You can also change the selected predecessor into the pointed out central node using the <ENTER> key. Similarly, you can point out a successor of the central node by pressing the <RIGHT> arrow. Then you can select any successor using the <DOWN> or <UP> arrows. Use the <LEFT> arrow to return to the pointed out central node or press <ENTER> to change the selected successor into the pointed out central node.

If a predecessor (or successor) to the central pointed out node is selected the window containing data corresponding to the arc SelectedPredecessor-CentralNode (or CentralNode-SelectedSuccessor) is displayed on the right hand side of the screen. The data are exchanged while selecting another predecessor (or successor); in fact, while selecting another arc. The arc window can be active or not. If the window is not active you can only examine the data inside of the window. To activate the window press the <TAB> key. Then you can modify the data according to the editing procedure presented in Section 6.5.4. To return to the predecessors (or successors) level press <ESC>.

Similarly as in the LIST NODES command here are two ways to finish the NETWORK command. By pressing the <ESC> key you simply leave the command and return control to the Editor Menu (in case of the direct execution) or to the place where the command was executed (by the <ALT>/<N> key). You can also finish the command by activating the EDIT windows (see Section 6.5) with a selected node. For this purpose you press <ENTER> what confirms the node selection given by the pointer (the pointed out central node).

Remark: Move the pointer to the central position before leaving the command (by pressing the <ENTER> key).

6.4.3 SELECTIONS command

After executing of the SELECTIONS command a special window is displayed. An example of the SELECTIONS window is given in Figure 17.

The selections which have been met during the editing process are listed in this window. On the right hand side of each selection name a list of the potential nodes belonging to the selection is given. The top position of the selections list is pointed out. You can move

each selection into the pointed out position by scrolling the list with the <DOWN> or <UP> arrows.

Sometimes the selections list is found empty (e.g., if the current network file has been just opened or if no selections have been inserted until now). Then the following massage is issued:

```
No selections - press a key
```

After pressing any key you will leave the command.

If you press <ENTER>, a special window for edit bounds of the pointed out selection appears. An example of the BOUNDS window is given in Figure 18.

The name of the pointed out selection is displayed at the top of the window. One of two fields corresponding to the LOWER and UPPER bounds is highlighted while editing. You can insert an integer number into the highlighted field and press the <ENTER> key or <DOWN> or <UP> arrows. Then the other field will be highlighted and you can edit the second bound. In order to leave the BOUNDS window you have to press the <ESC> key. Then, the SELECTIONS window appears again.

If the bounds have been inserted and the UPPER bound proves to be less than the LOWER bound then the following error message is displayed:

```
Can't be LOWER > UPPER  -  press any key
```

After pressing a key you have to edit bounds again to avoid this error. The editor also protests if you are going to insert a negative value as a bound.

Pressing the <ESC> key lets you leave the SELECTIONS command and return to the point where it was called.

6.4.4 OBJECTIVES command

Objective functions are defined by use of the OBJECTIVES command. Activating this command will issue the OBJECTIVES window. An illustration of such a window is given in Figure 7. While starting with a new file, the window is blank apart from the pattern line at the top. In order to define an objective you have to fill in the window form three items:

OBJECTIVE: The objective name.

TYPE: MIN if the objective is minimized and MAX if it is maximized.

ACTIVITY: YES if the objective is active and NO otherwise.

If you actually edit the OBJECTIVE or TYPE field, you can move the cursor to the next field (TYPE or ACTIVITY, respectively) using the <ENTER> key. If you press <ENTER> after the ACTIVITY is selected, the cursor moves to the next objective. Use <DOWN> or <UP> arrow to move the cursor along the objective list. To edit TYPE and ACTIVITY you have to select one of two options given in the edited field using the <LEFT> or <RIGHT> arrows: MIN or MAX and YES or NO, respectively.

If the editor works in the view mode the contents of the OBJECTIVES window is extended. Namely, on the right hand side of each objective record the optimal (efficient) value of the objective is given.

In order to leave the OBJECTIVES command you have to press the <ESC> key. Then the control will return to the point where the command was called.

6.5 EDIT windows

6.5.1 NODE windows

In order to start the network edit you have to call the LIST NODES or NETWORK command and leave it by activating the EDIT windows. Thus you have to press <ENTER> or type a node name (in case of the LIST NODES command).

Suppose at first that you have typed a node name. After the first character of the name is entered, the window presented in Figure 8 appears. The character is placed in the name field and you can continue the name edit. When the name is completed press <ENTER> or <ESC>.

Now you meet one of two options of the program performance: the name is new or the name has been inserted before. The new node name case is described below. The second option will be explained in Section 6.5.2.

While performing the first option, the window as in Figure 9 is displayed. You are informed that the node is not defined yet and you have to decide whether the node have to be fixed or potential. Select option "fixed" or "potential" moving cursor with the <LEFT> or <RIGHT> arrows and press <ENTER> or <ESC>. Then the new node is defined and dependently on the selected option the FIXED or POTENTIAL node window appears.

If the FIXED node option has been selected, the window illustrated in Figure 12 is presented on the screen. The inserted node name is found at the top. The message in the second line inform you that the node is fixed. The third line contains the balance field. Using the <DOWN> or <UP> arrows you can select the "name" or "balance" field and edit it.

If you have selected the potential node option, the window that is shown in Figure 10a is displayed. The entered node name is placed at the top. The "potential" type declaration is found in the second line. The third line contains the capacity field. You can edit the capacity and press <ENTER>. Then the first selection field appears. You can edit a selection name and press <ENTER> or only press <ENTER> leaving the field blank (or not changed, in general). Then the second selection field appears and you can do the same. Finally, the block of the objectives is displayed. You can edit several objective coefficients using the <DOWN> and <UP> arrows to change the field to be edited. An example of the complete potential node window is given in Figure 10b. You can use the <DOWN> or <UP> arrow to select any field and edit it again in order to change or correct the data.

Remark: Notice, that you can change the previously edited name. If you change a name into one of another node names, the message will be issued

Already exists such a node - Press any key

since all the node names have to be different. Thus you can change the previously edited name only into a name which is actually not used.

If the editor works in the view mode then the information contained inside of the potential node window is extended. At the left–top corner of the window a special character appears if the potential node occupying this window has been selected (located). Moreover, under the capacity line the optimal value of the flow through the node is given.

If you have finished the editing process of the current (fixed or potential) node then you have to press the <ESC> key in order to activate the CURRENT/FROM/TO screen.

6.5.2 CURRENT/FROM/TO screen

The CURRENT/FROM/TO screen (see Figure 11) consists of three windows: CURRENT NODE in the middle of the screen, NODE FROM on the left hand side and

NODE TO on the right hand side. This screen is displayed when a new node has been defined. It appears also if you have left the **LIST NODES** or **NETWORK** command pressing **<ENTER>** or typing a node name inserted before. The pointed out (in the **<ENTER>** case) or called (by the name) node is assumed to be current and occupies the CURRENT NODE window. Thus the CURRENT NODE window is simply the FIXED or POTENTIAL node window (see Figure 12 and 10b). The window represents a defined node, called current.

The NODE FROM window contains names of the nodes that precede the current node in the network, and names of the corresponding arcs (if they were introduced). Similarly, the NODE TO window contains names of the nodes and arcs which directly succeed the current node in the network. These windows may be empty if no predecessor or no successor to the current node have been inserted. Inside of each of the windows you can move the cursor along the node list using the **<DOWN>** or **<UP>** arrow. If a list is too long it is vertically scrolled.

You can activate any of these three windows using appropriate keys. If the CURRENT NODE window is active then you can activate the NODE FROM or NODE TO window using the **<CTRL>/<LEFT>** or **<CTRL>/<RIGHT>** arrow, respectively. On the other hand, if one of the side windows is active, you can activate the CURRENT NODE window pressing **<ESC>**.

If you want to make a node belonging to the NODE FROM or NODE TO list to be current then, at first, you have to activate the corresponding window. Next, using the **<DOWN>** or **<UP>** arrows you have to select the node and press **<ALT>/<C>**. In result, the selected node becomes current and is automatically inserted into the CURRENT NODE window. Moreover, its predecessors and successors enter the NODE FROM and NODE TO windows, respectively.

In order to return to the Editor Menu you have to activate the CURRENT NODE window and then press the **<ESC>** key.

6.5.3 Predecessors and successors editing

In order to edit predecessors or successors to the current node you have to call the CURRENT/FROM/TO screen and then activate one of the side windows (see Section 6.5.2). Suppose, the NODE FROM window has been activated. Then you can select one of the predecessors to the current node or type a node name (in particular, a new name). Thus the framework described in Section 6.5.1 comes to light again. If you select or type the name of an existing predecessor, the window containing the whole information corresponding to this node will be issued. But if you type a new predecessor name, the definition of the node will be processed. As a result of the both cases a (fixed or potential) NODE window corresponding to this predecessor will be displayed and you will be able to edit or examine the data in the window. After the edit (or examination) is finished you have to press the **<ESC>** key. Then you are asked to respond to the short question (see Figure 12)

ARC?

which appears at the bottom of the window. If you wish to visit the corresponding ARC window you should press the **<ENTER>** key. By pressing the **<ESC>** key you cause reactivating of the NODE FROM list of predecessors.

You can also visit the ARC window immediately while visiting the NODE FROM list. To attain it you have to select a predecessor from the list and then press **<CTRL>/<RIGHT>**

arrow. Then you can see the ARC window with contents corresponding to the arc leading from the selected predecessor to the current node. If you press <CTRL>/<RIGHT> arrow again, the NODE FROM list will be reactivated.

Any successor with their associated arc can be edited similarly, but by use of the NODE TO windows.

Troubleshooting

1. You cannot introduce a predecessor (or successor) name since the message
   Can't define a cyclic arc PRESS any key
   is issued while introducing.
   The network cannot contain any simple loop. To introduce this predecessor (or successor) you must either change its name or change the current node name so as these names will be different.
   If you really want to introduce the loop Nnode–Mnode you may define an artificial fixed node Nloop with the zero balance and the pair of the arcs: Nnode–Nloop, Nloop–Nnode. The loop is equivalent to the pair. The data corresponding to the loop characterize one of the arcs, but the other arc is characterized by the unbounded (sufficiently large) capacity and the vanishing objective coefficients.

2. You cannot introduce a predecessor (or successor) name since the message
   Can't define arc from potential to potential PRESS any key
   is issued while introducing.
   The network cannot contain any arc which joins two potential nodes directly.
   To introduce the arc Npot–Mpot (joining two potential nodes) you have to define an intermediate fixed node Nint with the zero balance and these arcs: Npot–Nint and Nint–Mpot. The data corresponding to the arc Npot–Mpot characterize one of these arc but the other arc is characterized by the unbounded (sufficiently large) capacity and the vanishing objective coefficients. The couple of the arcs defined this way is equivalent to the arc Npot–Mpot.

6.5.4 ARC window

As it was mentioned in Section 6.5.3 the ARC window can be activated while visiting NODE FROM/TO lists or while editing the actual predecessor (using <ENTER>). The window is designed to edit or examine the data corresponding to the arc that leads from the predecessor to the current node (see Figure 13).

The top line at the ARC window contains the predecessor and current node names and this line cannot be edited. The next line is assigned to the name of the arc. The name is optional and need not to be inserted. The arc capacity can be entered in the third line. The next two lines are free. If the editor works in the view mode then you can find the value of the optimal flow along the arc in the first of the free lines. At the end, the block for objective coefficients editing appears. You can edit several coefficients as well as the arc name and capacity using the <DOWN> or <UP> arrow and the <ENTER> key to select the field to be edited.

In order to close the ARC window you have to press the <ESC> key. You can also return directly to a NODE FROM/TO list by pressing <CTRL>/<RIGHT> arrow.

6.6 DELETE command

The editor allows you to delete a node, arc, selection or objective if they are superfluous or unwanted. The DELETE command is executed by activating an appropriate window, pointing out the name of the removed object and pressing <ALT>/<D>. A brief description of these operations is given below.

- Delete an objective:

  1. activate the OBJECTIVES window;
  2. point out the name of the objective to be deleted;
  3. press <ALT>/<D>.

  Warning: The whole information concerning the deleted objective, including the coefficients defined within each POTENTIAL NODE or ARC window, is lost. Therefore for temporal deactivation of the objective function you should use the ACTIVITY column in the OBJECTIVES window instead of the DELETE command.

- Delete a selection:

  1. activate the SELECTIONS window;
  2. point out the name of the selection to be deleted;
  3. press <ALT>/<D>.

  Remark: The name of the selection is also deleted from each POTENTIAL NODE window associated with the deleted selection.

- Delete a node:

  1. load the node to be deleted into the CURRENT NODE window and activate this window;
  2. point out the name of the node;
  3. press <ALT>/<D>.

  Remark: All the arcs that are incident to the deleted node are deleted with this command automatically.

- Delete an arc:

  1. load one of the arc endpoints into the CURRENT NODE window;
  2. the remainder endpoint is either predecessor or successor to the current node; activate the appropriate NODE FROM or NODE TO window and select this endpoint from the node list included inside of the window;
  3. press <ALT>/<D>.

  After pressing <ALT>/<D> you are asked whether you really want to delete the indicated object (objective, selection, node or arc). Namely, you have to answer a question like this:

  Do you really....

  You press then <Y> to confirm or <N> otherwise.
  Warning: No provision exists to restore a deleted object, so be careful while using the DELETE command.

7 References

Abernathy, W. J., Hershey, J. C. (1972). A spatial-allocation model for regional health-services planning. Oper. Res. 20, pp. 629–642.

Glover, F., Klingman, D. (1981). The simplex SON method for LP/embedded network problems. *Mathematical Programming Study* 15, pp. 148–176.

Glover, F., Klingman, D. (1985). Basis exchange characterization for the simplex SON algorithm for LP/embedded networks. *Mathematical Programming Study* 24, pp. 141–157.

Grauer, M., Lewandowski, A., Wierzbicki, A. (1984). DIDASS — theory, implementation and experiences. In M. Grauer, A.P. Wierzbicki (eds), Interactive Decision Analysis. Springer, Berlin 1984.

Ogryczak, W., Studzinski, K., Zorychta, K. (1987). A solver for the transshipment problem with facility location. In A. Lewandowski and A. Wierzbicki (Eds.), Theory, Software and Testing Examples for Decision Support Systems. IIASA, Laxenburg.

Ogryczak, W., Studzinski, K., Zorychta, K. (1988). A generalized reference point approach to multiobjective transshipment problem with facility location. IIASA, Laxenburg.

Orchard-Hays, W. (1968). Advanced Linear-Programming Techniques. McGraw-Hill, New York.

Schrage, L. (1975). Implicit representation of variable upper bounds in linear programming. *Mathematical Programming Study* 4, pp. 118–132.

Todd, M. J. (1982). An implementation of the simplex method for linear programming problems with variable upper bounds. *Mathematical Programming Study* 23, pp. 34–49.

Wierzbicki, A. P. (1982). A mathematical basis for satisficing decision making. *Math. Modelling* 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.

A Field editing

While working with DINAS you are frequently asked for some parameters specification. Especially, during the problem input with the network editor you have to give many names and numbers. In the whole DINAS system, i.e. in both the interactive system and the network editor the process of parameters input runs according to the same scheme. Namely, a special field or a group of fields organized in a special window is displayed and you can input parameters by editing these fields.

The size of the field depends on a specific parameter and it limits the maximal length of a name or number. Moreover depending on a specific parameter some characters (some keys) are non accepted as quite improper. For instance, while editing a number there is no possibility to type a letter expect of e or E (which can precede a decimal exponent). The <SPACE> bar is not accepted neither in numbers nor in names.

The process of a single field editing is controlled by a one–character cursor with the following keys: <LEFT> and <RIGHT> arrows, <BACKSPACE>, <DEL> and <INS>. The <LEFT> or <RIGHT> arrow moves the cursor one space left or right, respectively. The <DEL> key causes that the pointed character disappears, and the rest of the field moves to the left one space. Similarly, while using the <BACKSPACE> key the character preceding the cursor disappears, and the rest of the field moves to the left one space. There are available two editing modes: the insert mode and the overwrite mode. While editing in the overwrite mode the input character appears exactly at the cursor position replacing any previous character at this position. While editing in the insert mode the input character also appears exactly at the cursor position but the rest of the field (including the previous character from the cursor position) moves to the right one space. If there is no possibility to move the rest of the field to the right (since the field is full) the input is not accepted. Each field is opened with initialized the insert mode. The <INS> key is a toggle switch that allows you to change the editing mode.

There are a few ways to finish editing and close the field. By pressing the <ENTER> key you close the field and open the next field in case of a multiple fields window (see, e.g., aspiration/reservation edit in Section 5.2.4 or node definition in Section 6.6.1) or you cause the command execution in case of a single field (see e.g. RESTORE command in Section 5.4.3 or LOAD command in Section 6.2.1). By pressing the <ESC> key you close not only the edited field but also the whole window of the fields (see, e.g., the ARC window in Section 6.6.4 or aspiration/reservation edit in Section 5.2.4). Moreover the command or the phase of the command connected with the window is also finished. This possibility can be used, sometimes, to cancel the command (see e.g. RESTORE command in Section 5.4.3). While editing some window of fields you can also finish editing of a single field by pressing a specific movement key which closes the field and open another field in the window. The movement keys are always specified in the help line. Usually, the <UP> and <DOWN> arrows are movements keys, and sometimes the <CTRL>/<LEFT> and <CTRL>/<RIGHT> arrows are also movement keys.

In case of numbers editing, after closing the field the number is automatically verified. Any improper string is signaled by the message

```
error
```

which appears in the field. Moreover, while editing aspiration and reservation levels after closing the field, the number is presented in such a form as it has been taken by the system.

Troubleshooting:

1. You cannot input any character. Pressing any character key makes only the error noise.
   You work in the insert mode with a full field. Use the <DEL> or <BACKSPACE> key to make some space or activate the overwrite mode by the <INS> toggle key.

B Short reference

B.1 The Interactive System

- MENU BRANCHES

PROCESS — commands connected with processing of the multiobjective transshipment–location problem and generation of several efficient solutions (PROBLEM, CONVERT, PAY–OFF, EFFICIENT, QUIT).

SOLUTION — commands connected with the Current Solution (SUMMARY, BROWSE, SAVE, DELETE).

ANALYSIS — commands connected with operations on the Solution Base which collects up to nine efficient solutions (COMPARE, PREVIOUS, NEXT, LAST, RESTORE).

- PROCESS COMMANDS

PROBLEM — edit or input of a problem with the Network Editor. If the network file for the problem has been prepared before DINAS starts, PROCESS can be initialized directly using the CONVERT command.

CONVERT — convert the network file with error checking.

PAY–OFF — compute the pay–off matrix and the utopia and nadir vectors.

EFFICIENT — compute an efficient solution depending on introduced aspiration and reservation levels:

1. edit aspiration and reservation levels;
2. DINAS computes a new efficient solution depending on the aspiration and reservation levels;
3. the solution is put into the Solution Base and is presented as the Current Solution with special tables and bars.

The Current Solution is presented similarly as while using the SUMMARY command.

QUIT — leave the DINAS system.

- SOLUTION COMMANDS

SUMMARY — present short characteristics of the Current Solution such as:

table of selected locations,
table of objective values,
bars in the aspiration/reservation scale,
bars in the utopia/nadir scale.

The tables characterize the current and previous efficient solutions. The bars show a percentage level of each objective value with respect to the corresponding scale.
Change the kind of characteristic using the <PGDN> or <PGUP> key.

BROWSE — examine or print the Current Solution with the Network Editor.

SAVE — save out the Current Solution on a separate file for using in next runs.

DELETE — delete the Current Solution from the Solution Base.

- ANALYSIS COMMANDS

COMPARE — perform comparison of all the efficient solutions from the Solution Base or of some subset of them. (A solution can be temporarily deleted from the comparison using the <DEL> key.)
The following characteristics are used for the comparison:

table of selected locations,
table of objective values,
bars in the aspiration/reservation scale,
bars in the utopia/nadir scale.

The tables and bars characterize all the efficient solutions selected from the Solution Base. The bars are given for each objective and show a percentage level of the objective value with respect to the corresponding scale. Use the <PGDN> or <PGUP> key to change the kind of the current characteristic. Scroll bars for different objectives using the <UP> or <DOWN> arrows.

PREVIOUS — take the previous efficient solution as the Current Solution.

NEXT — take the next efficient solution as the Current Solution.

LAST — take the last efficient solution as the Current Solution.

RESTORE — restore some efficient solution (saved earlier with the SAVE command) to the Solution Base.

B.2 The Network Editor

- MENU COMMANDS

LOAD — select a file to be used to edit.
If the file name is typed without any extension the extension .NET is automatically appended to the name.

SAVE — write the edited file out without creating a .BAK file.

MPS — generate the MPS file and write it without creating a .BAK file.

QUIT — leave the Network Editor and return to the Main Menu.

PRINT NETWORK — print the entire data file and the solution included in the file.

LIST NODES — display a list of nodes (may be empty at the start).
Type or select a node name to activate the edit mode. Then the CURRENT NODE window including the selected (or typed) node is activated.

NETWORK — display a scheme of the transportation network. Select a node to activate the edit mode. Then the CURRENT NODE window is activated with the selected node as the current node.

SELECTIONS	list the nodes belonging to several selections. Use <ENTER> to display the BOUNDS window corresponding to the pointed out selection.
OBJECTIVES	define, modify or examine objectives.

- WINDOWS

CURRENT NODE	the node selected (or typed) with the LIST NODES or NETWORK command becomes the current node. Then it can be defined and its data can be edited or examined.
NODE FROM (list of predecessors)	list names of nodes and the corresponding arcs which precede the current node. Select a node or type a node name.
NODE FROM (a selected predecessor)	edit or examine data of the selected (or typed) node. If the <ESC> key is used after the edit is finished, the question ARC? appears. Press <Y> if an inspection of the corresponding arc is needed, or <N> otherwise.
NODE TO (list of successors)	list names of nodes and the corresponding arcs which succeed the current node. Select a node or type a node name.
NODE TO (a selected successor)	edit or examine data of the selected (or typed) node. If the <ESC> key is used after the edit is finished, the question ARC? appears. Press <Y> if an inspection of the corresponding arc is needed, or <N> otherwise.
BOUNDS	edit or examine lower and upper bounds on a number of potential nodes which can be used in the given selection.

- ALT COMMANDS

ALT C change the actual node from/to into the current node.

ALT D delete an objective, selection, node or arc using the OBJECTIVES, SELECTIONS, CURRENT NODE or NODE FROM/TO command, respectively.

ALT L execute the LIST NODES command.

ALT N execute the NETWORK command.

ALT O execute the OBJECTIVES command.

ALT S execute the SELECTIONS command.

- CTRL COMMANDS

CTRL LEFT ARROW move from CURRENT NODE to NODE FROM.

CTRL RIGHT ARROW move from CURRENT NODE to NODE TO.

CTRL RIGHT ARROW display the ARC window for a node selected from the NODE FROM/TO list, and reversely.

- USING ANOTHER KEYS

ESC	leave the current command/window, e.g., from NODE FROM/TO to CURRENT NODE, from CURRENT NODE to the Editor Menu, and out of HELP.
ENTER	command execution or node selection.
ARROWS	move the pointer or scroll data.

C Installation

DINAS runs under Disk Operating System (DOS version 3.00 or higher) on an IBM-PC XT/AT or compatible and requires 640 K RAM. Using of a color display with EGA or CGA card is recommended but DINAS can also work with a monochrome display and can use Hercules card. A hard disk is recommended but not necessary. One double–sided double–density floppy disk drive is sufficient to run DINAS.

The system can be installed in two versions: with taking advantages of the Numeric Data Processor chip, or without using the NDP chip. However, for solving large real–life problems the version with the NDP chip is strongly recommended.

The DINAS system can cooperate with a mouse. In that case the mouse should be installed (see documentation of the given mouse).

A printer is useful but not necessary since all the reports can be routed directly to a printer or to a disk file for printing at a later time.

The DINAS system is distributed on one double–sided double density diskette. The distribution diskette contains the following files:

README	introductory information
DINAS.EXE	main executable program of the system
WIND.EXE	editor program
SOLVER.N87	optimizer without using NDP
SOLVER.Y87	optimizer with using NDP
DINAS.HLP	DINAS help file
EDITOR.HLP	editor help file
DEMO.NET	tutorial example

Before launching the system you have to make a working copy of the distribution diskette and then put the original one away. You can also install the system on your hard disk.

In order to install the system you have to perform the following operations:

1. make the destination disk (and/or directory) to be default;
2. copy both the EXE files from the distribution diskette;
3. copy both the HLP files from the distribution diskette;
4. copy the DEMO.NET file from the distribution diskette;
5. depending on your hardware configuration copy the SOLVER.N87 or SOLVER.Y87 file and rename it as SOLVER.EXE.

DINAS is not copy protected and thereby all the files from the distribution diskette can be copied using the DOS command COPY. While working without a hard disk, it is usually a good idea to install DINAS on a diskette which has been previously formatted as the so–called system diskette and thereby holds the COMMAND.COM file (see IBM Disk Operating System manual for details).

After finishing of the installation process the following files have to appear in the default directory:

DINAS.EXE main executable program of the system

WIND.EXE editor program

`SOLVER.EXE` optimizer
`DINAS.HLP` `DINAS` help file
`EDITOR.HLP` editor help file
`DEMO.NET` tutorial example

Warning: The working diskette cannot be write protected.

D System files

The **DINAS** system consists of five system files:

DINAS.EXE
SOLVER.EXE
WIND.EXE
DINAS.HLP
EDITOR.HLP

- **DINAS.EXE** loads and runs all the system programs.
- **SOLVER.EXE** contains the Interactive System.
- **WIND.EXE** contains the Network Editor.
- **DINAS.HLP** and **EDITOR.HLP** are help files to the Interactive System and Network Editor, respectively.

All the other files are created by **DINAS** as various data files connected with performed analysis.

The network files contain the problems data in a special internal format. They are created by the Network Editor while saving the edited problems. The network file has, usually, the extension **.NET**. Namely, if the problem is specified without any extension then the extension **.NET** is automatically assumed. However, you can specify the problem name with another extension. Moreover you can route the network file to another directory or drive using the corresponding path in the name.

The solution files contain data for the several efficient solutions. They are created by the Interactive System while executing of the **SAVE** command. The solution files have the extension **.EFF** and they are always created in the default directory.

The output files contain network or solution reports. They are created by the Network Editor while executing of the **PRINT NETWORK** command. The reports are, usually, routed directly to the printer. However, you have an opportunity to specify an output file instead of the printer. The output file has the extension **.PRN** provided that you had not specified another one. The name of the output file can be preceded by a path in order to create the file in the corresponding directory.

Remark: We suggest you to depend on the default extensions (i.e., to specify all the names without extensions) in order to avoid mistakes.

DINAS creates also some working files and erases them before finishing the run. Thus you should never see these files in the directory. However, sometimes due to an extraordinary finish of the run such a file can appear in the directory. The working files get unique names of the type **XTEMP$$$.$$$**. Do not be afraid to delete such a file whenever you find it.

The system files take up 160 K of a disk space leaving about 200 K on a standard double–sided double–density disk. However, you should not treat the working diskette as archives for all the old problems and solutions. Files which are not used in the current analysis should be copied to another disk and deleted from the working one.

E Description of the MPS file

The MPS file is a standard input file for professional mathematical programming solvers, e.g., for the MPSX package. It is a text file containing all the data associated with the problem to be analyzed, as they are presented in Section 3.1. As it was mentioned in this section potential nodes have been transformed into artificial arcs using a duplication node technique. Denote by y_k ($k \in N_p$) the decision variable equal to 1 for selected potential node and 0 otherwise. (N_p denotes the set of potential nodes.) Then the mathematical model of our problem before the problem has been transformed can be formulated as follows

$$
\begin{aligned}
\textit{minimize} \quad & \sum_{(i,j)\in A} f_{ij}^p x_{ij} + \sum_{k\in N_p} f_k^p y_k, && p = 1,2,\ldots,n_0 \\
\textit{subject to} \quad & \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} = b_i, && i \in N \\
& \sum_{(i,j)\in A} x_{ij} \le c_i y_i, && i \in N_p \\
& g_k \le \sum_{p\in S_k} y_p \le h_k, && k = 1,2,\ldots,n_s \\
& 0 \le x_{ij} \le c_{ij}, && (i,j) \in A \\
& 0 \le y_k \le 1 && k \in N_p \\
& y_k \text{ are integer} && k \in N_p
\end{aligned}
$$

Of course, f_k^p and c_i denote objective coefficients and capacities for potential nodes, respectivelly. As we have assumed $b_i = 0$ for potential nodes.

This problem is equivalent to the problem (1)–(6), but it is simplier since its dimension is less. The `MPS` command of `DINAS` generates a MPS file just for the problem formulated above. The file consists of a number of lines containing data of the problem. The lines are of two kinds:

- identifier lines which open different sections of the data,
- ordinary lines (or simply lines) in which the data are placed.

The data are divided into five sections: ROWS, COLUMNS, RHS, RANGES and BOUNDS.

Inside of an ordinary line data are placed in several fields. The fields occupy following positions:

Field 1 – columns 2–3,
Field 2 – columns 5–12,
Field 3 – columns 15–22,
Field 4 – columns 25–36,
Field 5 – columns 40–47,
Field 6 – columns 50–61,

The following identyfiers are used to form identyfier lines: NAME, ROWS, COLUMNS, RHS, RANGES, BOUNDS, ENDATA. Each identyfier is written to its line without spaces, begining from the 1'st column. Besides, inside of the NAME line (Field 3) a name of the MPS file is included (extension is omitted).

In general, the MPS file generated with the `MPS` command has the form:

NAME line
ROWS line
    lines containing row names and corresponding relations
COLUMNS line
    lines containing data
RHS line
    lines containing balances and upper bounds for selections
RANGES line
    lines containing selection ranges
BOUNDS line
    lines containing arc capacities and upper bounds for integer variables
ENDATA line

Now, we are going to briefly describe successive sections of the lines.

Row lines

There are four types of the lines:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
N	Objective				
E	Fixed node				
L	BPotential node				
L	Selection				

Codes N, E, L correspond to these relations respectivelly: non-restriction, equality, inequality (less or equal).

Column lines

There are two types of columns (variables) in our problem:

- arc columns corresponding to the flow variables x_{ij},
- potential node columns corresponding to the integer variables y_k.

Each column of the first type is represented by this sequence of lines:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	Arc	Objective	Coefficient		
	...	...	...		
	Arc	Node from	+1	Node to	−1
	Arc	Node from	+1		

The last line is included into the sequence only if the Node from is potential one.

The set of all lines corresponding to columns of the second type is bracketed with the lines:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	INPOTB	'MARKER'		'INTORG'	
	INPOTB	'MARKER'		'INTEND'	

A single column of the second type is represented by these lines:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	XPotential node	Objective	Coefficient		
	...	...	...		
	XPotential node	BPotential node	Coefficient		
	XPotential node	Selection1	+1	Selection2	+1

The last line in the sequence either may be omitted (if the Potential node is not included into any selection) or may have one of the form:

XPotential node	Selection1	+1		
XPotential node	Selection1	+1	Selection2	+1

(depending on the node belongs to only one or to two selections).

RHS lines

The RHS lines are of two types:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	RHS	Fixed node	balance		
	RHS	Selection	1		

Range lines

Typical form of the range line is following:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
	RANGE	Selection	*range*		

where *range* denotes the difference $h_i - g_i$ for the given selection.

Bound lines

There are two types of the bound lines:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
UP	BOUND	Arc	capacity		
UP	BOUND	XPotential node	1		

The code *UP* denotes that the given bound is upper one.

The MPS file for the network file `Health.net` studied in this manual is now presented.

```
NAME          demo
ROWS
 N  Invest
 N  Satisf
 N  Dist
 N  Prox
 E  Acer
 E  Arnika
 E  Betula
 E  Bobrek
 E  Bush
 E  Erica
 E  Fiord
 E  Hill
 E  Ice
 E  Larix
 E  Litwor
 E  Oasis
 E  Picea
 E  Pinus
 E  Pond
```

```
 E  Ribes
 E  Robur
 E  Rumex
 E  Tie
 L  BBush
 L  BFiord
 L  BIce
 L  BOasis
 L  North
 L  South
COLUMNS
    BURO      Invest    0
    BURO      Satisf    0
    BURO      Dist      0.01825
    BURO      Prox      5.21
    BURO      Bush      +1              Robur     -1
    BURO      BBush     +1
    BAR       Invest    0
    BAR       Satisf    0
    BAR       Dist      0.01125
    BAR       Prox      13.72
    BAR       Bush      +1              Arnika    -1
    BAR       BBush     +1
    BUPI      Invest    0
    BUPI      Satisf    0
    BUPI      Dist      0.01225
    BUPI      Prox      11.57
    BUPI      Bush      +1              Pinus     -1
    BUPI      BBush     +1
    BAC       Invest    0
    BAC       Satisf    0
    BAC       Dist      0.007375
    BAC       Prox      31.92
    BAC       Bush      +1              Acer      -1
    BAC       BBush     +1
    BUBO      Invest    0
    BUBO      Satisf    0
    BUBO      Dist      0.005625
    BUBO      Prox      54.87
    BUBO      Bush      +1              Bobrek    -1
    BUBO      BBush     +1
    BUPIC     Invest    0
    BUPIC     Satisf    0
    BUPIC     Dist      0.019625
    BUPIC     Prox      4.51
    BUPIC     Bush      +1              Picea     -1
    BUPIC     BBush     +1
    BUBET     Invest    0
    BUBET     Satisf    0
```

BUBET	Dist	0.01675		
BUBET	Prox	6.19		
BUBET	Bush	+1	Betula	-1
BUBET	BBush	+1		
BER	Invest	0		
BER	Satisf	0		
BER	Dist	0.01425		
BER	Prox	8.55		
BER	Bush	+1	Erica	-1
BER	BBush	+1		
FIORI	Invest	0		
FIORI	Satisf	0		
FIORI	Dist	0.012625		
FIORI	Prox	10.89		
FIORI	Fiord	+1	Ribes	-1
FIORI	BFiord	+1		
FIOLA	Invest	0		
FIOLA	Satisf	0		
FIOLA	Dist	0.015125		
FIOLA	Prox	7.59		
FIOLA	Fiord	+1	Larix	-1
FIOLA	BFiord	+1		
FIORU	Invest	0		
FIORU	Satisf	0		
FIORU	Dist	0.005625		
FIORU	Prox	54.87		
FIORU	Fiord	+1	Rumex	-1
FIORU	BFiord	+1		
FIOPI	Invest	0		
FIOPI	Satisf	0		
FIOPI	Dist	0.003375		
FIOPI	Prox	152.42		
FIOPI	Fiord	+1	Pinus	-1
FIOPI	BFiord	+1		
FIORA	Invest	0		
FIORA	Satisf	0		
FIORA	Dist	0.013625		
FIORA	Prox	9.35		
FIORA	Fiord	+1	Acer	-1
FIORA	BFiord	+1		
FIPIC	Invest	0		
FIPIC	Satisf	0		
FIPIC	Dist	0.01375		
FIPIC	Prox	9.18		
FIPIC	Fiord	+1	Picea	-1
FIPIC	BFiord	+1		
HIRU	Invest	0		
HIRU	Satisf	0		
HIRU	Dist	0.018		

```
HIRU      Prox      5.36
HIRU      Hill      +1              Rumex     -1
HIP       Invest    0
HIP       Satisf    0
HIP       Dist      0.01725
HIP       Prox      5.83
HIP       Hill      +1              Pinus     -1
HIA       Invest    0
HIA       Satisf    0
HIA       Dist      0.008125
HIA       Prox      26.30
HIA       Hill      +1              Acer      -1
HIB       Invest    0
HIB       Satisf    0
HIB       Dist      0.017375
HIB       Prox      5.75
HIB       Hill      +1              Bobrek    -1
HIPIC     Invest    0
HIPIC     Satisf    0
HIPIC     Dist      0.008375
HIPIC     Prox      24.75
HIPIC     Hill      +1              Picea     -1
HILI      Invest    0
HILI      Satisf    0
HILI      Dist      0.00925
HILI      Prox      20.29
HILI      Hill      +1              Litwor    -1
HIB       Invest    0
HIB       Satisf    0
HIB       Dist      0.008125
HIB       Prox      26.30
HIB       Hill      +1              Betula    -1
HIER      Invest    0
HIER      Satisf    0
HIER      Dist      0.0155
HIER      Prox      7.23
HIER      Hill      +1              Erica     -1
IRI       Invest    0
IRI       Satisf    0
IRI       Dist      0.00975
IRI       Prox      18.26
IRI       Ice       +1              Ribes     -1
IRI       BIce      +1
ILA       Invest    0
ILA       Satisf    0
ILA       Dist      0.005625
ILA       Prox      54.87
ILA       Ice       +1              Larix     -1
ILA       BIce      +1
```

```
IRO     Invest   0
IRO     Satisf   0
IRO     Dist     0.0165
IRO     Prox     6.38
IRO     Ice      +1          Robur    -1
IRO     BIce     +1
IAR     Invest   0
IAR     Satisf   0
IAR     Dist     0.017
IAR     Prox     6.01
IAR     Ice      +1          Arnika   -1
IAR     BIce     +1
IRU     Invest   0
IRU     Satisf   0
IRU     Dist     0.011875
IRU     Prox     12.31
IRU     Ice      +1          Rumex    -1
IRU     BIce     +1
IPI     Invest   0
IPI     Satisf   0
IPI     Dist     0.0075
IPI     Prox     30.86
IPI     Ice      +1          Pinus    -1
IPI     BIce     +1
OAR     Invest   0
OAR     Satisf   0
OAR     Dist     0.014625
OAR     Prox     8.12
OAR     Oasis    +1          Arnika   -1
OAR     BOasis   +1
OPI     Invest   0
OPI     Satisf   0
OPI     Dist     0.01725
OPI     Prox     5.83
OPI     Oasis    +1          Pinus    -1
OPI     BOasis   +1
OAC     Invest   0
OAC     Satisf   0
OAC     Dist     0.0065
OAC     Prox     41.09
OAC     Oasis    +1          Acer     -1
OAC     BOasis   +1
OBO     Invest   0
OBO     Satisf   0
OBO     Dist     0.006875
OBO     Prox     36.73
OBO     Oasis    +1          Bobrek   -1
OBO     BOasis   +1
OLI     Invest   0
```

OLI	Satisf	0		
OLI	Dist	0.019375		
OLI	Prox	4.62		
OLI	Oasis	+1	Litwor	-1
OLI	BOasis	+1		
OBET	Invest	0		
OBET	Satisf	0		
OBET	Dist	0.01275		
OBET	Prox	10.68		
OBET	Oasis	+1	Betula	-1
OBET	BOasis	+1		
OER	Invest	0		
OER	Satisf	0		
OER	Dist	0.01025		
OER	Prox	16.52		
OER	Oasis	+1	Erica	-1
OER	BOasis	+1		
POR	Invest	0		
POR	Satisf	0		
POR	Dist	0.01725		
POR	Prox	5.83		
POR	Pond	+1	Ribes	-1
POL	Invest	0		
POL	Satisf	0		
POL	Dist	0.010875		
POL	Prox	14.68		
POL	Pond	+1	Larix	-1
PORO	Invest	0		
PORO	Satisf	0		
PORO	Dist	0.009125		
PORO	Prox	20.85		
PORO	Pond	+1	Robur	-1
POA	Invest	0		
POA	Satisf	0		
POA	Dist	0.00725		
POA	Prox	33.03		
POA	Pond	+1	Arnika	-1
PORU	Invest	0		
PORU	Satisf	0		
PORU	Dist	0.018875		
PORU	Prox	4.87		
PORU	Pond	+1	Rumex	-1
POP	Invest	0		
POP	Satisf	0		
POP	Dist	0.0095		
POP	Prox	19.24		
POP	Pond	+1	Pinus	-1
POAC	Invest	0		
POAC	Satisf	0		

```
POAC      Dist      0.01675
POAC      Prox      6.19
POAC      Pond      +1              Acer      -1
POB       Invest    0
POB       Satisf    0
POB       Dist      0.016
POB       Prox      6.78
POB       Pond      +1              Bobrek    -1
TIEP      Invest    0
TIEP      Satisf    0
TIEP      Dist      0
TIEP      Prox      0
TIEP      Tie       +1              Pond      -1
TIEH      Invest    0
TIEH      Satisf    0
TIEH      Dist      0
TIEH      Prox      0
TIEH      Tie       +1              Hill      -1
TIEI      Invest    0
TIEI      Satisf    0
TIEI      Dist      0
TIEI      Prox      0
TIEI      Tie       +1              Ice       -1
TIEF      Invest    0
TIEF      Satisf    0
TIEF      Dist      0
TIEF      Prox      0
TIEF      Tie       +1              Fiord     -1
TIEB      Invest    0
TIEB      Satisf    0
TIEB      Dist      0
TIEB      Prox      0
TIEB      Tie       +1              Bush      -1
TIEO      Invest    0
TIEO      Satisf    0
TIEO      Dist      0
TIEO      Prox      0
TIEO      Tie       +1              Oasis     -1
INTPOTB   'MARKER'                  'INTORG'
XBush     Invest    186
XBush     Satisf    100
XBush     Dist      0
XBush     Prox      0
XBush     BBush     -50
XBush     South     +1
XFiord    Invest    212
XFiord    Satisf    87
XFiord    Dist      0
XFiord    Prox      0
```

```
    XFiord    BFiord    -60
    XFiord    North     +1
    XIce      Invest    200
    XIce      Satisf    176
    XIce      Dist      0
    XIce      Prox      0
    XIce      BIce      -50
    XIce      North     +1
    XOasis    Invest    201
    XOasis    Satisf    192
    XOasis    Dist      0
    XOasis    Prox      0
    XOasis    BOasis    -60
    XOasis    South     +1
    INTPOTE   'MARKER'                  'INTEND'
RHS
    RHS       Acer      -16
    RHS       Arnika    -20.5
    RHS       Betula    -13
    RHS       Bobrek    -22
    RHS       Erica     -23.5
    RHS       Hill      0
    RHS       Larix     -25
    RHS       Litwor    -20.5
    RHS       Picea     -15
    RHS       Pinus     -20
    RHS       Pond      0
    RHS       Ribes     -24.5
    RHS       Robur     -21
    RHS       Rumex     -19
    RHS       Tie       240
    RHS       North     1
    RHS       South     1
RANGES
    RANGE     North     1
    RANGE     South     1
BOUNDS
 UP BOUND     BURO      50
 UP BOUND     BAR       50
 UP BOUND     BUPI      50
 UP BOUND     BAC       50
 UP BOUND     BUBO      50
 UP BOUND     BUPIC     50
 UP BOUND     BUBET     50
 UP BOUND     BER       50
 UP BOUND     FIORI     60
 UP BOUND     FIOLA     60
 UP BOUND     FIORU     60
 UP BOUND     FIOPI     60
```

```
 UP BOUND      FIORA     60
 UP BOUND      FIPIC     60
 UP BOUND      HIRU      90
 UP BOUND      HIP       90
 UP BOUND      HIA       90
 UP BOUND      HIB       90
 UP BOUND      HIPIC     90
 UP BOUND      HILI      90
 UP BOUND      HIB       90
 UP BOUND      HIER      90
 UP BOUND      IRI       50
 UP BOUND      ILA       50
 UP BOUND      IRO       50
 UP BOUND      IAR       50
 UP BOUND      IRU       50
 UP BOUND      IPI       50
 UP BOUND      OAR       60
 UP BOUND      OPI       60
 UP BOUND      OAC       60
 UP BOUND      OBO       60
 UP BOUND      OLI       60
 UP BOUND      OBET      60
 UP BOUND      OER       60
 UP BOUND      POR       100
 UP BOUND      POL       100
 UP BOUND      PORO      100
 UP BOUND      POA       100
 UP BOUND      PORU      100
 UP BOUND      POP       100
 UP BOUND      POAC      100
 UP BOUND      POB       100
 UP BOUND      TIEP      100
 UP BOUND      TIEH      90
 UP BOUND      TIEI      50
 UP BOUND      TIEF      60
 UP BOUND      TIEB      50
 UP BOUND      TIEO      60
 UP BOUND      XBush     1
 UP BOUND      XFiord    1
 UP BOUND      XIce      1
 UP BOUND      XOasis    1
ENDATA
```

MCBARG - Enhanced
A System Supporting Multicriteria Bargaining

Lech Kruś, Piotr Bronisz, Bożena Łopuch
Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland.

Abstract

The MCBARG is a decision support system designed to help in the analysis of decision situations and in the mediation in multicriteria bargaining problems. It supports reaching the final outcome in the problem. The report describes an enhanced version of the MCBARG system. It provides the user with a theoretical foundation and with information necessary to use the system.

The system and 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 in Laxenburg, Austria, and with financial support of the Research Program CPBP 02.15 of the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland.

1 Introduction

This paper deals with decision problems in multiactors game-like-situations in which the decision makers (individuals, firms, nations - "players") have conflicting interests and their final outcomes are determined within interactive bargaining. There are many complex situations in this class of problems in which the decision makers need help to learn about possible decision options and decision consequences. The MCBARG system enables learning process of the players, and supports reaching the final outcome in the multicriteria bargaining problem. It is based on some new theoretical results on interactive mediation processes with an application of aspiration-led methodology of decision support system.

The multicriteria bargaining problem is a generalization of classical bargaining problem, under the assumption that there are not given explicitly the utility functions of players. This generalization follows from the fact that an aggregation of players' objectives is often impossible because of various practical limitations of the utility theory. The problem is defined by an agreement set — the set of outcomes that can be reached under unanimous agreement of the players, and by a disagreement (status quo) point which is a result of the problem if there is no such an agreement.

The interactive mediation process is described by a procedure which consists in generation of a sequence of outcomes leading to a nondominated solution. The procedure is based on a limited confidence principle, taken from practical observation, which says that the players have limited confidence in their ability to predict consequences and possible outcomes, hence each player tries to prevent other players from receiving disproportionally large gains. The generated outcomes are consistent with preferences of the players.

The procedure assures some fairness rules and is resistant to the various manipulations of the players.

The MCBARG system includes a generator and an editor of the model of the bargaining problem for which the interactive procedure is performed. The model describes the agreement set in form of a set of inequalities, and the status quo point. The generator and the editor enable introducing linear or nonlinear formulae defining the inequalities using standard arithmetic operators and some common functions.

This latest version of the MCBARG system includes the following extensions in comparison to the older version described by Bronisz P., L. Krus, B. Lopuch (1988):

— A new user interface of the system has been elaborated including a new system of menus, new options, and new graphical presentation of results, which makes, as it seems, the system more user-friendly.

— In this version, explicit division of roles of the system analyst (named in the system "OPERATOR") and of the players has been introduced. It is assumed, that at first the bargaining problem is prepared by the system analyst, next, the system analyst begins and conducts the negotiation procedure in which human players participate.

— The players have limited access to the counterplayers information. The information is protected by passwords.

— The players can simulate and analyze the counterplayers decisions and their impact on the obtained outcomes. In the older version, the counterplayers decisions during the unilateral analysis were assumed by default values of the system.

— A new option OLD SESSION has been introduced. The system enables a user to save a performed session on a disk. The saved session can be analyzed, and if needed, can be used to restart the negotiation procedure.

— The editor of the model and the solver procedures have been modified. In the model of the bargaining problem minimized and maximized criteria can be explicitly formulated. In the older version all criteria were assumed to be maximized. New routines for checking formal correctness of the model and proper formulation of the bargaining problem have been included.

— A new illustrative example referring to an international cooperation on the acid rains problem has been elaborated and used as a tutorial in the paper.

This report is organized as follows. Section 2 contains mathematical formulation of the multicriteria bargaining problem and methodological background of solution concept and of interactive algorithm. Section 3 explains the use of the system, i.e. how to create model of a bargaining game and how to support mediation. Section 4 presents an illustrative example of a bargaining game played with the system support. Section 5 contains technical information.

Section 2 is a revised version of the theoretical foundations presented in the description of the previous version of the MCBARG system. Sections 3, 4 and 5 have been completely rewritten to cover the new version of the MCBARG system.

2 Theoretical foundations

2.1 Introduction

In most approaches (see Nash, 1950, Raiffa, 1953, Kalai and Smorodinsky, 1975, 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.

We consider n players, each with several objectives, so we deal with a multiobjective bargaining problem. In this problem, the players are faced with an agreement set of feasible outcomes. 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 status quo point is the result. If there are feasible outcomes which all participants prefer to the status quo point, then there is an incentive to reach an agreement. In most situations, players differ in their opinions about which outcome is most preferable, hence there is a need for bargaining and negotiation.

When dealing with multiple payoffs, we do not assume that there exist explicitly given utility functions of the players. In this case the solution can be found in an interactive procedure. Such a procedure is considered here. The procedure starts from the status quo point and leads 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.

2.2 Problem formulation and definitions

Let $N = \{1, 2, \ldots, n\}$ be the finite set of players, each player having m_i objectives. A multiobjective bargaining problem is defined as a pair (S, d), where an agreement set S is a subset of $\sum_{i=1}^{n} m_i$ – dimensional Euclidean space, called R^{NM}, and a disagreement point (status quo point) d belongs to S.

The bargaining problem has the following interpretation: every point $x \in R^{NM}$, $x = (x_1, x_2, \ldots, x_n)$, $x_i = (x_{i1}, x_{i2}, \ldots, x_{im_i})$, 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, the disagreement point is the result.

Each objective can be maximized or minimized. For simplicity of problem formulation in Section 2 we assume that all objectives are maximized.

We employ a convention that for $x, y \in R^k$, $x \geq y$ implies $x_i \geq y_i$ for $i = 1, \ldots, k$, $x > y$ implies $x \geq y$, $x \neq y$, $x \gg y$ implies $x_i > y_i$ for $i = 1, \ldots, k$. 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 strict Pareto optimal point in X if there is no $y \in X$ such that $y > x$.

We confine our consideration to the class of all multicriteria bargaining games (S, d) satisfying the following conditions:

(i) S is compact and there is $x \in S$ such that $x > d$,
(ii) S is comprehensive, i.e. for $x \in S$ if $d \leq y \leq x$ then $y \in S$.
(iii) For any $x \in S$, let $Q(S, x) = \{ i : y \geq x,\ y_i > x_i \text{ for some } y \in S \}$. Then for any $x \in S$, there exists $y \in S$ such that $y \geq x$, $y_i > x_i$ for each $i \in Q(S, x)$.

Condition (i) states that the set S is closed, upper bounded and the problem is not degenerated. 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. $Q(S, x)$ is the set of all coordinates in R^{NM}, payoffs of whose members can be increased from x in S.

Condition (iii) states that the set of Pareto optimal points in S contains no "holes". We do not assume convexity of S, however, any convex set satisfies Condition (iii).

The problem consists in supporting the players in reaching a nondominated solution, agreeable and close to their preferences.

Definitions:

A point $x^i \in S$ is defined as i-nondominated, $i \in N$, if there is no $y \in S$ such that $y_i > x_i^i$. A point $u \in R^{NM}$ is defined as a *utopia point relative to aspirations* (RA utopia point) if for each player $i \in N$, there is an i-nondominated point $x^i \in S$ such that $u_i = x_i^i$.

The i-nondominated point is an outcome which could be achieved by a rational player i if he would have full control of the moves of the other players. Let us observe that if in the unicriterial set there is only one i-nondominated point, then in the multicriteria case considered here there is a set of such points. Each player i, $i \in N$, is required then to investigate the set of i-nondominated points in S as m_i-dimensional multicriteria decision problem and then to select one i-nondominated point as his most preferable outcome.

The RA utopia point generated by the selected i-nondominated points, $i \in N$, carries information about the most preferable outcomes for all the players. The RA utopia point significantly differs from the ideal (utopia) point defined by the maximal values of all objectives in set S.

2.3 Interactive procedure

We are interested in a constructive procedure that is acceptable to all the players, starts at the status quo point and leads to a strict 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 $d^0 = d$, $d^t \in S$, $d^t \geq d^{t-1}$, for $t = 1, 2, \ldots$, d^k is a strict Pareto optimal point in S. (The assumption $d^t \geq d^{t-1}$ 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 preferable reference point $r_i^t \in R^{m_i}$, $r_i^t > d_i^t$ defining his improvement direction $\lambda_i^t \in R^{m_i}$, $\lambda_i^t = r_i^t - d_i^t$, and proposes his confidence coefficient $\alpha_i^t \in R$, $0 < \alpha_i^t \leq 1$. 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, 1988).

We propose an interactive negotiation process defined by a procedure:

$$(*) \qquad \begin{aligned} &\{d^t\}_{t=0}^{\infty} \qquad \text{such that} \quad d^0 = d, \\ &d^t = d^{t-1} + \varepsilon^t * [u(S, d^{t-1}, \lambda^t) - d^{t-1}] \qquad \text{for} \quad t = 1, 2, \ldots, \end{aligned}$$

where

$\lambda^t \in R^{NM}$, $\lambda^t = (\lambda_1^t, \lambda_2^t, \ldots, \lambda_n^t)$, is the improvement direction specified jointly by all the players,

$u(S, d^{t-1}, \lambda^t) \in R^{NM}$ is the utopia point relative to the direction λ^t at round t defined by

$$u(S, d^{t-1}, \lambda^t) = (u_1(S, d^{t-1}, \lambda_1^t), u_2(S, d^{t-1}, \lambda_2^t), \ldots, u_n(S, d^{t-1}, \lambda_n^t)),$$

$$u_i(S, d^{t-1}, \lambda_i^t) = \max_{\geq} \left\{ x_i \in R^{m_i} : x \in S,\ x \geq d^{t-1},\ x_i = d_i^{t-1} + a\lambda_i^t \text{ for some } a \in R \right\},$$

$\varepsilon^t = \min(\alpha_1^t, \alpha_2^t, \ldots, \alpha_n^t, \alpha_{\max}^t) \in R$, where $\alpha_{\max}^t$ is the maximal number α such that $d^{t-1} + \alpha\left[u(S, d^{t-1}, \lambda^t) - d^{t-1}\right]$ belongs to S.

The utopia point $u(S, d^{t-1}, \lambda^t)$ relative to the aspirations of the players (relative to direction λ^t) reflects the preferences of the particular players when the improvement direction λ^t is specified at round t. The individual outcome $u_i(S, d^{t-1}, \lambda_i^t)$ is the maximal payoff in S for the i-th player from d^{t-1} according to the 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. The preferable direction λ_i^t at round t is specified on the basis of interactive scanning of a number of solutions generated for assumed by players different reference points. 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 be an influence on the selection of solutions by changing reference points. To solve the problem, directional maximization is applied, using a bisection method. (see Bronisz and Krus, 1988). The scanning (called in the system the improvement directions testing) is performed independently by each of players. Given the information about the current status quo and ideal point the player proposes a number of reference points and confidence coefficient. For each reference point $r_i^t > d_i^t$ and confidence coefficient α_i^t given by the player at the round t, the system calculates:

RA-utopia:

$$u_i(S, d^{t-1}, \lambda_i^t) = \max_{\geq}\left\{x_i \in R^{m_i} : x \in S,\ x \geq d^{t-1},\ x_i = d_i^{t-1} + a\lambda_i^t \text{ for some } a \in R\right\},$$

one-shot solution:

$$x^t = \max_{\geq}\left\{x \in S : x = d^{t-1} + a * [u(S, d^{t-1}, \lambda^t) - d^{t-1}] \quad \text{for some} \quad a \in R\right\},$$

anticipated solution:

$$y^t = d^{t-1} + \varepsilon^t * [u(S, d^{t-1}, \lambda^t) - d^{t-1}]$$

maximal confidence coefficient:

$$\alpha_{\max}^t = \max_{\geq}\left\{a \in R : d^{t-1} + a * [u(S, d^{t-1}, \lambda^t) - d^{t-1}] \in S \quad \text{for some} \quad a \in R\right\},$$

where $\lambda^t \in R^{NM}$, $\lambda^t = (\lambda_1^t, \lambda_2^t, \ldots, \lambda_n^t)$, $\lambda_j^t = \lambda_j^{t-1}$ for $j \neq i$, $\lambda_i^t = r_i^t - d_i^t$, $u(S, d^{t-1}, \lambda^t) \in R^{NM}$ is the utopia point relative to the direction λ^t.

Having the above information for a number of reference points, the player then selects his prefered one. It defines the improvement direction of this player. Defined in this way, improvement directions of all the players λ_i^t are used for the calculation of the result d^t of the negotiation round.

The procedure is based on the following theoretical result (see: Bronisz, Krus, Lopuch, 1987, and Bronisz, Krus, Wierzbicki, 1988):

Theorem 1. For any multicriteria bargaining game (S, d) satisfying conditions (i), (ii) and (iii) and for any confidence coefficients α_i^t such that

$$0 < \varepsilon \leq \alpha_i^t \leq 1,\ t = 1, 2, \ldots, T$$

there is a unique process d^t, $t = 0, 1, \ldots, T$, $T \leq \infty$, described by $(*)$ satisfying the following postulates:

P1. $d^0 = d,\ d^t \in S$ for $t = 1, 2, \ldots, T$,

P2. $d^t \geq d^{t-1}$ for $t = 1, 2, \ldots, T$,

P3. d^T ($= \lim_{t \to \infty} d^t$ if $T = \infty$) is a strict Pareto optimal point in S.

P4. *Principle of α-limited confidence.* Let $0 < \alpha_i^t \leq 1$ be a given confidence coefficient of the i-th player at round t. Then acceptable demands are limited by:

$$d^t - d^{t-1} \leq \alpha_{\min}^t [u(d^{t-1}) - d^{t-1}] \text{ for } t = 1, \ldots, T,$$

where $\alpha_{\min}^t$ is a joint confidence coefficient at round t, $\alpha_{\min}^t = \min\{\alpha_1^t, \ldots, \alpha_n^t\}$, $u(d^{t-1})$ is the RA-utopia point which reflects the preferences of the players in the subset of the set S given by $\{ x \in S : x \geq d^{t-1} \}$

P5. *Principle of recursive rationality.* Given d^t, at each round t, there is no such outcome $x \in S,\ x > d^t$, that x satisfies P4 (x substitutes d^t in P4).

P6. *Principle of proportional gains.* For each round t, $t = 1, \ldots, T$, there is a number $\beta > 0$ such that

$$d^t - d^{t-1} = \beta[u(d^{t-1}) - d^{t-1}].$$

The presented approach has been examined in a case of one-round process with confidence coefficients of the players equal to one. The corresponding one-shot solution has been characterized axiomatically (Bronisz and Krus, 1987a). It is easy to notice that in the unicriterial case, each game (S, d) has a unique utopia point which coincides with the ideal point and the one-shot solution coincides with the Raiffa solution (see Raiffa, 1953, Roth, 1979).

It is assumed that after testing of right amount of reference points each player selects his preferable direction. However, it may happend that a player has not sufficiently tested his set of nondominated points and selects a weak Pareto outcome as his preferable result. In such a case, even if all the players assume the confidence coefficients greater than the values of maximal confidence coefficients, the procedure should proceed in several iterations more, untill it reaches a strict Pareto solution in S. In the system this inconvenience is removed by the application of an option of lexicographical improvement of a weak Pareto solution to a strict Pareto one without interaction of the players. The option is used only in the case that all the players assume their confidence coefficients greater than the values of maximal confidence coefficients. In such a case it is assumed that they are going to finish the interactive process. The lexicographical improvement proceeds in the following way:

Let us assume that in round t the obtained agreement point

$$d^t = d^{t-1} + \varepsilon * [u(S, d^{t-1}, \lambda^t) - d^{t-1}]$$

is a weak Pareto optimal. For a finite subsets of integer numbers I, J, let

$$e(I, J) = (e_1(I, J), \ldots, e_n(I, J)) \in R^{NM}$$

be such that $e_{ij}(I, J) = \lambda^t$ for $i \in I$ and $j \in J$, otherwise $e_{ij}(I, J) = 0$.

Given $y \in S$ with $Q(S, y) \neq \emptyset$, define $x(S, y) \in S$ by

$$x(S, y) = \max_{\geq} \left\{ x \in S : x = y + a * e(Q(S, y)) \quad \text{for some} \quad a \in R \right\}.$$

Intuitively, the vector $e(Q(S,y))$ includes all the coordinates of vector λ^t, and along with the solution may be improved. Otherwise, the corresponding coordinate of the vector $e(Q(S,y))$ is equal to 0. Then the lexicographical improvement can be defined by the sequence $\{x^j\}_{j=0}^{\infty}$ such that $x^0 = d^t$, and $x^j = x(S, x^{j-1})$ for $j = 1, 2, \ldots$. It is shown (Bronisz and Krus, 1988) that there is exactly one such sequence, moreover this sequence is finite.

The presented lexicographical improvement has been examined in a case of one-round process, i.e. when a weak Pareto optimal solution is reached in the first round. It is shown (Bronisz and Krus, 1988) that in such a case the solution of the bargaining process can be described with the Rawlsian lexmin principle (Rawls, 1971). Moreover, in the unicriterial case, the solution coincides with the Imai solution (Imai, 1983).

3 User guide

3.1 General information

The MCBARG system is a decision support system designed to help in analysis of a decision situation and mediation in multicriteria bargaining problem in which a mathematical model of the problem can be formulated by a status-quo point and a system of inequalities describing agreement set in objective space of the players.

The program is recorded on one diskette that can be installed on an IBM–PC–XT, AT or compatible computer with Hercules Graphics Card, Color Graphic Adapter (CGA) or Enhanced Graphics Adapter (EGA). A diskette contains compiled code of the program together with some data files for a demonstrative example of the bargaining problem.

The system supports the following general functions:

1. The definition and edition of a model of the bargaining problem.

2. The analysis of the bargaining problem and performing interactive negotiation-mediation procedure.

The model definition and edition should be done by a system analyst called further in the system an `OPERATOR`. The `OPERATOR` prepares and conducts the interactive negotiation procedure. He gives a control to players during performing negotiation-mediation session. He takes control over the system again after the session is interrupted or finished.

The interactive procedure is performed by human players. It proceeds in a number of iterations (rounds) and in each iteration the system supports the players in:

- Initial multiobjective analysis of the bargaining problem resulting in an estimation of bounds on efficient outcomes and learning about the extreme and neutral outcomes.

- Unilateral, interactive analysis of the problem with stress on learning, organized through system response to user specified confidence coefficients and aspiration levels for objective outcomes.

- Calculation of the multilateral, cooperative solutions according to player preferences.

The system is operated with the use of a set of menus. Particular options can be selected by the option first letter or by cursor and <Enter> key.

The system is self-explaining, it includes a set of information facilitating working with the system.

All of the screens include a highlighted head bar which displays the current phase of the system where the user is, and a bottom help bar displaying operating keys.

In the following subsections more detailed information about particular options of the system menus is presented.

3.2 Main Menu

INFORMATION option

The INFORMATION option presents general information about the system.

MODEL option

The MODEL option enables definition and edition of a model of bargaining problem. The bargaining problem is described by a set of players having several criteria, by an agreement set and by a status quo point.

The following options of MODEL menu allow:

INFORMATION– present the information about the actual phase of work,

LOAD – load an existing model or give a name to a new one,

EDIT – edit an existing model or create a new one,

BUILD – translate the loaded model into an internal form and perform initial calculations,

QUIT – exit to MAIN menu.

The LOAD option enables the selection of a model from displayed model names or the insertion of a new model name (each model is stored in a file having the model name and the default extention .MOD). Short information about the highlighted model including its name, number of players, their names and numbers of criteria is given in the window on the right side of the screen.

The EDIT option enables edition or creation of a model. The built-in editor is especially designed for this purpose. This option invokes the EDIT MODEL menu described in the following section.

To utilize the model in the NEGOTIATION option of the MAIN menu first it has to be translated into an internal form. This is done by the BUILD option. The system performs the needed initial calculations and checks the correctness of the model of bargaining problem. In particular, it is checked whether the agreement set is bounded, and the status quo point belongs to the agreement set.

OLD SESSION option

The option makes it possible load and view an old session. It is useful for analysis as well as for restarting from the previously performed and saved session. Options displayed at OLD SESSION menu enable to:

INFORMATION– present the information about the actual phase of work,

LOAD – load an old session,

VIEW – view the session results,

PRINT – print the session report,

QUIT – exit to **MAIN** menu.

The **LOAD** option enables the selection of a session from displayed session names (each session is stored in a file having the session name and the default extension .**SES**). Short information about the highlighted session including: model and session name, number of players, their names and numbers of criteria, is given in the window on the right side of the screen.

The **VIEW** option supports reviewing the results obtained in succeeding iterations of the session selected with the use of the **LOAD** option. The following information are presented on the screen:

- player name,
- number of reviewed iteration,
- status of the session (finished or not),
- confidence coefficients: joint and assumed by the player

next in the upper table:

- status quo,
- agreement outcomes in the previous iteration,
- agreement outcomes in the given iteration,
- one-shot solution,
- ideal point

and in the lower table

- graphical presentation of results. The bars represent improvements obtained at the agreement solution in comparison to the status quo (scaled by the distance between the status quo and the ideal point), starting from the left side of the window for the criteria to be maximized, and from the right side - for the minimized ones. The highlighted part of the bar represents the improvement obtained in comparison to the previous iteration result.

The **OPERATOR** can use following set of keys:

- <left, right arrow> - scroll the reviewed iterations,
- <Ctrl PgUp, Ctrl PgDn> - set active criterion (to be scrolling),
- <PgUp, PgDn> - scroll the criteria,
- <Ctrl P> - scroll the players.

The **PRINT** option makes it possible to print a report of the current session in aggregated or in detailed form. In the aggregated report only the status quo, the ideal point and the agreement points at succeeding iterations are presented. In the detailed report all the data presented at **VIEW** option are printed.

NEGOTIATION option

The **NEGOTIATION** option activates the negotiation-mediation procedure. This version of the system is built for one computer, therefore the players can enter into the system in a sequence, once at each iteration. In the case where one of the players interrupts the procedure or the session is finished, the control should be given to the **OPERATOR**.

The information devoted to a particular player is protected with the use of a password. The password is fixed during the first entry of the player or the **OPERATOR**. Before the session, the players should agree upon a level of availability of their information to the counterplayers. There are three possibilities in the system, called access levels:

LEVEL 0 – no access to information about counterplayers criteria nor results,

LEVEL 1 – access to information about counterplayers criteria and past results, possible simulation of the counterplayers decisions and observation of their effect on the player results,

LEVEL 2 – access to information about counterplayers criteria and past results, possible simulation of the counterplayers decisions, and observation of their effect on the player and the counterplayers results.

The agreed player access level, to be enforced in the session, is entered by the **OPERATOR** at the beginning of the session.

Any player or **OPERATOR** enters the system by selection of his name. The player takes control over the system with use of the **NEGOTIATION** menu, while the **OPERATOR** with use of the **OPERATOR** menu.

3.3 Edit Model Menu

To create a new model the following values must be entered with use of the **EDIT MODEL** menu :

- number of players and their names (**PLAYERS** option).
  Minimum: 2 players. Maximum: 5 players.
- number of criteria of each player (**CRITERIA** option).
  Minimum: 1 criterion. Maximum: 8 criteria.
- description of criteria (<F2> key in **CRITERIA** option).
- criteria units (<F3> key in **CRITERIA** option).
- status of the criteria - min for minimized and max for maximized (<F4> key in **CRITERIA** option).
- status quo value for each criterion (<F3> key in **CRITERIA** option).
  Default: 0.0.
- set of formulae describing the agreement set (**FORMULAE** option).

The players are recognized by their names. To facilitate a description of criteria names in formulae, the criterion of the given player has the following alphanumeric code: "`ln`", where `l` means letter of the given player (`a` or `A` — first player, `b` or `B` — second player, and so on), `n` denotes number of the criterion. (For example, "`c2`" means variable for the second criterion of the third player).

The set of formulae describing an agreement set should be in the form:

```
formula_name_1 = expression_1 ;
formula_name_2 = expression_2 ;
.....................................
formula_name_N = expression_N ;
```

which gives the following set of inequalities:

$$\texttt{expression\_I} \geq 0 \quad \text{for} \quad I = 1, 2, \ldots, N.$$

A `formula_name_I`, $I = 1, \ldots, N$, should begin with a letter. It can be treated as a comment to an `expression_I`. An `expression_I`, $I = 1, \ldots, N$, is Pascal-language-like expression, which can include:

- constants — in normal or in exponential form;
- variables for criteria — as presented in the above alphanumeric code "`ln`";
- operators divided into 4 categories, listed by their order of precedence:
  1) Exponentiation: `^` ,
  2) Unary minus: `-` ,
  3) Multiplying operators: `*`, `/` ,
  4) Adding operators: `+`, `-` ;
- parentheses: `(`, `)` ;
- arithmetic functions: `sin`, `cos`, `exp`, `log`.

  `sin` – returns the sine of the argument (the argument is expressed in radians),
  `cos` – returns the cosine of the argument (the argument is expresses in radians),
  `exp` – returns the exponential of the argument,
  `log` – returns the natural logarithm of the argument.

  The argument of each function should be written in parentheses.

EXAMPLE of a formula for two players, each one having two criteria:

```
formula = 1000 - a1 - exp(a2+b1) - ( b2 - 1.05e-2 )^2 ;
```

An edited model is stored in the memory, if you want to use the same model in the future you should write it on the disk when leaving the `EDIT MODEL` option.

The `BUILD` option preforms the same actions as the `BUILD` option in the `MODEL` menu (translates the model into an internal form and performs needed initial calculations). It is included into this menu for user convenience.

3.4 Negotiation Menu

Each player enters the system by selection of his name and typing the password. The following options of NEGOTIATION menu allow him to:

INFORMATION – present the information about the actual phase of work,

DECISION – test and make decision in the given iteration,

VIEW – view obtained results,

PRINT – print the current session report,

QUIT – exit to the NEGOTIATION option.

DECISION option

The DECISION option enables testing different improvement directions starting from the last agreement point (in the first iteration it is a status quo point). It allows for an interactive scanning of variants of outcomes. The variants are generated by the player for specified reference points. The points correspond to the improvement directions. The resulting outcomes are calculated for the player at the assumed confidence coefficient and at the assumed improvement directions of the other players. By the scanning the player can test his preferences and select his preferable outcome and corresponding reference point.

The following options of the DECISION menu allow the player:

CONF_COEF – specification of confidence coefficient,

NEW_REF – specification of reference point and creation of a new variant,

VIEW_CNTR_PL – displaying results for simulated counterplayers actions (accessible for the LEVEL 1 and the LEVEL 2 only),

SELECT – selection of the preferable variant,

QUIT – exit to the DECISION menu.

The keys presented in bottom help lines enable:

- scrolling the variants (<Tab> and <Up, Dn arrows>),
- scrolling the criteria (<Ctrl PgUp>, <Ctrl PgDn>, <PgUp>, <PgDn>),
- displaying the anticipated solution in the graphical form (<Ctrl G>).

Using these options the player can create and compare a number of outcome variants. The system presents:

- status quo,
- cooperative solution obtained at the previous iteration (Prev. sol.),
- ideal point.

These values define a range of possible outcomes and reasonable reference points.

The created variants are stored on the disk. The first variant is created by the system, under the assumption that the improvement direction is defined according to the ideal point, and serves the player introductory information. The solution presented at this variant is called the neutral solution.

For each variant the system calculates and presents:

- anticipated outcomes (Ant. sol.),
- utopia points relative to aspirations of the player (RA-utopia point).

The RA-utopia point denotes the outcomes the player could obtain under the assumption that the other players get no improvement at this iteration. The aniticipated outcomes are calculated for the assumed actions of the counterplayers. By system default, these actions are based on the variants selected by the counterplayers at the previous iteration (in the 1-st iteration the ideal point is taken).

The CONF_COEF and NEW_REF options can be used for generation of a new variant. The CONF_COEF option makes it possible to specify to the player the assumed value of this iteration confidence coefficient (within the range $(0, 1]$). The NEW_REF option enables one to enter the references for the player's objectives for which the outcome will be calculated. In the case of LEVEL 0 the values for the counterplayers references are specified by the system. In the case of LEVEL 1 and LEVEL 2, the player can accept the default values for the counterplayer references or specify them himself.

Comparison of outcomes from different assumed actions of counterplayers is possible with use of the VIEW_CNTR_PL option. This option can be activated only for the access LEVEL 1 and LEVEL 2. Selecting this option the player obtains information on his one-shot outcomes (in the case of LEVEL 1 access) and on one-shot outcomes of counterplayers (in case of the LEVEL 2 access). The one-shot solution denotes outcomes the player could obtain under the assumption of the full confidence (i.e. confidence coefficient equal to one). The information is presented in two tables - the table on the left side of the screen relates to the particular counterplayer, the table on the right side - to the active player. Scrolling the counterplayers can be done with use of <Ctrl P> keys, toggling the left and right tables with use of <left and right arrow> keys, scrolling variants and criteria - as mentioned above.

After making some tests for different improvement directions, the player should specify the preferable variant. The SELECT option is used for this purpose. This variant will be the base for the agreement point calculation. The agreement solution at the given iteration is calculated when all the players have made their decisions. Then the system starts the next iteration.

VIEW option

The VIEW option supports reviewing the results obtained in the previous iterations. The following information is presented on the screen:

- player or the counterplayer name to which the information refers,
- number of reviewed iterations,
- status of the session (finished or not),

- confidence coefficients: joint and assumed by the operating player,

below of this in the upper table:

- status quo,
- agreement outcomes in the previous iteration,
- agreement outcomes in the given iteration,
- one-shot solution,
- ideal point

and in the lower table

- graphical presentation of results.

The bars represent improvements obtained at the agreement solution in comparison to the status quo (scaled by the distance between the status quo and the ideal point), starting from the left side of the window for the criteria to be maximized, and from the right side - for the minimized ones. The highlighted part of the bar represent the improvement obtained in comparison to the previous iteration result.

The player can use the following set of keys:

- <left, right arrow> - scroll the reviewed iterations,
- <Ctrl PgUp, Ctrl PgDn> - set active criterion (to be scrolling),
- <PgUp, PgDn> - scroll the criteria,
- <Ctrl C> - view the counterplayers (<Ctrl P> - scroll the counterplayers, <Ctrl Y> - return to the operating player).

The information about the counterplayers is presented only if the access LEVEL 1 or LEVEL 2 has been chosen.

PRINT option

The PRINT option makes it possible to print a report of the current session in aggregated or in detailed form. In the aggregated report only the status quo point, the ideal point and the agreement points at succeeding iterations are presented. In the detailed report all the data presented at VIEW option are printed. Results of the counterplayers are printed in the case of the LEVEL 1 and LEVEL 2 accesses.

QUIT option

Warning!

The player should not leave the NEGOTIATION menu before making a decision defining the preferable improvement direction. It is done by selecting the preferable variant with the use of the SELECT option of the DECISION menu. Otherwise, the system assumes that the player interrupts the session.

3.5 Operator Menu

The **OPERATOR** can take control over the system during the negotiation procedure at each iteration. He should take the control if the session has been interrupted or finished. The **OPERATOR** menu includes the following options which allows him to:

VIEW – view the players results,

PRINT – print the session report,

CONTINUE – continue the session,

QUIT – exit to **MAIN** menu.

The description of the **VIEW** and **PRINT** options is analogical as in the case of **OLD SESSION** menu. The **OPERATOR** has full access to the information, disregarding the access level enforced in this session.

The **CONTINUE** option allows the players to continue the negotiation-mediation procedure.

At the **QUIT** option the **OPERATOR** can save the performed session.

4 Example session

After starting from the distributed disk using command line: **MCBARG**, the system will be loaded and you will see on the first screen full name of the program, names of the authors and institutions, version number and date of release (Figure 1).

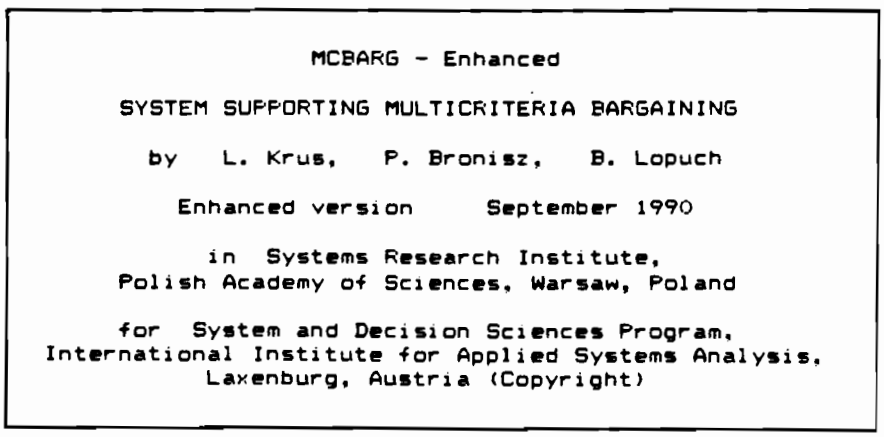


Figure 1

In this example session it is assumed that you play the role of an **OPERATOR** (the system analyst who prepares and conducts the session) and the roles of players participating in the mediation procedure.

You start the work as the **OPERATOR**. Just press any key. The program will ask you to enter the **OPERATOR** password. If you do not use the password just press the <Enter> key. After that the first menu will be displayed (Figure 2). It is the **MAIN** menu and appears in the left upper corner of the screen.

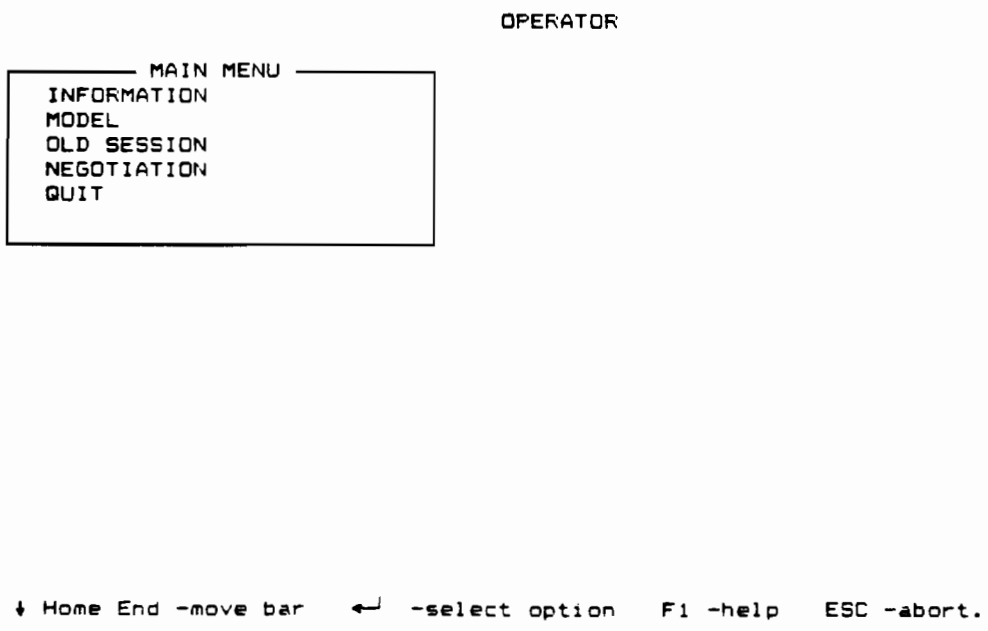


Figure 2

On the top of the screen you can see the highlighted bar (head bar). In the bar, **OPERATOR** or Player names are displayed depending on who has access to the system at the current phase.

At the bottom of the screen you can see another highlighted bar (help bar) with a brief explanation of all active keys at the moment. The help bars will appear during all phases of the work with the system.

Notice that one of the commands (options) of menu is highlighted. This is the command where the cursor is resting. You can activate this command by pressing the <Enter> key or you can move to another command using the arrow keys. You can also activate the requested command by pressing its first letter key. There is, however, one thing you should remember: you can not use the commands in any order. First you have to establish the active model (the **MODEL** or **OLD SESSION** option) and then you can start playing with it. If it is not obvious what to do next, just press <F1> key (**HELP**) or select the **INFORMATION** option for more information.

Let us continue our session. If the cursor remains on the **MODEL** option, just press the <Enter> key. The system will transfer control to the **MODEL** menu displayed in Figure 3.

Once again, you are asked to select an option from the above menu. At first, select the **LOAD** option. (Other options will be inactive). Your screen will look like Figure 4.

In the bottom window you will see names of the models created during the previous sessions. Choose the model named **ACID**.

The ACID model relates to the problem of cooperation of two countries trying to decrease deposition of acid rains (in particular of sulfur). This will be our illustrative example to show how you can work within the MCBARG system. The story describing

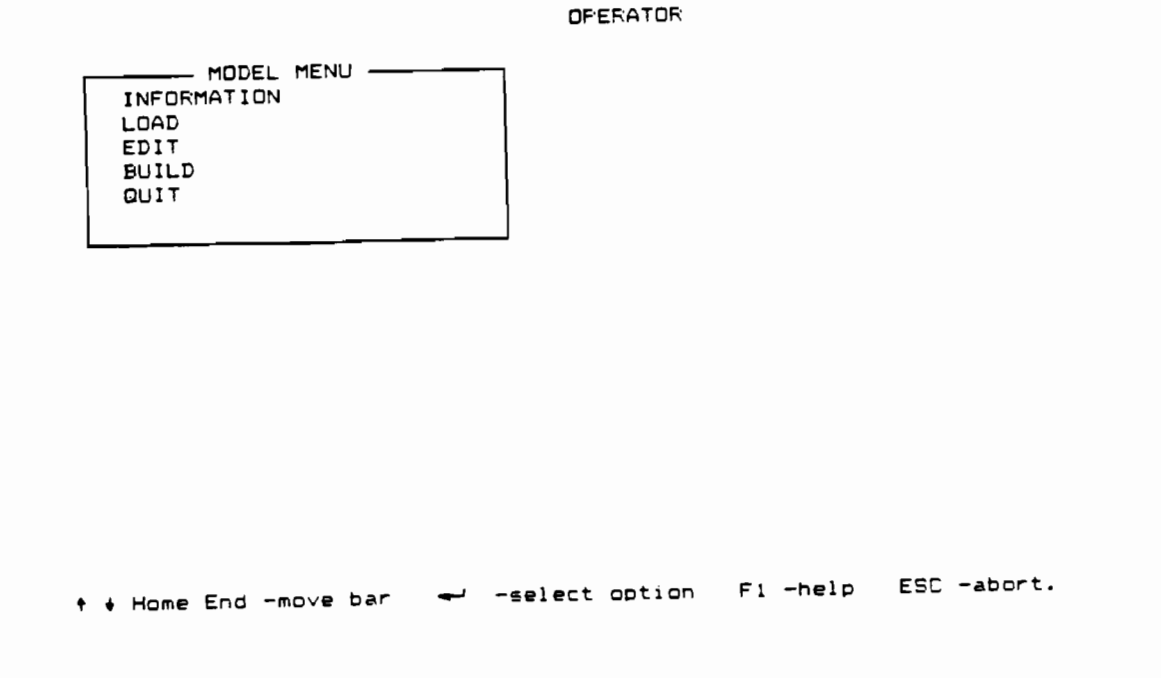


Figure 3

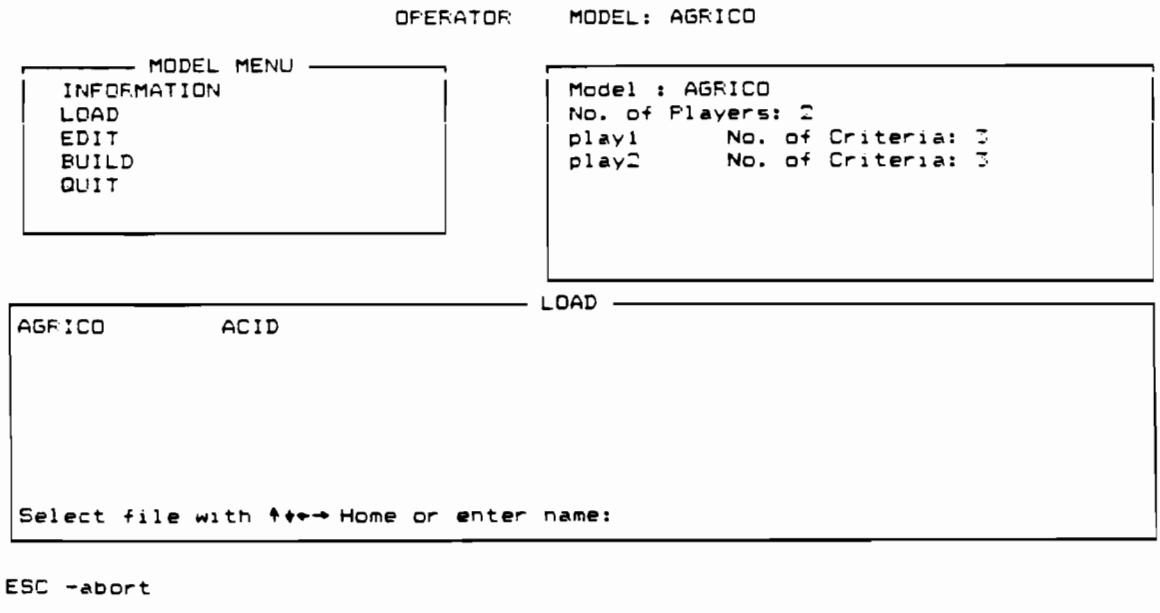


Figure 4

a bargaining problem is as follows:

There are two countries disputing programs reducing sulfur emissions. Each country is assumed to have an adopted plan for emission control and expects the emission level $\underline{E}_i$. However it is also assumed that the deposition levels resulting from the $\underline{E}_i, i = 1,2$ are regarded as unacceptable, therefore an additional emission control program is requested and required for the program expenditures are discussed.

For each country $i = 1,2$ there is a given cost function describing minimal required total cost C_i of reducing the total emission from the level $\underline{E}_i$ to the level E_i. The function is assumed to be decreasing and piece-wise linear.

The sulfur depositions in country i is described by the equation:

$$D_i = a_{i1}E_1 + a_{i2}E_2 + \underline{D}_i, i = 1,2$$

where $a_{i,j}$ $(i,j = 1,2)$ are parameters of the atmospheric transportation matrix (so called European Monitoring and Evaluation Programme matrix), $\underline{D}_i$ are depositions from emission of the other countries.

(The model was inspired by the paper by Bergman L., H. Cesar, G. Klaassen (1990), however in our paper a different approach based on the interactive bargaining is proposed).

The following two situations can be considered.

The first case deals with unilateral actions of the countries. In this case each country is assumed to enforce independently his additional program reducing sulfur emission. Paying $\overline{X}_i = \overline{C}_i$, it achieves the emission $\overline{E}_i$ and deposition $\overline{D}_i$ calculated with use of the cost function and the deposition equation (where $\overline{C}_i$ is cost of the program).

In the second case cooperation of the countries is assumed, in a form of a bilateral agreement on a joint reducing program. In this case a joint fund is created, and the following equation is added to the model description:

$$X_1 + X_2 = C_1 + C_2$$

where X_i, $i = 1,2$ are shares of the countries in the joint fund, C_i, $i = 1,2$ are costs of reducing the emissions in particular countries. In this case a multicriteria optimization problem can be considered in which the expenditures X_1, X_2 and the depositions D_1, D_2 are minimized subject to the constraints described by the cost functions, the deposition equations and the equation defining the joint fund with respect to the variables: the costs C_1, C_2 and the emissions E_1, E_2. Solutions of the problem lay on a Pareto boundary of a simplex S in the four-dimensional objective space. Let S_+ be defined by

$$S_+ = \{(X_1,X_2,D_1,D_2) : (X_1,X_2,D_1,D_2) \leq (\overline{X}_1,\overline{X}_2,\overline{D}_1,\overline{D}_2), (X_1,X_2,D_1,D_2) \in S\}$$

and called an agreement set. If S_+ is not empty, it describes benefits the countries can achieve as an effect of cooperation in comparison to the first case. This is an incentive to cooperation.

The bargaining problem consists of looking for an efficient solution in an agreement set, being subset of the simplex S, of all points dominating the point $d = (\overline{X}_1,\overline{X}_2,\overline{D}_1,\overline{D}_2)$, called further a status quo or a disagreement point. The solution should be selected according to the preferences of the countries considered further as players, and should assure the "fairness" rule. Roughly speaking, the problem consists in proper, agreeable to both the countries allocation of the benefits resulting from the cooperation. The MCBARG system supports analysis of the problem and selection of such a solution.

In the ACID model each of the two players (countries) has two objectives, namely: the expenditures, and the sulfur deposition, to be minimized. Denoting the objectives by

a1, a2 for the first player and by b1, b2 for the second one, respectively, the agreement set has been described by a set of linear inequalities presented in Figure 7. The description has been obtained for assumed forms of the cost functions and given parameters in deposition equations. The parameters have been assumed in such a way that the first player represents developing country with highly polluting technologies and very limited funds for the reducing program, while the second player represents highly developed country, with advanced technologies.

To see how the model is introduced into the system, first select the **EDIT** option and next select in a sequence the options : **PLAYERS**, **CRITERIA** and **FORMULAE** exiting each of them by pressing <F9> key. You will obtain respectively the screens presented in Figures 5 , 6 and 7.

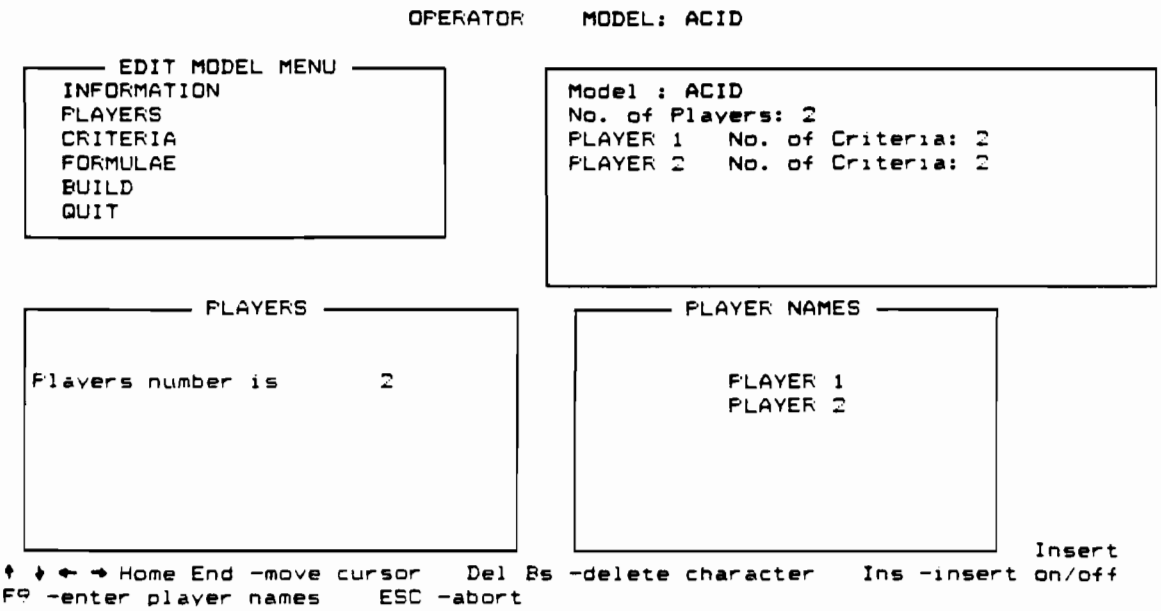


Figure 5

In Figure 5 you can see the number of players (2) and their names (**PLAYER1**, **PLAYER2**). In Figure 6 the names of criteria, their description, status, units and status quo point are given. The criteria names are used in the description of the formulae defining the agreement set, as you can see in Figure 7. In the figure only part of the formualae is presented. To see all of the formulae use scrolling <Up, Down arrows > , <PgDn> and <PgUp> keys.

Leave the model unchanged and quit the **EDIT** option.

Next, select the **BUILD** option of the **MODEL** menu. Wait please until the system has translated the model into the internal form and made the initial calculations.

Next, return to the **MAIN** menu and the start negotiation-mediation procedure by selection of the **NEGOTIATION** option.

Before starting the mediation, the system will ask you to specify the level of players' access to the information about the counterplayers. Press <F1> key (Help) to get detailed information about the possibilities provided by the system. Select the **LEVEL** 2 if you want players to receive all the available information in the system about the actions of their counterplayers.

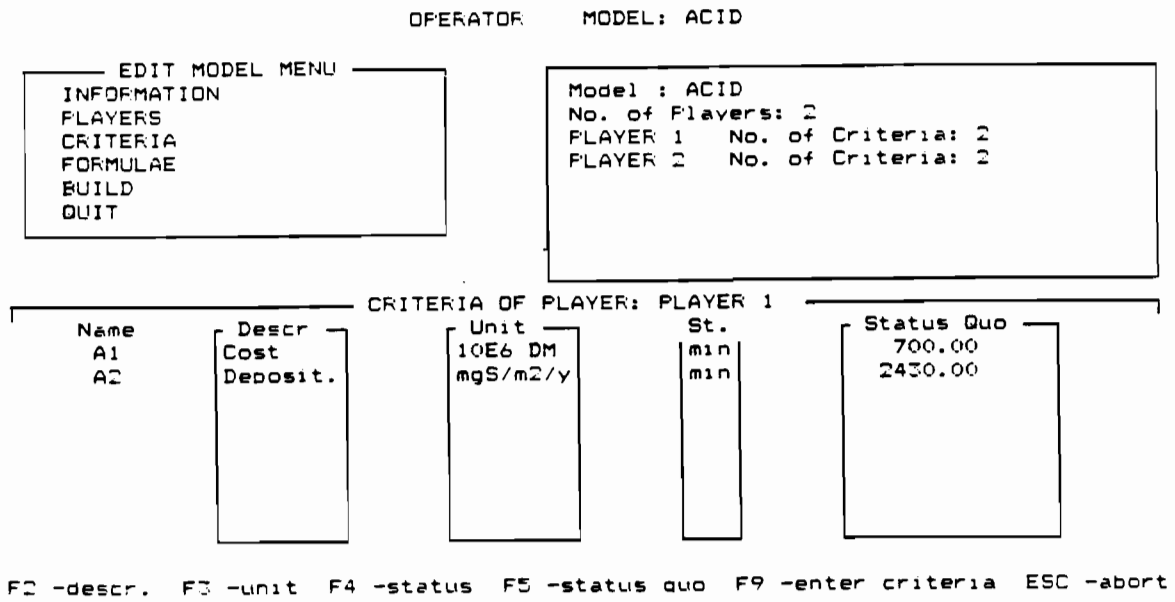


Figure 6

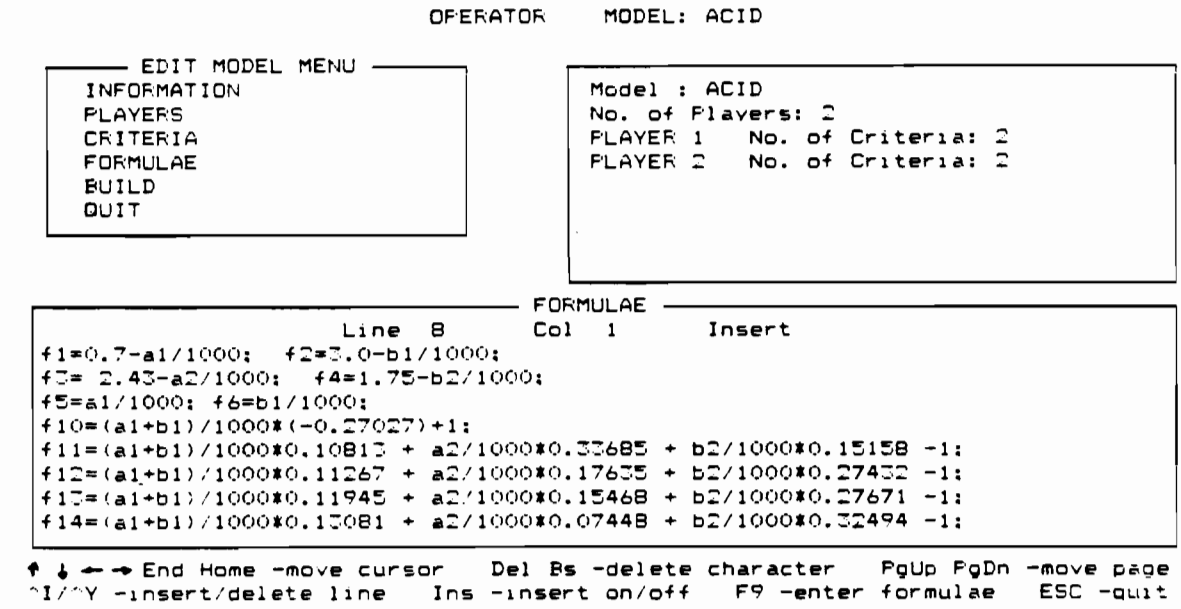


Figure 7

Now your screen will look like Figure 8.

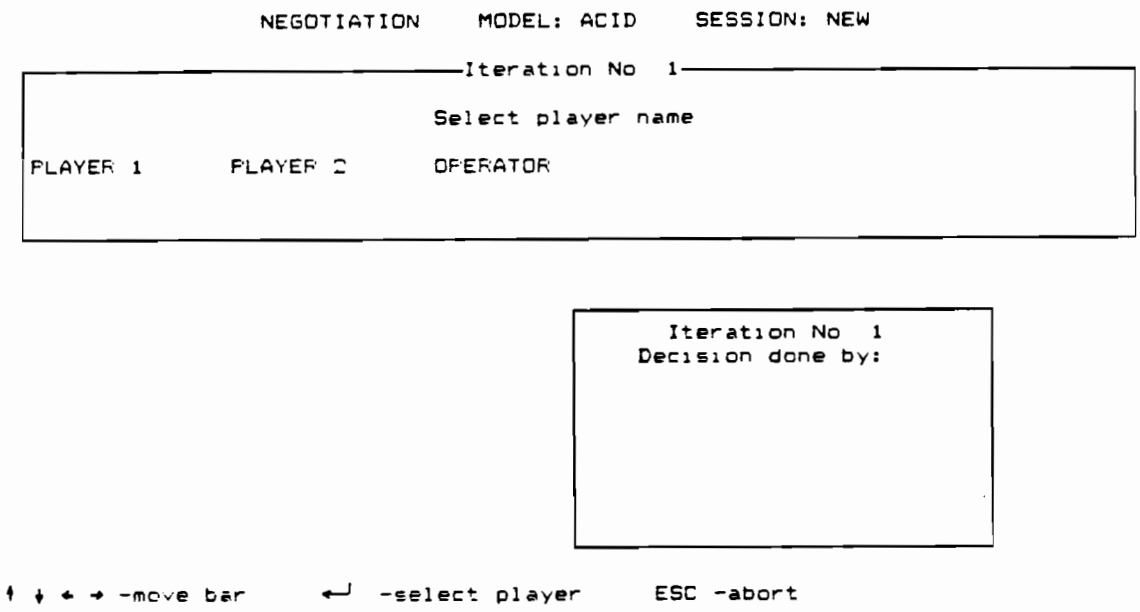


Figure 8

It allows the players to take part in the negotiation procedure. From now on, you can play the role of one of the players or of the **OPERATOR**.

The negotiation procedure is performed in a number of iterations. At each iteration the players make their decisions in a sequence, once per iteration. The **OPERATOR** can observe and review the players results, and save the finished or interrupted session.

To continue the tutorial example select the **PLAYER1**. Type a password. The systems invokes the **NEGOTIATION** menu presented in Figure 9.
Select the **DECISION** option. Within this option you can test the nature of the bargaining problem by generation of a number of anticipated solutions, obtained for different confidence coefficients and reference points specified in your objective space and in the space of the counterplayers.

On the screen presented in Figure 10 you can see the **DECISION** menu in the top line and the first variant of anticipated solution in the table.
It has been calculated for the reference point defined according to the direction from the status quo to the ideal point. The table contains, apart from the anticipated solution (Ant. sol.), the following values: number of variant (Var.), status quo, solution obtained in the previous itertaion (Prev. sol.), reference point (Ref. point), RA-utopia point and ideal point. The status quo, the previous solution and the ideal point are constant within the analysis, while the other values depend on the confidence coefficient, on the reference point and on assumed actions of the counterplayer. The status quo and the previous solution are equal in the first round. The previous solution and the ideal point define reasonable bounds for the new reference points.

Using the **CONF_COEF** option you can modify the confidence coefficient. The lower the value of the coefficient means you have a more limited confidence on the future outcomes of the problem. There will be a greater number of the iterations in the mediation procedure. In our example take the value 0.3.

PLAYER: PLAYER1 MODEL: ACID SESSION: NEW

NEGOTIATION MENU
INFORMATION
DECISION
VIEW
PRINT
QUIT

↑ ↓ Home End -move bar ↵ -select option F1 -help ESC -abort.

Figure 9

INFO CONF_COEF NEW_REF VIEW_CNTR_PL SELECT QUIT

DECISION PLAYER: PLAYER1 MODEL: ACID SESSION: NEW Iter.: 1

Your confidence coefficient: 1.0000

Crit/St.	Var	Status quo	Prev. sol.	Ref. point	Ant. sol.	RA-utopia	Ideal
Cost 10E6 DM [min]	1	700.00	700.00	94.68	315.92	94.68	0.00
Deposit. mgS/m2/y [min]	1	2430.00	2430.00	1187.79	1641.80	1187.79	993.48

← → Home End -move bar ↵ -select option Tab -act. var. ↑ ↓ -scroll var.
^PgUp ^Pgdn -active crit. PgUp PgDn -scroll crit. ^G -graph F1 -help Esc -abort

Figure 10

The **NEW_REF** option allows you to specify reference points and to generate different variants of anticipated solutions, depending on your preferences. Notice that there is one variant already calculated and presented on the screen, so you start from the variant two.

Let us assume that as the first player you prefer to obtain some reasonable decrease of both the expenditures and the sulfur depositions in comparison to the status quo. Select the **NEW_REF** option and type the reference for the cost and deposition (400.00 and 1100.00 respectively). This will be variant 2. The system asks if you accept the default assumptions on the counterplayer decision. (By default the system assumes the decisions of the counterplayers on the base of the previous round, in the case of the first round on base of the direction generated by the ideal point). Press Y (yes) to proceed further. Wait until the system has finished the calculations and has presented the results.

Test the next, third variant. Try to increase the cost and to decrease the deposition. Type the values 700.00 and 1000.00 respectively. Wait for results. The screen will look like Figure 11.

```
 INFO   CONF_COEF   NEW_REF   VIEW_CNTR_PL   SELECT   QUIT

    DECISION    PLAYER: PLAYER1    MODEL: ACID    SESSION: NEW    Iter.: 1

Your confidence coefficient:    0.3000
```

Crit/St.	Var	Status quo	Prev. sol.	Ref. point	Ant. sol.	RA-utopia	Ideal
Cost	1	700.00	700.00	94.68	518.41	94.68	0.00
10E6 DM	2			400.00	609.36	397.85	
[min]	3			700.00	700.00	700.00	
Deposit.	1	2430.00	2430.00	1187.79	2057.34	1187.79	993.48
mgS/m2/y	2			1100.00	2028.14	1090.47	
[min]	3			1000.00	1999.04	993.48	

```
← → Home End -move bar   ↵ -select option   Tab -act. var.   ↑ ↓ -scroll var.
^PgUp ^Pgdn -active crit. PgUp PgDn -scroll crit. ^G -graph F1 -help Esc -abort
```

Figure 11

You can compare the generated variants in a graphical form by pressing <Ctrl G> key. Try it. You see the horizontal bars representing the improvements of the anticipated solution in comparison to the status quo. In the case of the maximized criterion, the bar begins at the left hand side of the screen, in the case of the minimized, at the right hand side.

To analyze how the counterplayer decisions influence your outcomes, generate the variant 4 in the place of the variant 3, typing the same reference values as in the variant 2 but answer N (no) to the system question about the assumptions on the counterplayer decision. You will have a posibility to choose and to type counterplayer reference values, different than the system default values. The anticipated solution will be computed using your values for the counterplayer.

To look at the results of the simulated counterplayer actions, use the **VIEW_CNTR_PL** option. An example of the displayed screen is presented in Figure 12. The results are

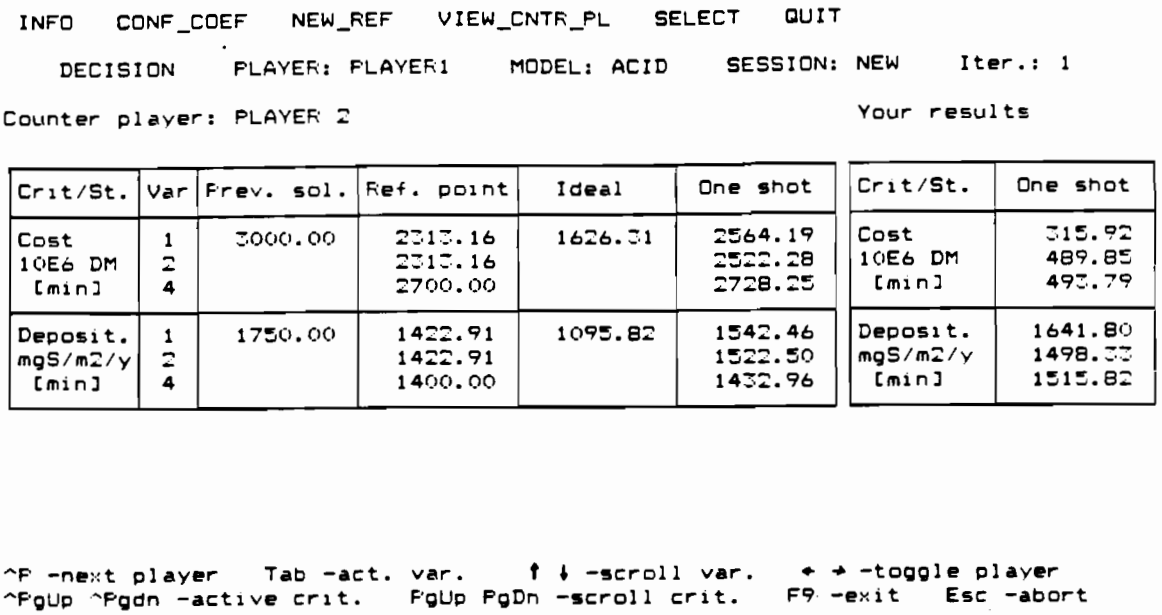


Figure 12

presented in two tables. The table on the left relates to the counterplayer and the one on the right, to your results. You see one-shot solutions. These are solutions that lay on the Pareto frontier of the agreement set.

Return to the **DECISION** menu pressing <F9> key. Now you should make your decision. Indicate the variant which represents your preferences and should be taken as the base in calculating the cooperative solution. Obviously, in a real decision analysis, you will generate a greater number of variants to be sure that the selected variant expresses your preferences. In this tutorial example select the second variant. To do this enter the **SELECT** option and using <Tab> key set the variant 2 to be the active variant (the active variant is highlighted). Press <Enter> key.

In the bottom help lines you can see keys that were not used in the example. Here is their description. The <up, down arrows> scroll the active variants, <Ctrl PgUp, Ctrl PgDn> set the active criterion, and <PgUp, PgDn> scroll the criteria. The scrolling of the variants makes it possible to display on the screen any three of the generated variants. The scrolling of the criteria is useful when the number of criteria of the particular player is greater than 4.

Quit the **DECISION** menu, and quit the **NEGOTIATION** menu. You will be back at the screen presented in Figure 8. Select the **PLAYER2**. Next select the **DECISION** option in the **NEGOTIATION** menu. You will obtain the screen similar to the screen of the **PLAYER1** presented in Figure 9. Perform the interactive analysis in the same way as for the **PLAYER1**.

Assume that the **PLAYER2** has less limited confidence in the future outcomes than the **PLAYER1**. Describe his confidence with the coefficient equal to 0.4.

Next, type the reference values for the objectives of this player. The **PLAYER2**, representing highly developed country is interested mainly in a decreasing of the sulfur deposition. He decides to have the cost closely to the status quo level. Generate two variants for this player. Type the reference of the second variant on the level 2800.0, 1400.0 for the cost and the deposition respectively. This will be the variant 2. Next create the variant 3

with the reference values 3000.0, 1200.0 respectively. In both cases accept the default values for the counterplayer decision. You will obtain the screen presented in Figure 13.

INFO CONF_COEF NEW_REF VIEW_CNTR_PL SELECT QUIT

DECISION PLAYER: PLAYER2 MODEL: ACID SESSION: NEW Iter.: 1

Your confidence coefficient: 0.4000

Crit/St.	Var	Status quo	Prev. sol.	Ref. point	Ant. sol.	RA-utopia	Ideal
Cost 10E6 DM [min]	1 2 3	3000.00	3000.00	2313.16 2800.00 3000.00	2725.26 2882.46 3000.00	2313.16 2706.15 3000.00	1626.31
Deposit. mgS/m2/y [min]	1 2 3	1750.00	1750.00	1422.91 1400.00 1200.00	1619.16 1544.30 1488.33	1422.91 1235.76 1095.82	1095.82

← → Home End -move bar ↵ -select option Tab -act. var. ↑ ↓ -scroll var.
^PgUp ^Pgdn -active crit. PgUp PgDn -scroll crit. ^G -graph F1 -help Esc -abort

Figure 13

Select the second variant as the preferable one. Quit the **DECISION** option and the **NEGOTIATION** menu.

Wait until the system has calculated the cooperative solution of the iteration. Now you are ready to start the next iteration. You see the screen presented in Figure 8.

Select **PLAYER1** and press the <Enter> key. You will invoke the **NEGOTIATION** menu. Before making a decision for the second iteration you can look at the first iteration, results selecting the **VIEW** option. Try it. You will see the screen presented in Figure 14.

Return to the **NEGOTIATION** menu and select the **DECISION** option. Follow the actions of **PLAYER1** presented in Figure 15 and select the variant 3. Note that the confidence coefficient has been assumed on the level of 1.000. Next quit the **DECISION** option and **NEGOTIATION** menu.

Select **PLAYER2**, next the **DECISION** menu and follow the actions of **PLAYER2** presented in Figure 16. Select the variant 3 as the preferable variant, quit the **DECISION** and the **NEGOTIATION** menu.

The system will calculate the final, efficient cooperative solution, proposed to the players as a mediation solution.

Try to play the role of **OPERATOR** to review the session results selecting **VIEW** option in the **OPERATOR** menu, or to print the results using **PRINT** option.

A summary printing of the session ACID2Y.SES is presented in Figure 17. Observe that at the final solution **PLAYER1** has obtained a reasonable decrease, in comparison to the status quo, of both - the cost and the deposition levels, while **PLAYER2** has obtained the cost and the deposition levels relevant to the assumed preferences - large decrease of the deposition level under costs close to the status quo.

The results can be reviewed and printed by any of the players (**VIEW** and **PRINT** options of the **NEGOTIATION** menu.

VIEW PLAYER: PLAYER1 MODEL: ACID SESSION: NEW Iter.: 2

Player: PLAYER 1 Iteration: 1 Session not finished
Confidence coefficients joint: 0.3000 assumed by player: 0.3000

Criteria names	Status	Units	Status quo	Previous agr. point	Agreement point	One-shot solution	Ideal point
Cost	min	10E6 DM	700.00	700.00	609.36	495.29	0.00
Deposit.	min	mgS/m2/y	2430.00	2430.00	2028.14	1522.44	993.48

Criteria	St.	Graphical representation of results
Cost	min	
Deposit.	min	

← → -scroll iter. ^PgUp ^PgDn -act. crit. PgUp PgDn -scroll crit.
^C -counter players F9 -exit ESC -abort

Figure 14

INFO CONF_COEF NEW_REF VIEW_CNTR_PL SELECT QUIT

DECISION PLAYER: PLAYER1 MODEL: ACID SESSION: NEW Iter.: 2

Your confidence coefficient: 1.0000

Crit/St.	Var	Status quo	Prev. sol.	Ref. point	Ant. sol.	RA-utopia	Ideal
Cost	1	700.00	609.36	397.85	490.77	418.40	0.00
10E6 DM	2			500.00	548.33	515.98	
[min]	3			550.00	574.95	557.90	
Deposit.	1	2430.00	2028.14	1090.47	1502.43	1181.59	993.48
mgS/m2/y	2			1000.00	1454.43	1150.27	
[min]	3			1000.00	1432.23	1136.82	

← → Home End -move bar ↵ -select option Tab -act. var. ↑ ↓ -scroll var.
^PgUp ^Pgdn -active crit. PgUp PgDn -scroll crit. ^G -graph F1 -help Esc -abort

Figure 15

INFO CONF_COEF NEW_REF VIEW_CNTR_PL SELECT QUIT

DECISION PLAYER: PLAYER2 MODEL: ACID SESSION: NEW Iter.: 2

Your confidence coefficient: 1.0000

Crit/St.	Var	Status quo	Prev. sol.	Ref. point	Ant. sol.	RA-utopia	Ideal
Cost 10E6 DM [min]	1	3000.00	2911.84	2706.15	2807.88	2738.00	1626.31
	2			2500.00	2726.63	2617.68	
	3			2800.00	2863.41	2833.00	
Deposit. mgS/m2/y [min]	1	1750.00	1595.73	1235.76	1406.80	1291.50	1095.82
	2			1250.00	1440.24	1348.79	
	3			1100.00	1381.03	1246.25	

← → Home End -move bar ↵ -select option Tab -act. var. ↑ ↓ -scroll var.
^PgUp ^Pgdn -active crit. PgUp PgDn -scroll crit. ^G -graph F1 -help Esc -abort

Figure 16

MCBARG MODEL: ACID SESSION: ACID2Y 9/ 7/1990-13: 0

PLAYER: PLAYER1

Criteria names	Cost	Deposit.
Units	10E6 DM	mgS/m2/y
Status	min	min
Status Quo	700.00	2430.00
Ideal Point	0.00	993.48
Agreement point		
at iteration: 1	609.36	2028.14
2	575.35	1439.11

PLAYER: PLAYER2

Criteria names	Cost	Deposit.
Units	10E6 DM	mgS/m2/y
Status	min	min
Status Quo	3000.00	1750.00
Ideal Point	1626.31	1095.82
Agreement point		
at iteration: 1	2911.84	1595.73
2	2859.74	1364.78

Session finished

Figure 17

Quit the OPERATOR menu, not saving the session. The session of this tutorial example is already saved in the ACID2Y.SES file.

Quit the MAIN menu to leave the system.

5 Technical information

The program is distributed on a diskette that should be installed on an IBM PC compatible computer. This diskette contains the following files:

- MCBARG.EXE — binary code of the program,
- INFOSYS.DAT — file of records containing information presented in the program by INFO option,
- HELPSYS.DAT — file of records containing information presented in the program by HELP option,
- ACID.MOD, AGRICO.MOD — files containing the illustrative examples of bargaining problems (description of AGRICO.MOD - see Bronisz, Krus, Lopuch, 1988),
- ACID2Y.SES — file containing the illustrative session.

The program is activated by the command: MCBARG. It automatically recognizes one of the following graphic cards: HERCULES, CGA, EGA.

The program enables a user to create new models (sessions) of bargaining problems and to save the models (sessions) on the active disk. Default extension of the files containing the models is .MOD (the session - .SES). During a session with the system the following auxiliary files are generated on the disk: BACKUP.DDD, BACKUP.SSS. They are erased after a session.

6 References

Bergman L., H. Cesar and G. Klaassen (1990) A Scheme for Sharing the Costs of Reducing Sulfur Emissions in Europe. WP-90-005, IIASA, Laxenburg, Austria.

Bronisz P. and 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 and B. Lopuch (1987) An Experimental System Supporting Multiobjective Bargaining Problem. A Methodological Guide. In A. Lewandowski, A.P. Wierzbicki, eds., Theory, Software and Testing Examples for Decision Support Systems. WP-87-26, IIASA, Laxenburg, Austria, pp. 193–202.

Bronisz P., L. Krus and B. Lopuch (1988) MCBARG. A System Supporting Muliticriteria Bargaining. WP-88-115, IIASA, Laxenburg, Austria.

Bronisz P., L. Krus and A. Wierzbicki (1988). Towards interactive solutions in Bargaining Problem. In A. Lewandowski, A.P. Wierzbicki, eds., Aspiration Based Decision Support Systems. *Lecture Notes in Economics and Mathematical Systems*, Vol. 331, Springer-Verlag, Berlin, pp.251-268.

Fandel G. and 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. and M. Smorodinsky (1975). Other Solutions to Nash's Bargaining Problem. *Econometrica*, Vol. 43, pp. 513–518.

Luce R.D. and H. Raiffa (1957). Games and Decisions: Introduction and Critical Survey, New York: Wiley.

Nash J.F. (1950). The Bargaining Problem. *Econometrica*, Vol. 18, pp. 155–162.

Nash J.F. (1953). Two-Person Cooperative Games. *Econometrica*, Vol. 21, pp. 129–140.

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.

Rogowski T., J.Sobczyk, A.P. Wierzbicki (1987). IAC–DIDAS–L, A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Linear and Dynamic Linear Models on Professional Microcomputers. In A. Lewandowski, A.P. Wierzbicki, eds., Theory, Software and Testing Examples for Decision Support System. WP-87-26, IIASA, Laxenburg, Austria, pp. 106–124.

Roth A.E. (1979a). An Impossibility Result Concerning n-Person Bargaining Games. *International Journal of Game Theory*, Vol. 8, pp. 129–132.

Roth A.E. (1979b). Axiomatic Models of Bargaining. *Lecture Notes in Economics and Mathematical Systems*, Vol. 170, Springer-Verlag, Berlin.

Wierzbicki A.P. (1982). A Mathematical Basis for Satisficing Decision Making. *Mathematical Modelling*, Vol. 3, pp. 391–405.

MPS – Decision Support System for Multiobjective Project Scheduling

R. Słowiński*, B. Soniewicki**, J. Węglarz*
* *Institute of Computing Science, Technical University of Poznań, 60-965 Poznań, Poland*
** *Center for Education and Development in Agriculture, 61-659 Poznań, Poland*

Abstract

The report presents a decision support system (DSS) for multiobjective project scheduling under multiple-category resource constraints. It handles quite a general class of nonpreemptive scheduling problems with renewable, nonrenewable and doubly-constrained resources, multiple performing modes of activities, precedence constraints in the form of an activity network and multiple project performance criteria of time and cost type. The DSS has been implemented on a microcomputer compatible with IBM PC, and called MPS. It is based in three kinds of heuristics: parallel priority rules, simulated annealing and branch-and-bound. The last algorithm can even yield exact solutions when sufficient processing time is available. Some parts of the MPS are interactive, in particular, the search for a best compromise schedule. Graphical facilities enable a thorough evaluation of feasible schedules. The report starts with a methodological guide presenting the problem formulation and the three heuristics. Then, the general scheme of the MPS is given together with an executive guide. An expanding menu and all its options are described and illustrated with a simple example. The last part presents a real problem solving consisting in scheduling 40 farm activities.

1 INTRODUCTION

Simply stated, the problem adressed is how to schedule precedence and resource-constrained activities of a project in order to accomplish a given managerial objective. Over the past twenty five years, a number of techniques have been developed to help project managers answer this question, the applicability of each technique being a function of project characteristics and the managerial objective (see, for example, [1, 2, 3, 4]).

In [5], a scheduling technique was proposed that is capable of heuristically or optimally solving most of the nonpreemptive forms of project scheduling problems previously examined in the literature. This includes time-based, time-cost trade-off, time-resource trade-off, and resource constrained projects. In addition, the proposed algorithm permits the scheduling of activities where activity performance can increase as well as decrease with the availability of resources such as cash, and where resource-duration interactions exist. The algorithm is a branch-and-bound procedure of the backtracking variety.

In practice, however, project scheduling problems usually should involve multiple objectives. It follows mainly from joint consideration of resources submitted to two different

kinds of availability constraints: (i) on the amount available at every moment of project duration, and (ii) on the total consumption over a given time period. Depending on what constraint is imposed, we we have resources of different categories: renewable (e.g. machines, manpower, equipment) if only (i) is imposed, nonrenewable (e.g. money, fuel, raw materials) if only (ii) is imposed, and doubly constrained (e.g. power, rate of investment, fuel flow) if both (i) and (ii) are imposed. The utilization of resources submitted to constraint (i) is usually concerned with time criteria (e.g. minimization of machine idle time is equivalent to minimization of project duration) while the consumption of resources submitted to constraint (ii) is measured by cost criteria. Thus, joint consideration of multiple-category resources involves time and cost criteria which are in general conflicting.

The above facts have motivated a construction of a DSS for multiobjective project scheduling, called MPS. Its general scheme was introduced in [6, 7, 13]. In the next section, we present a methodological guide of MPS, with a general problem formulation, the structure of the system and the algorithms used in the calculation phase. Then, in section 3, an executive guide of MPS is given with a detailed description of all the options from an expanding menu. The final section presents the steps of solving a real scheduling problem consisting of 40 agricultural activities.

2 METHODOLOGICAL GUIDE

2.1 Problem formulation

It is assumed that all time characteristics of the project are integer and that the time horizon T_h is divided into periods of unit length, t=1,...,T_h .

The project is characterized by four components: set of resources R, set of activities A, precedence constraints in set A, and set Q of project performance measures (objectives, criteria).

Set R is composed of:

- p types of renewable resources $R_1^r, ..., R_p^r$ with the usage limited to N_{kt}^r units in every period $t(k = 1, ..., p; t = 1, ..., T_h)$,
- v types of nonrenewable resources $R_1^n, ..., R_v^n$ with the consumption limited to B_k^n units (k=1,...,v),
- u types of doubly constrained resources $R_1^d, ..., R_u^d$ with consumption limited to B_k^d units and usage to N_{kt}^d units $(k = 1, ..., u; t = 1, ..., T_h)$.

Set A is composed of n activities which have discrete resource requirements. For each activity $A_j \in A$ we have in general w performing modes, i.e. feasible assignments of resource amounts to this activity. Performing mode m of activity A_j is defined by the vector $r_{mj} = [r_{mj1}^r ... r_{mjp}^r \ r_{mj1}^n ... r_{mjv}^n \ r_{mj1}^d ... r_{mju}^d]$ whose elements determine the usage of renewable and doubly constrained resources and consumption per period (constant for every activity) of nonrenewable resources (m=1,...,w; j=1,...,n). For each mode m the duration of A_j is known: $D_{mj}(m = 1, ..., w; j = 1, ..., n)$. Ready time a_j and due date d_j are also specified for each A_j (j=1,...,n).

The precedence constraints are represented by a directed acyclic graph where activities are represented by integer labeled nodes, such that the label of a node is always higher than the labels of all its predecessor nodes. Arrows represent precedence relationships

between activities. A unique ending dummy activity A_{n+1} with zero duration and resource requirement is appended to the graph.

The set Q of project performance measures is composed of following time and cost criteria:

- project completion time: CT,
- smoothness of the resource profile: $S_k^r \quad (k = 1, ..., p)$ and $S_k^d \quad (k = 1, ..., u)$,
- mean weighted lateness: MWL,
- total number of tardy activities: TNTA,
- mean weighted flow time: MWFT,
- total resource consumption: $TC_k^r \quad (k = 1, ..., v)$ and $TC_k^d \quad (k = 1, ..., u)$,
- weighted resource consumption: WC,
- net present value: NPV.

Scheduling of the project consists in such an allocation (considered in time) of resources from set R to activities from set A that all activities are completed, the constraints are satisfied and the best compromise between criteria from set Q is reached.

The above model of multicriteria project scheduling has been considered first for the preemptive case [8, 9]. The nonpreemptive case has been investigated in [10, 11] and [12, 13] where an interactive procedure for multiobjective project scheduling has been proposed. It is using the implicit enumeration algorithm by Patterson at al. [5] in the calculation phase. The mathematical programming formulation of this problem has been given in [13]. It is based on the following definition of the zero-one decision variable:

$$x_{jmt} = \begin{cases} 1 & \textit{if activity } A_j \textit{ performed under mode } m \\ & \textit{is completed in period } t, \\ 0 & \textit{otherwise} \end{cases}$$

(j=1,...,n; m=1,...,w; t=1,...,T_h). Associated with each activity A_j are its critical-path determined early finish completion period e_j and late finish completion period l_j . Both e_j and l_j are calculated in the usual way but using the set of minimum duration modes for all activities. When determining late finish completion periods, l_{n+1} is set equal to T_h . Of course, $x_{jmt} \equiv 0$ for $t < e_j$ and $t > l_j$. Let P_j be the set of all immediate predecessors of activity $A_j \quad (j = 1, ..., n)$. Now, the multiobjective project scheduling problem can be formulated as the following zero-one MOLP problem:

$$\min CT = \sum_{m=1}^{w} \sum_{t=e_{n+1}}^{l_{n+1}} t x_{(n+1)mt} \tag{1}$$

$$\min S_k^r = \max_{1 \le t \le T_h} \left\{ \sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{q=t}^{t+D_{mj}-1} r_{mjk}^r x_{jmq} \right\} \quad k = 1, \ldots \tag{2}$$

$$\min S_k^d = \max_{1 \le t \le T_h} \left\{ \sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{q=t}^{t+D_{mj}-1} r_{mjk}^d x_{jmq} \right\} \quad k = 1, \ldots \tag{3}$$

$$\min MWL = (1/n) \sum_{j=1}^{n} v_j \sum_{m=1}^{w} \sum_{t=d_j+1}^{l_j} (t - d_j) x_{jmt} \tag{4}$$

$$\min TNTA = \sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{t=d_j+1}^{l_j} x_{jmt} \tag{5}$$

$$\min MWFT = (1/n) \sum_{j=1}^{n} v_j \sum_{m=1}^{w} \sum_{t=e_j}^{l_j} (t x_{jmt} - a_j) \tag{6}$$

$$\min TC_k^n = \sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{t=e_j}^{l_j} r_{mjk}^n \, x_{jmt} \quad k = 1, \ldots, v \tag{7}$$

$$\min TC_k^d = \sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{t=e_j}^{l_j} r_{mjk}^d \, D_{jm} \, x_{jmt} \quad k = 1, \ldots, u \tag{8}$$

$$\min WC = \sum_{k=1}^{v} c_k^n \, TC_k^n + \sum_{k=1}^{u} c_k^d \, TC_k^d \tag{9}$$

where c_k^n and c_k^d are unitary costs of R_k^n and R_k^d, respectively,

$$\max NPV = \sum_{t=1}^{T_h} \alpha_t \left\{ \sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{q=t}^{t+D_{mj}-1} x_{jmq} r_{mjk}^n / D_{mj} - \sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{q=t-1}^{t+D_{mj}-2} x_{jmq} r_{mjk}^n / D_{mj} \right\} \tag{10}$$

where R_k^n is cash, $\alpha_t = (1 - \delta)^{1-t}$ is a discount factor and δ is a discount rate,
subject to the constraints:

- on the performance of each activity using one mode only

$$\sum_{m=1}^{w} \sum_{t=e_j}^{l_j} x_{jmt} = 1 \quad j = 1, \ldots, n \tag{11}$$

- on the precedence

$$\sum_{m=1}^{w} \sum_{t=e_j}^{l_j} (t - D_{mj}) x_{jmt} - \sum_{m=1}^{w} \sum_{t=e_f}^{l_f} t x_{fmt} \geq 0 \quad j = 1, \ldots, n+1, \quad \bigvee f \tag{12}$$

- on resource availability

  - renewable

$$\sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{q=t}^{t+D_{mj}-1} r_{mjk}^r x_{jmq} \leq N_{kt}^r \quad k = 1 \ldots, p; \quad t = 1, \ldots, T_h \tag{13}$$

  - nonrenewable

$$\sum_{j=1}^{n} \sum_{m=1}^{w} \sum_{t=e_j}^{l_j} r_{mjk}^n x_{jmt} \leq B_k^n \quad k = 1, \ldots, v \tag{14}$$

  - doubly constrained

$$\sum_{j=1}^{n}\sum_{m=1}^{w}\sum_{q=t}^{t+D_{mj}-1} r^d_{mjk}x_{jmq} \le N^d_{kt} \quad k=1\ldots,u; \quad t=1,\ldots,T_h \tag{15}$$

$$\sum_{j=1}^{n}\sum_{m=1}^{w}\sum_{t=e_j}^{l_j} r^d_{mjk}D_{mj}x_{jmt} \le B^d_k \quad k=1,\ldots,u \tag{16}$$

Even for a single criterion, the above problems are NP-hard and need implicit enumeration algorithms which are rather not very efficient. The interactive procedure [13] proposed for solving the multiobjective problem makes an aggregation of objectives in a scalarizing function of the augmented weighted Chebyshev form. Generation of compromise solutions in successive iterations needs solving as many complex optimization problems. If one would like to solve them optimally, the waiting time for a next proposal would be very long and perhaps unacceptable for the decision maker (DM). For this reason, when constructing the DSS for the most general case of project scheduling, we decided to use heuristic algorithms.

2.2 Structure of the MPS

The MPS is composed of four modules:

- model editor,
- heuristic algorithms for single-objective project scheduling,
- interactive procedure for multiobjective project scheduling,
- display of results.

Model editor permits to enter and modify the data required to set up the model of a particular project scheduling problem. Its functional description will be made in the next section.

Three kinds of heuristic algorithms for single-objective project scheduling are used:

- parallel priority heuristics,
- simulated annealing,
- branch-and-bound type heuristics.

2.3 Priority heuristics

The idea of the parallel priority heuristics can be summarized in the following steps (cf.[14]):

Step 1. Using a priority function create a list of activities. Set the current time equal to zero.

Step 2. Choose the first activity from the list which is ready to be performed, i.e. such that all its predecessors are completed in the current time. If the amount of available resources is sufficient to cover the requirements of the first performing mode of the chosen activity, then allot the required amount to this activity. Otherwise try another performing mode and if it also fails, increase the current time by one unit and retry the operation. Remove the scheduled activity from the list.

Step 3. Augment the current time to the earliest period of the completion of one of the scheduled activities.

Step 4. If the list is empty then stop, otherwise go to step 2.

In MPS, twelve priority functions take into account time or/and resource characteristics of the project:

1. $p_j = 1/LFT_{1j}$, where LFT_{1j} is the latest completion time of activity A_j, following from the critical path analysis for the first performing modes of activities[1].
2. $p_j = 1/LST_{1j}$, where LST_{1j} is the latest starting time of A_j.
3. $p_j = 1/EST_{1j}$, where EST_{1j} is the earliest starting time of A_j.
4. $p_j = 1/(EST_{1j} + d_j)$, where d_j is the due date of A_j .
5. $p_j = 1/d_j$.
6. $p_j = 1/(a_j + d_j)$, where a_j is the ready time of A_j .
7. $p_j = 1/a_j$.
8. $p_j = D_{1j}$, where D_{1j} is the duration of A_j performed under mode 1.
9. $p_j = D_{1j} + \sum_{i \in S_j} D_{1i}$, where S_j is the set of predecessors of A_j .
10. $p_j = r^r_{1j1}$, where r^r_{1j1} is the requirement for renewable resource R^r_1 of A_j performed under mode 1.
11. $p_j = \sum_{k=1}^{v} r^n_{1jk}$, where r^n_{1jk} is the requirement for nonrenewable resource R^n_k of A_j performed under mode 1 (k=1,...,v).
12. $p_j = R_j + \sum_{i \in S_j} R_i$,
    where activities start in their EST_{1j}, $R_j = \sum_{k=1}^{p} (r^r_{1jk}/N^r_k)(V_k/V_{max})D_{1j}$, $V_k = U_k + E_k N^r_k$, $V_{max} = \max_k(V)$, U_k is the latest time period in which there is a non-zero requirement for resource R^r_k, E_k is the total excess requirement for resource R^r_k over the availability N^r_k.

Moreover, in order to obtain the greatest possible variety of feasible schedules, for each priority heuristic 5 mutations are generated for a priority list corresponding to a particular priority function. The mutations are generated in such a way that the next activity to be scheduled is taken either from the top of the original list, or is randomly chosen from first 2, or 3, or 4, or 5 activities. In this way, we can obtain up to 60 different feasible schedules evaluated from the viewpoint of the chosen criteria.

Additionally, we use a heuristic procedure specialized in smoothing resource profiles. It is based on the algorithm proposed by Harris [15] for the case of a single renewable resource and one performing mode per activity. For a given project completion time, it tends to minimize a "turning moment" of the resource profile. It works in the following steps:

[1]The activity performing modes are ordered according to increasing duration when time criteria are considered, and according to increasing cost when cost criteria are considered.

Step 1. Assume mode 1 for each activity and calculate their EST_{1j} (j=1,...,n) ignoring all resource constraints.

Step 2. Create a subset of activities without predecessors in the current set.

Step 3. For each activity from the subset, calculate a coefficient of improvement after delaying it by all possible (integer) lapses of time under all possible performing modes. Select an activity with the best positive coefficient, update its starting time and performing mode, and remove it from the subset and the current set. If there is no activity with a positive coefficient then remove all the activities.

Step 4. Repeat steps 2 and 3 until the current set becomes empty.

Step 5. If no improvement was made in the last iteration then stop, otherwise return to step 2 for pushing activities in an opposite direction (predecessors have to be replaced by successors and vice versa).

2.4 Simulated annealing

The simulated annealing procedure starts the search from the best solution obtained using parallel priority heuristics. Instead of a simple local search algorithm, simulated annealing attempts to avoid becoming trapped in a local optimum by sometimes accepting a neighbourhood move which increases the value of the objective z to be minimized (cf.[16, 17]. The acceptance or rejection of an uphill move is determined by a sequence of random numbers, but with a controlled probability. The probability of accepting a move which causes an increase d of z is called the acceptance function and is normally set to $exp(-\delta/T)$ where T is a control parameter which corresponds to temperature in the analogy with physical annealing. This acceptance function implies that small increases in z are more likely to be accepted than large increases, and that when T is high most moves will be accepted, but as T approaches zero most uphill moves will be rejected. So in simulated annealing, the algorithm is started with a relatively high value of T, to avoid being prematurely trapped in a local optimum. The algorithm proceeds by attemping a certain number of neighbourhood moves at each temperature, while the temperature parameter is gradually dropped.

In order to apply the simulated annealing one has to define the neighbourhood of any solution and an efficient method of moving from one solution to its neighbourhood solution with a simultaneous calculation of a new value of the objective. In the case of project scheduling, the neighbourhood is defined in the following way. Two activities from among those which are not precedence-related are randomly chosen. Then, those two activities are permuted on the priority list corresponding to the current solution. The new list is used by the parallel heuristic to construct the neighbourhood solution.

The simulated annealing for project scheduling is organized in the following steps:

Step 1. Take a starting solution, i.e. the best solution obtained using parallel priority heuristics.

Step 2. Set the value of the control parameter (temperature):

$$T = \Delta / ln(x_0^{-1})$$

where Δ is an acceptable deterioration of the score on considered objective z in relation to the starting solution (e.g.20%), x_0 is an acceptance coefficient close to one (e.g.0.9).

Step 3. Construct a neighbourhood solution to the starting one.

Step 4. If the value of z has improved, then the probability of accepting the neighbourhood solution as a new starting solution is equal to one; otherwise calculate the probability as $P = exp(-\delta/T)$, where δ is an actual deterioration of the score on z.

Step 5. Generate a random number from the interval [0,1] according to the uniform distribution. If this number is less than or equal to P then accept the solution constructed in step 3 as a new starting point; otherwise don't change the starting solution.

Step 6. Repeat steps 3, 4 and 5 until the number of new starting solutions attains a given constant which corresponds to an approximate equilibrium.

Step 7. Set the control parameter $T = 0.9 \times T$. Repeat steps 3 to 6 until there is no improvement of z for three consecutive values of T. Then STOP.

2.5 Branch and bound algorithm

The branch and bound algorithm uses the strategy of the depth-first-search and starts backtracking from the best solution obtained using parallel priority heuristics. The search tree consists of all precedence-feasible permutations of activities combined with their performing modes. The details of the algorithm for time and cost criteria are presented in [5]. The results of a computational experiment with the algorithm are presented in [18]. The search process can be arbitrarily interrupted (then resumed if desired) and a current best schedule can be displayed.

2.6 Multicriteria scheduling

In the case of multicriteria scheduling, the interactive search for the best compromise solution is organized on the set of nondominated schedules obtained using parallel priority heuristics and simulated annealing, possibly improved using the branch and bound algorithm.

The interactive procedure is organized in three steps repeated iteratively (cf.[7, 13]):

Step 1. (Starting step). Construct the $k \times k$ pay-off table Z on the set of nondominated schedules. Elements z_{ij} of Z are values of criterion i for the best schedule from the viewpoint of criterion j. The diagonal of Z defines an "ideal" solution which is unfeasible in general. The vector composed of the worst scores z_{ij} on particular criteria is a "nadir" solution.

Step 2. (Calculation step). Find the nondominated schedule being the closest to the "ideal" one in the sense of the scalarizing function (augmented weighted Chebyshev norm).

Step 3. (Decision step). Present the schedule found in step 2 to the decision maker (DM). If he finds it satisfactory on all criteria then STOP; otherwise ask the DM to specify a satisfactory criterion to be relaxed and amount of the relaxation in order to gain on other criteria. The relaxation is translated into a penalty function appended to the scalarizing function and the algorithm returns to step 2.

The scalarizing function used in the calculation step has the following form:

$$s(\underline{z},\underline{\pi}) = \max_{1\le i\le k}(\pi_i(z_i - u_i)) + \epsilon\sum_{i=1}^{k}\pi_i(z_i - u_i)$$

where

$$\pi_i = \Phi/\sum_{i=1}^{k}\Phi_i, \quad \Phi_i = (U_i - u_i)/U_i, \quad i = 1,\ldots,k,$$

$$\epsilon = 0.001, \quad u_i = \min_{\underline{z}\in N}(z_i), \quad U_i = \max_{\underline{z}\in N}(z_i), \quad i = 1,\ldots,k,$$

N is the set of nondominated schedules.

Relaxation Δz_i determined in the decision step is translated to the penalty function appended to the scalarizing function:

$$S(\underline{z}) = s(\underline{z},\underline{\pi}) + b_j(\Delta\underline{z},\underline{\pi}),$$

$$b(\Delta\underline{z},\underline{\pi}) = 1/2\sum_{i=1}^{k}(\max(0,\pi_i(z_i - z_i^{j-1} - \Delta z_i)))^2$$

where j is the iteration index and $\underline{z}^{j-1} \in N$ is the schedule found in previous iteration. Morever, the coefficient π_i of the relaxed criterion is reduced to zero in iteration j.

In MPS, the DM has yet two other possibilities of scanning the set of nondominated schedules when looking for the best compromise one. The first consists in moving from one schedule to the next one from the list of nondominated schedules (option: "another solution"). The second possibility consists in moving from one schedule to the next improved schedule from the viewpoint of an indicated criterion (option: "next improved solution").

3 USER'S MANUAL

3.1 Installing the MPS

MPS runs on a PC compatible with IBM having the following minimal configuration :

- RAM 640 kb,
- MS-DOS ver. 3.10 or higher,
- one floppy disk drive or, preferably, a hard disk.

The MPS diskette or the MPS directory on the hard disk must contain the following files :

- MPS.EXE a compiled system
- 4x6.fon fonts for CGA card
- 8x8.fon fonts for EGA or VGA card
- 14x9.fon fonts for HERCULES card

MPS has been written in TURBO-PASCAL 5.0 with Borland Graphics Toolbox 4.0. Program MPS must be compiled for a graphic card available on a user's computer.

3.2 Menu of the system

MPS starts when you write on the computer monitor its name and press the key ENTER. Then, the invitation screen appears (cf. Fig. 2 in the Appendix). If you want to go to the top of the menu, than press any key. From this state you can choose one of three options :

- DATA HANDLING,
- HEURISTICS,
- EXIT.

To choose an option use vertical arrows which move a highlighted field. In order to accept a highlighted option press key ENTER. Then you go to the next step of the menu.

After choosing DATA HANDLING option three next options appear:

- NEW PROJECT,
- MODIFICATIONS,
- EXIT.

Fig. 3. shows a copy of the screen after choosing the DATA HANDLING option.

Option HEURISTICS offers the following possibilities in the next step of the menu :

- SINGLE OBJECTIVE (optimisation),
- MULTIOBJECTIVE (optimisation),
- EXIT.

Next step of the menu gives the choice of the following approximation algorithms to be used in calculations :

- PRIORITY PROCEDURES,
- SIMULATED ANNEALING,
- BAB TYPE PROCEDURE,
- EXIT.

The last step of this expanding menu permit to choose one criterion in the case of single objective optimisation or two to seven criteria in the case of multiobjective optimisation. The seven criteria are the following :

- COMPLETION TIME,
- RESOURCE SMOOTHING,
- MEAN WEIGHTED ACTIVITY LATENESS,
- TOTAL NUMBER OF TARDY ACTIVITIES,
- MEAN WEIGHTED FLOW TIME,

- NET PRESENT VALUE,
- WEIGHTED RESOURCRE CONSUMPTION,
- EXIT.

Fig. 4. presents the full MPS menu with chosen options being highlighted. To accept his choice, the user must press the key ENTER. The system then goes automatically to the calculation phase.

3.3 Data operating

3.3.1 The new project building

In order to introduce a new project scheduling problem, the user must choose the option DATA HANDLING and next NEW PROJECT.

First, MPS reads precedence constraints among activities, i.e. relation of partial ordering in the set of project activities. It is represented in a form of lists of immediate predecessors of particular activities. The following conditions have to be fulfilled :

1. Project cannot have more than 100 activities.
2. The name of the activity must be unique. If not, MPS will write a message.
3. The name of a predecessor of the activity must be previously defined as a name of the activity. Activities without predecessors must be defined first.
4. Because of an "activity on arc" convention for a graphical representation of the precedence constraints, activities going out from a vertex must have the same list of predecessors. If the user wants a dummy activity, he must add it himself.

The name of the activity must be written on the highlighted field preceded by the text "Name of the activity". This name has not be longer then 20 letters. Then, one can press a vertical arrow key and move to the bottom where the list of predecessors is composed. In order to pass to the next activity one must press key F2, and in order to go back to the previous activity one must press F3. Fig. 5. displays an example of the screen in the stage of reading precedence constraints for a project. When the last activity has been entered, press F4 to go to read global data of the project. The global data are composed of :

1. Due date for the project (in time units).
2. How many types of renewable resources are in the project (max 4).
3. Name of a renewable resource. The length of this name should have no more then 10 characters. The user can use only small letters.
4. Number of available units of ... (here name of a renewable resource).
5. How many types of nonrenewable resources are in the project (maximally 3).
6. Name of a nonrenewable resource. The length of this name should have no more then 10 characters (small letters only).
7. Number of available units of ... (here name of a nonrenewable resource).

8. Inflow(+) or outflow(-) of ... (name of nonrenewable resource) at the end of the project.

9. Interest rate (in percents). This rate refers in principle to u money as a nonrenewable resource.

It should be specified that MPS treats a doublyconstrained resource as a simultaneously renewable and nonrenewable. For example, if money would be a doubly constrained resource, then from the viewpoint of its total consumption it is considered as a nonrenewable resource, while from the viewpoint of the intensity of its consumption in particular time periods, it is considered as a renewable resource.

The user must insert the data in the highlighted fields which can be moved by pressing the vertical arrow keys. After pressing ENTER, MPS goes to the next phase - phase of reading local data for each particular project activity. There are then three further possibilities :

1. After pressing keys Shift and E, option DATA HANDLING is terminated and data memorized in the file.

2. After pressing keys Shift and Q, option DATA HANDLIND is terminated without changing data in the file.

3. After pressing keys Shift and N the program goes to the activity with a specified number.

In Fig. 6, one can see a copy of the screen in the phase of reading global data.

In the phase of reading local data, specific for each activity, MPS reads the following parameters :

1. Release time.

2. Due date.

3. Number of the parralell activity before which the currently displayed activity cannot start - (Name of the current activity) cannot start before activity ...

4. Weighting factor.

5. Number of the mode (maximally 3).

6. Activity duration for mode (number).

7. Requirement of renewable resource (name) per unit duration for mode (number).

8. Requirement of nonrenewable resource (name) per unit duration, for mode (number).

9. Inflow(+) or outflow(-) of (name of a nonrenewable resource) per unit duration, for mode (number).

After entering data for a particular activity, the user must press ENTER to go to the next activity. In order to go to the activity with another number, the user must press simultaneously Shift and N , and then give its number. The stage DATA HANDLING terminates after pressing Shift and E (the data file is saved) or pressing Shift and Q (the data file is not saved).

If the user arrives to the last activity and accepts the inserted data by pressing the key ENTER, all data are saved automatically in a file.

3.3.2 Modification of data

MODIFICATINS of data are similiar to data entering for a new project. At the beginning, the user must choose the name of the file with the data to be modified. When going through three phases described in 3.3.1., the user can modify all the parameters of the project.

In the first phase he can :

- add an activity at the end of the list of activities,
- delete the activity by inserting spaces instead of its name,
- delete one of predecessors of the activity by inserting spaces instead of its name.
- add a predecessor to an activity.

In the two next phases, the user can modify project and activity parameters. It can be done in the same manner as in the case of building a new project.

3.4 Calculation

The project scheduling problem characterized in an activated data file can be solved using one of the three algorithms described in section 2 :

- priority heuristics,
- simulated annealing,
- branch and bound (BAB).

The same selection of algorithms is offered for single and multiobjective optimization. The priority heuristics give up to 60+p feasible schedules (p is the number of renewable resource types). In the single objective case, the best schedule from the viewpoint of a chosen criterion is adopted, while in the multiobjective case, a subset of nondominated schedules is identified and on iteractive search over this subset is organized by the system.

Simulated annealing is activated for each selected criterion (cf. Fig. 8). In the multiobjective case, each generated schedule is tested for dominance and it is saved when nondominated (cf. Fig. 12). This algorithm is especially worthwhile when the user is interested in getting a large variety of nondominated schedules for the interactive phase.

Branch and bound algorithm (BAB) improves nondominated schedules obtained using priority heuristics. At its output, one get either the best improved solution from the viewpoint of a single criterion or a set of nondominated schedules from the viewpoint of multiple criteria. The user can break every procedure by pressing any key during its run. In the case of BAB, MPS displays after the break the current best schedule and asks the user about the continuation of the calculation. After pressing key "Y". MPS continues calculation with BAB, key "N" terminates the procedure.

In the multiobjective case, the number of nondominated schedules generated and saved in the calculation phase is displayed (NNDS=...) in the upper right corner of the screen (cf. Figs. 12-15).

3.5 Display of results

The feasible schedules submitted for evaluation of the user are displayed in a graphic and table form. An axample of such a display is shown in Figs. 10, 16. Gantt charts and resource profiles are shown in four windows of the screen. A highlighted window is an active one and can be either enlarged on the whole screen (cf. Fig. 9) or used to display another picture from an offered variety, or a table schedule specifying start and end times of activities, their modes and resource requirements in time. The user can change the active window by pressing arrow keys. The screen can be copied on the printer by pressing key H (HARDCOPY). When pressing key O, the user gets a new menu offering the possibility of changing the contents of the active window. Key Q is used to abort the current stage and go to the main menu.

In the multiobjective case, there are three possibilities of scanning the set of nondominated schedules (cf. p. 2.6) :

1. To see the next nondominated schedule from the generated set (key A).
2. To see a nondominated schedule with the minimal improvement on the criterion indicated by the cursor (key N).
3. To relax one criterion indicated by the cursor and search for an improved schedule. A value of the relaxation is showed in a lower line.

An error of filling the cursor field can be erased after pressing key D. Position of the cursor can be moved using the keys of horizontal arrows. The user can view the current schedule after pressing key V. In Figs. 13, 14 and 15, examples of displays in the interactive phase of the multicriteria optimisation are shown.

3.6 Example of scheduling

In this section, we shall illustrate the functioning of MPS on an example of scheduling 40 farm operations subject to precedence and resource constraints (cf.[6]). The list of operations is given in Table 1.

1.	harrowing wheat	21.	tedding grassland
2.	harrowing rape	22.	raking grassland
3.	sowing rape	23.	croopping grassland
4.	croopping harv lupin	24.	desication potato
5.	drying lupine	25.	croopping sugar beet
6.	croopping of strorer	26.	fertilisation lupine
7.	cutting down lucerne	27.	croopping fodder bee
8.	cutting down grass	28.	fertilization wheat
9.	tedding grassland	29.	ploughing
10.	raking grassland	30.	skimming lupine
11.	croopping grassland	31.	harrowing lupine
12.	desication potato	32.	harrowing wheat
13.	harrowing wheat	33.	sowing wheat
14.	harrowing rape	34.	croopping sugar beet
15.	sowing rape	35.	croopping potato
16.	croopping harv lupin	36.	harrowing potato
17.	drying lupine	37.	ploughing
18.	croopping of strorer	38.	sowing corn
19.	cutting down lucerne	39.	fertilization fodder
20.	cutting down grass	40.	ploughing corn

Table 1. List of activities or the agricultural project

There are two types of renewable resources: manpower and tractors, available in 200 and 150 units, respectively. The only nonrenewable resource, money, is available in 20000 units at the beginning of the project. For majority of project activities there are specified three performing modes differing by resource requirements and duration. Ready times and due dates of activities follow from an agricultural calendar. Those data are presented in Table 2. A graphical representation of precedence constraints in the set of activities is given in Fig.1.

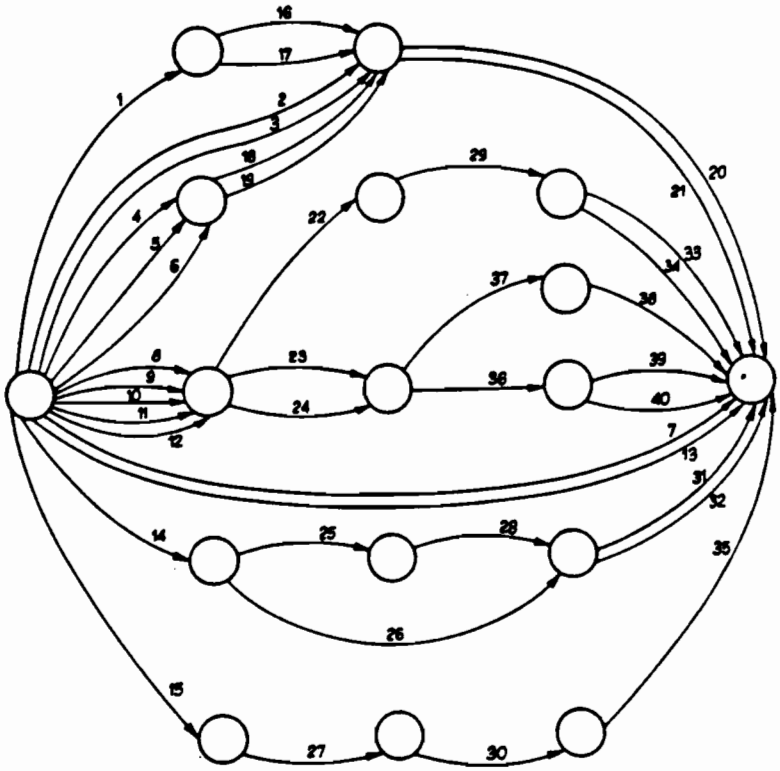


Fig. 1. Precedence constraints in the set of agricultural activities.

Table 2. Characteristics of agricultural activities

Lp	d_{i1}	d_{i2}	d_{i3}	α_i	δ_i	R^r_{i11}	R^r_{i12}	R^r_{i13}	R^n_{i11}	R^n_{i12}	R^n_{i13}
						R^r_{i21}	R^r_{i22}	R^r_{i23}			
1	8	10	16	0	28	4	3	2	80	50	30
						4	3	2			
2	8	10	16	0	28	13	11	7	150	140	110
						13	11	7			
3	9	12	18	0	48	20	15	10	200	170	150
						11	9	6			
4	11	15	22	0	28	25	20	13	230	200	170
						10	8	5			
5	10	13	20	0	28	6	5	3	500	400	350
6	16	25	32	0	28	33	25	17	200	180	150
						12	9	6			
7	8	12	16	0	11	10	16	8	160	130	100
						10	16	8			
8	11	16	22	0	23	10	8	5	160	120	90
						10	8	5			
9	20	30	40	0	23	23	18	12	200	150	110
						23	18	12			
10	10	15	20	0	23	16	12	8	150	120	80
						16	12	8			
11	17	25	34	0	23	26	20	13	80	75	70
						8	6	4			
12	12	18	24	10	22	12	9	6	160	150	140
						12	9	6			
13	20	30	40	0	28	56	42	28	400	370	350
						50	37	25			
14	12	18	24	46	81	14	11	7	220	200	190
						10	8	5			
15	40	60	80	42	93	65	50	33	470	450	400
						9	7	5			
16	6	9	12	8	48	20	15	10	240	220	200
						12	9	6			
17	20	30	40	8	48	12	9	6	220	200	180
						12	9	6			
18	6	9	12	16	48	23	18	12	150	100	90
						23	18	12			
19	4	6	8	16	48	10	8	5	150	100	80
						10	8	5			
20	6	9	12	28	58	4	3	2	80	60	50
						4	3	2			

Table 2. cont.

Lp	d_{i1}	d_{i2}	d_{i3}	α_i	δ_i	R^r_{i11} R^r_{i21}	R^r_{i12} R^r_{i22}	R^r_{i13} R^r_{i23}	R^n_{i11}	R^n_{i12}	R^n_{i13}
21	20	30	40	28	58	12	9	6	250	180	150
						7	5	4			
22	29	44	57	23	80	46	35	23	800	700	600
						42	32	21			
23	40	60	80	22	83	36	27	18	500	400	300
						20	15	10			
24	24	36	48	22	83	106	80	53	700	600	550
						6	4	3			
25	12	18	24	58	98	19	15	10	200	170	150
						19	15	10			
26	12	18	24	58	93	14	11	7	190	160	140
						10	8	5			
27	4	6	8	84	88	6	5	3	120	100	70
						5	4	2			
28	19	29	38	70	117	15	11	8	260	240	200
						15	11	8			
29	13	20	26	80	93	14	11	7	360	350	320
						10	8	5			
30	8	12	16	88	105	20	15	10	240	210	200
						16	12	8			
31	23	34	40	70	117	30	23	25	250	220	200
						17	14	9			
32	24	36	48	70	117	44	33	22	700	600	500
						44	33	22			
33	24	36	48	93	117	48	36	24	580	500	420
						38	29	19			
34	24	36	48	93	117	13	10	7	250	240	230
						13	10	7			
35	10	5	20	96	117	6	5	3	70	60	50
						6	5	3			
36	6	9	12	62	97	19	15	11	130	120	110
						15	12	8			
37	12	18	24	62	95	56	42	28	350	320	300
						6	5	3			
38	13	19	26	74	108	6	5	3	80	70	60
						6	5	3			
39	20	30	40	68	117	58	45	29	400	350	340
						48	36	24			
40	18	27	36	68	117	20	15	10	250	220	200
						20	15	10			

In Table 2, the following notation has been used: d_{i1}, d_{i2}, d_{i3} - duration of activity i for the first, the second and the third performing mode, respectively α_i - ready time of activity i, δ_i - due date of activity i, R^r_{ijk} - requirement of activity i for the j-th renewable resource

under k-th performing mode, R^n_{ijk} - requirement of activity i for the j-th nonrenewable resource under k-th performing mode.

Screen copies shown in the Appendix summarize the process of single and multiobjective calculations for the agricultural project.

References

[1] E.W. Davis and J.H. Patterson. *A comparison of heuristic andoptimal solutions in resource-constrained project scheduling.* Management Science 21 (8), 1975, pp 944-955.

[2] S.E. Elmaghraby. *Activity networks: project planning and control by network models.* John Wiley & Sons, New York, 1977.

[3] J.H. Patterson. *A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem.* Management Science 30 (7), 1984, pp 854-867.

[4] R. R.Słowiński, J.Węglarz (eds.). *Advances in Project Scheduling.* Elsevier, Amsterdam, 1989.

[5] J.H. Patterson,R.Słowiński, F.B.Talbot, J. Węglarz. *An algorithm for a general class of precedence and resource constrained scheduling problems.* in: R.Słowiński, J. Węglarz (eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp 3-28.

[6] R.Słowiński, B. Soniewicki, J. Węglarz. *MPS - komputerowy system wspomagania decyzji dla wielokryterialnego rozdzialu zasobow.* Archiwum Automatyki i Telemechaniki (to appear).

[7] R.Słowiński, B. Soniewicki, J. Węglarz. *System wspomagania decyzji dla wielokryterialnego rozdzialu zasobow, Postepy Cybernetyki.* (to appear).

[8] R.Słowiński. *Multiobjective network scheduling with efficient use of renewable and nonrenewable resources.* Europ. J. Opl. Res., 7 (3), 1981, pp 265-273.

[9] R.Słowiński. *Problrmes d'allocation de moyens limitws en prwsence de critrres multiples, Cahier de LAMSADE, no.36, Universitw de Paris-Dauphine.* Paris, 1981.

[10] R.Słowiński. *Modelling and solving multicriteria project scheduling problems.* in A.Straszak (Ed.), Large Scale Systemst-t Theory and Applications, Pergamon Press, Oxford, 1983, pp 469-474.

[11] J. Błażewicz, W. Cellary, R.Słowiński, J. Węglarz. *Scheduling Under Resource Constraints - Deterministic Models.* J.C.Baltzer AG, Annals of Operations Research, vol.7, 1986, Basel.

[12] R.Słowiński, J. Węglarz. *An interactive algorithm for multiobjective precedence and resource constrained scheduling problems.* in: Vriethoff, J.Visser, H.H.Boerma (Eds.), Proc. 8th INTERNET Congress, Elsevier/North-Holland, Amsterdam, 1985, pp 866-873.

[13] R.Słowiński. *Multiobjective project scheduling under multiple-category resource constraints.* in: R.Słowiński, J. Węglarz (eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp 151-167.

[14] J.E. Kelley Jr. *The critical-path method: resource planning and scheduling.* in: J.F.Muth and G.L.Thompson (eds.), Industrial Scheduling, Prentice Hall, Engl. Cliffs, New Jersey, 1963.

[15] R.B. Harris. *Precedence and Arrow Networking Techniques for Construction.* Wiley, New York, 1978.

[16] S. Kirkpatrick, C.D. Gellat Jr. M.P.tVecchi. *Optimization by simulated annealing.* Science 220, 1983, pp 671-680.

[17] V. Cerny. *The termodynamical approach to the travelling salesman problem: An efficient simulation algorithm.* J. Opt. Theory Appl. 45, 1985, pp 41-51.

[18] J.H. Patterson, R.Słowiński, F.B. Talbot, J. Węglarz. *Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems.* Europ. J. Opl. Res. (to appear).

Appendix

Screen copies of the MPS

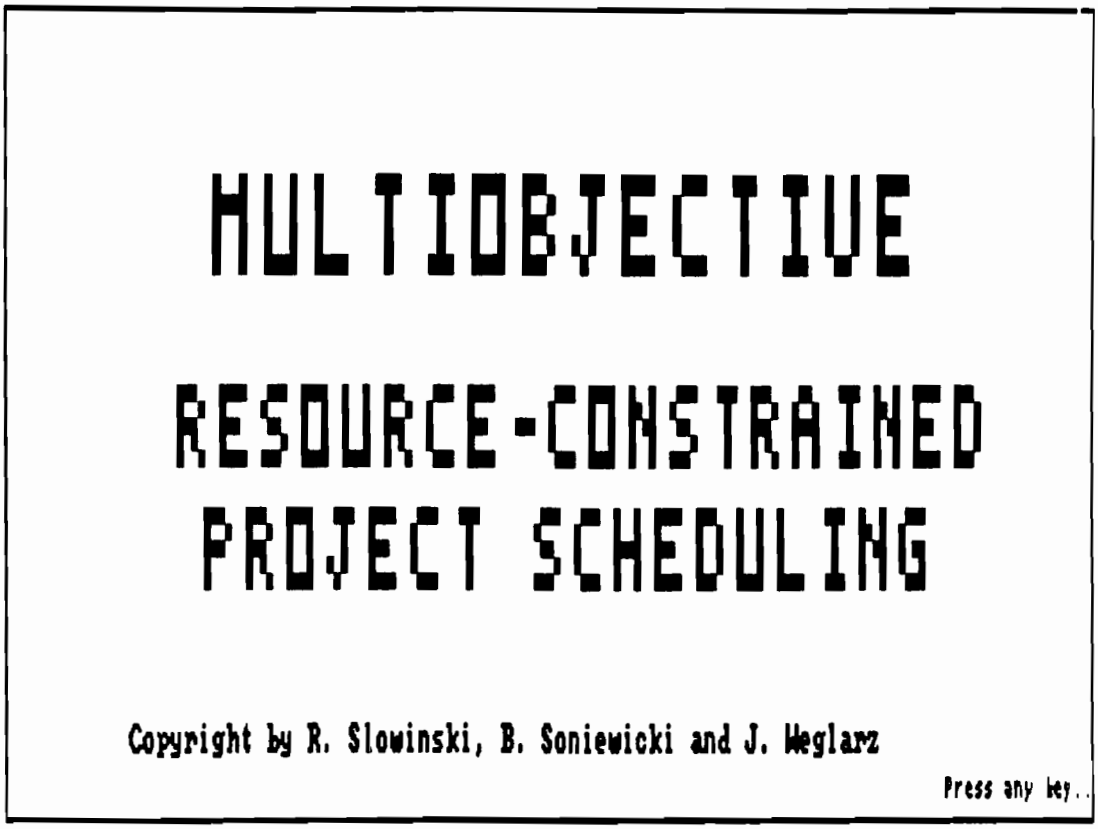


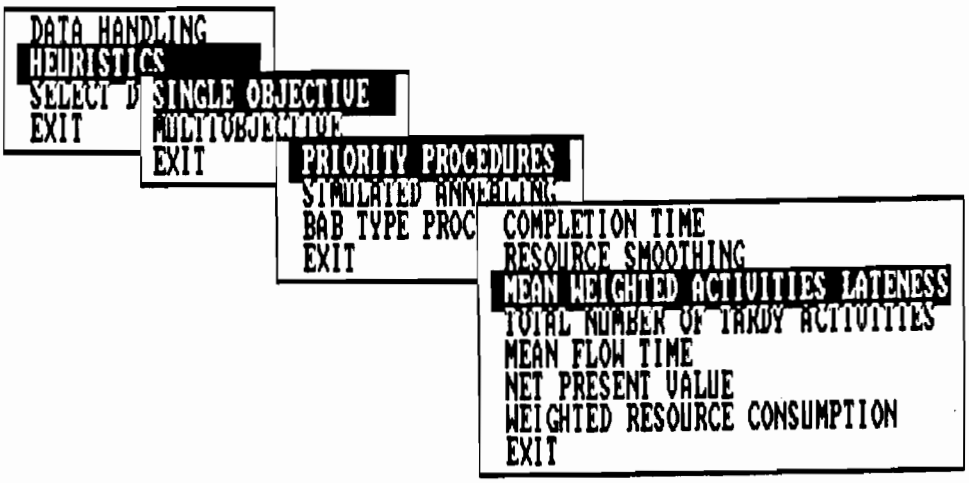
Fig. 2

Multiobjective Resource-Constrained Project Scheduling

The Name of the Active File :

Select item with ↑ or ↓ , then hit RETURN

Fig. 3



The Name of the Active File : RADZIMRA

Select item with ↑ or ↓ , then hit RETURN

Fig. 4

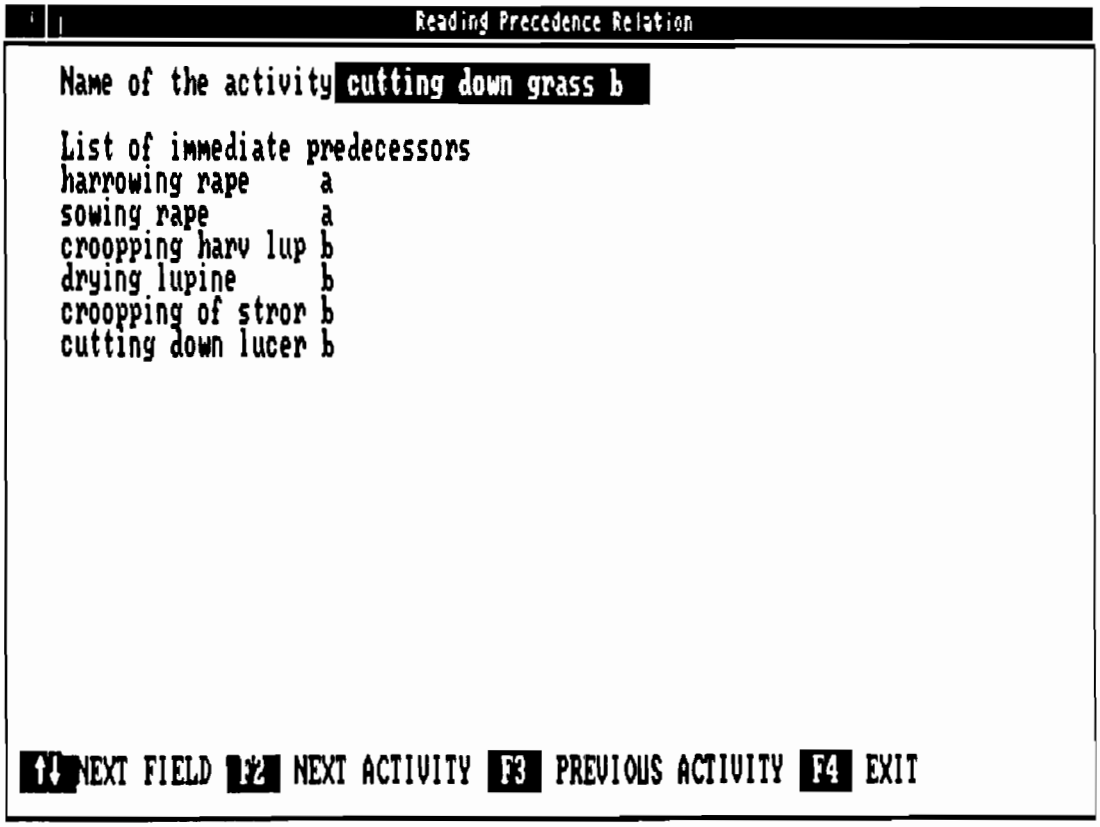


Fig. 5

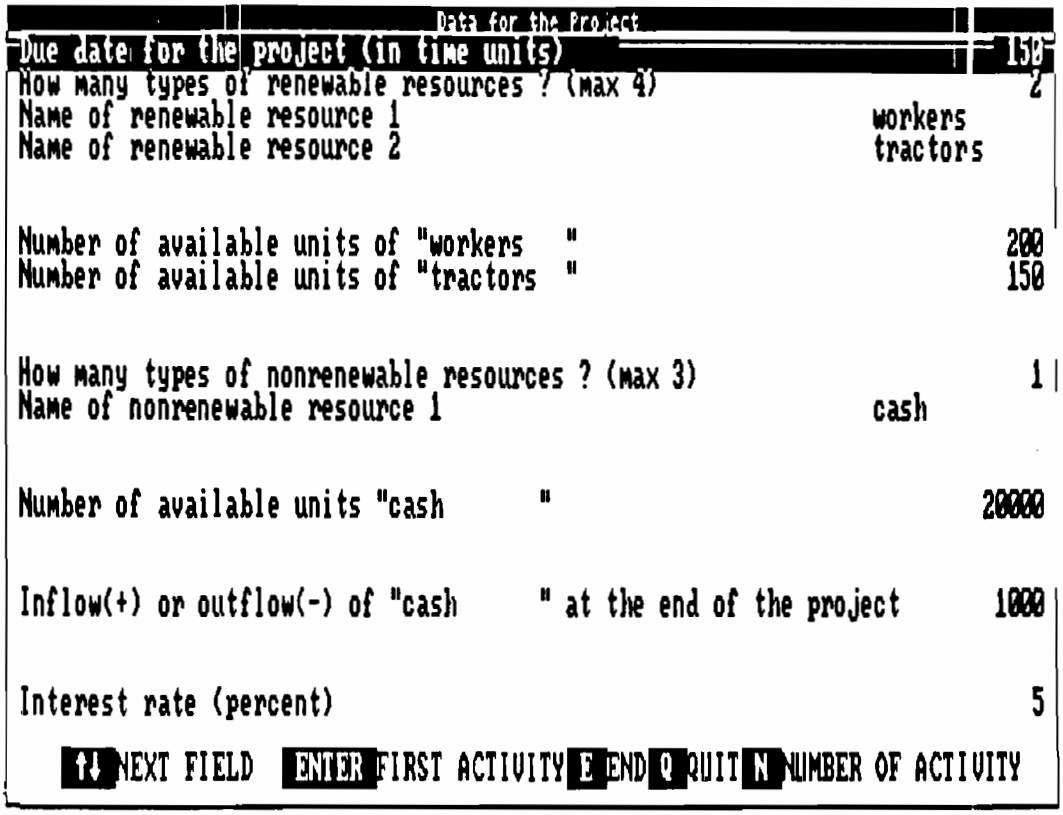

Data for the Project
Due date for the project (in time units) 150
How many types of renewable resources ? (max 4) 2
Name of renewable resource 1 workers
Name of renewable resource 2 tractors
Number of available units of "workers " 200
Number of available units of "tractors " 150
How many types of nonrenewable resources ? (max 3) 1
Name of nonrenewable resource 1 cash
Number of available units "cash " 20000
Inflow(+) or outflow(-) of "cash " at the end of the project 1000
Interest rate (percent) 5
↑↓ NEXT FIELD ENTER FIRST ACTIVITY E END Q QUIT N NUMBER OF ACTIVITY

Fig. 6

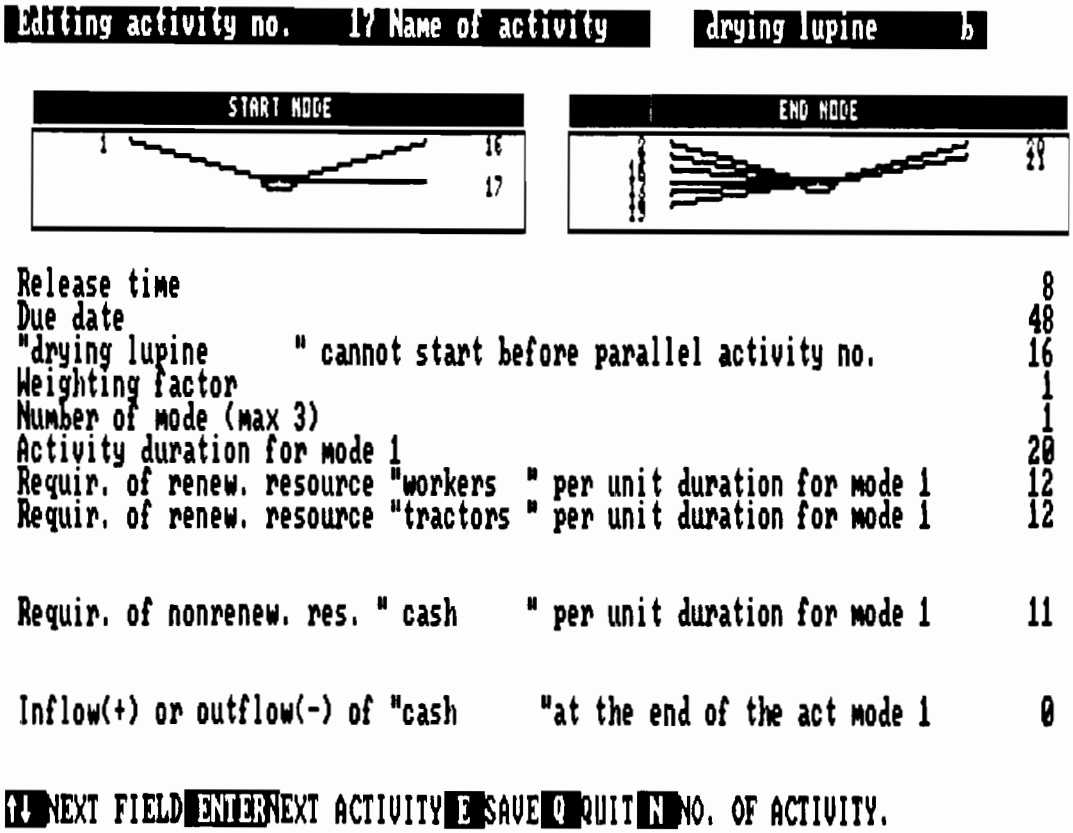
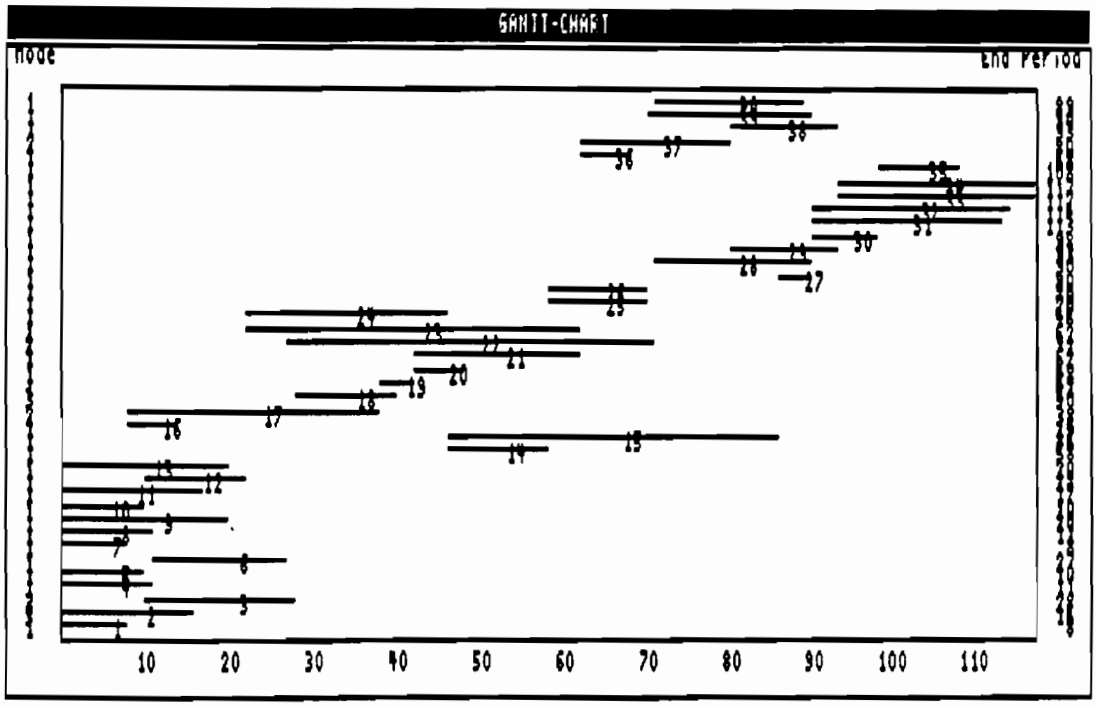

Editing activity no. 17 Name of activity drying lupine b
START NODE
END NODE
Release time 8
Due date 48
"drying lupine " cannot start before parallel activity no. 16
Weighting factor 1
Number of mode (max 3) 1
Activity duration for mode 1 20
Requir. of renew. resource "workers " per unit duration for mode 1 12
Requir. of renew. resource "tractors " per unit duration for mode 1 12
Requir. of nonrenew. res. " cash " per unit duration for mode 1 11
Inflow(+) or outflow(-) of "cash "at the end of the act mode 1 0
↑↓ NEXT FIELD ENTER NEXT ACTIVITY E SAVE Q QUIT N NO. OF ACTIVITY.

Fig. 7

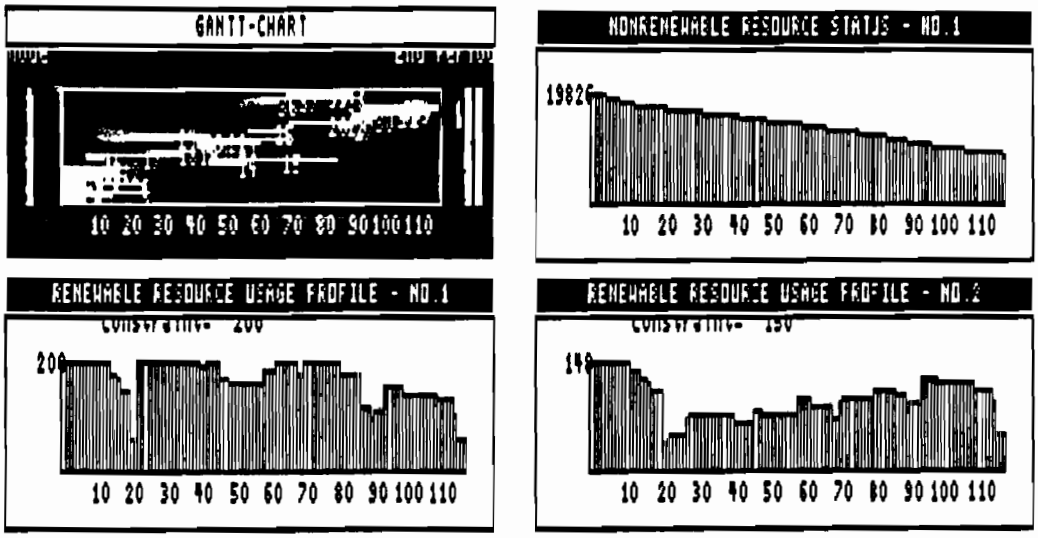
```
CT = 118




Permuted activities  18  34
                        Acceptance = YES
                        Probability of acceptance = 0.1711
                        Temperature=  20.2
                        Current value of  CT   = 141.00
                        Previous value of CT   = 141.00

Single Objective Optimisation
Simulated Annealing Procedure
Project Duration
Please, wait...
```

Fig. 8



H- HARDCOPY Q- QUIT

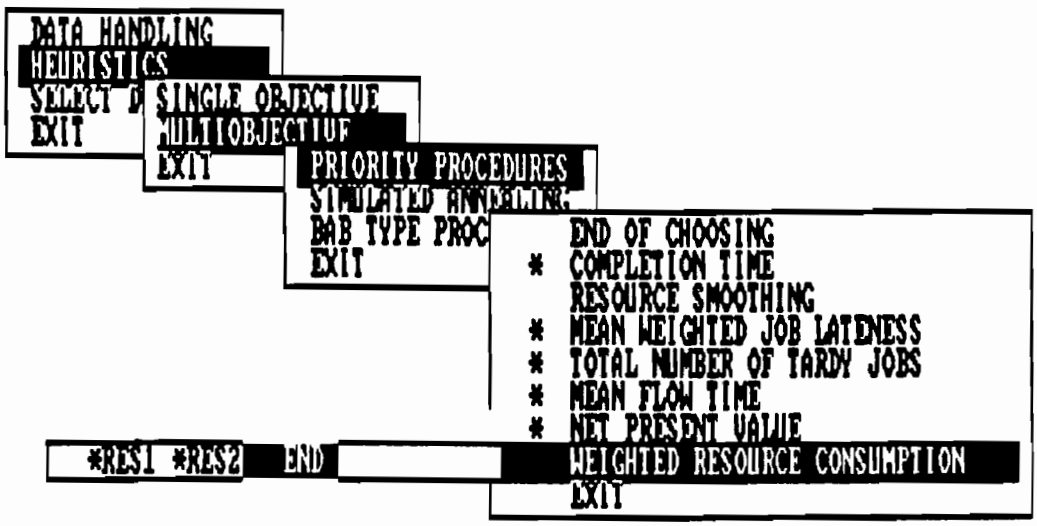
Fig. 9



O- OPTION ↑↓→← MOVE WINDOW E- ENLARGEMENT H- HARDCOPY Q- QUIT

Fig. 10

Multiobjective Resource-Constrained Project Scheduling



The Name of the Active File : PRZ3

Select item with ↑ or ↓ , then hit RETURN

Fig. 11

```
                                                          NNDS= 25

     Current point                    Current ideal point
      CT  =   131.00                   CT  =   118.00
     MWJL =    53.00                  MWJL =    23.00
     TNTJ =     8.00                  TNTJ =     3.00
     MWFT =   902.00                  MWFT =   788.00
     NPV1  = 17415.05                 NPV1  = 17807.78
     WRC1  = 10576.00                 WRC1  = 10131.00


  Permuted activities   2  22
                        Acceptance = YES
                        Probability of acceptance = 1.0000
                        Temperature=4500.0
                        Current value of  NPV1  = 17556.05
                        Previous value of NPV1  = 17536.36


  Multiobjective Optimisation
  Please, wait...
```

Fig. 12

Comparison of the obtained solution with the previous one and the reference point

NNDS= 24

Objectives	CT	MWJL	TNTJ	MWFT	NPV1	WRC1
Ideal point	118	23	3	788	17808	10131
Nadir point	161	303	16	1300	17313	10829
Previous but one	0	0	0	0	0	0
Previous	0	0	0	0	0	0
Obtained	122	34	7	811	17369	10465
Relax level						
Relaxed level						

N ND SOLUTION WITH IMPROVED VALUE OF A CHOSEN CRITERION
A ANOTHER ND SOLUTION
↑↓ CHOOSE THE FIELD ENTER ACCEPT D DELETE CHANGING V VIEW

If you relaxed at least one field, you continue the calculation

Fig. 13

Comparison of the obtained solution with the previous one
and the reference point

NNDS= 24

Objectives	CT	MWJL	TNTJ	MWFT	NPV1	WRC1
Ideal point	118	23	3	788	17808	10131
Nadir point	161	303	16	1300	17313	10829
Previous but one	128	47	7	887	17548	10373
Previous	122	34	7	811	17369	10465
Obtained	122	59	7	788	17313	10307
Relax level						
Relaxed level		60				

N ND SOLUTION WITH IMPROVED VALUE OF A CHOSEN CRITERION
i ANOTHER ND SOLUTION
-/+ CHOOSE THE FIELD ENTER ACCEPT D DELETE CHANGING V VIEW

If you relaxed at least one field, you continue the calculation

Fig. 14

Comparison of the obtained solution with the previous one
and the reference point

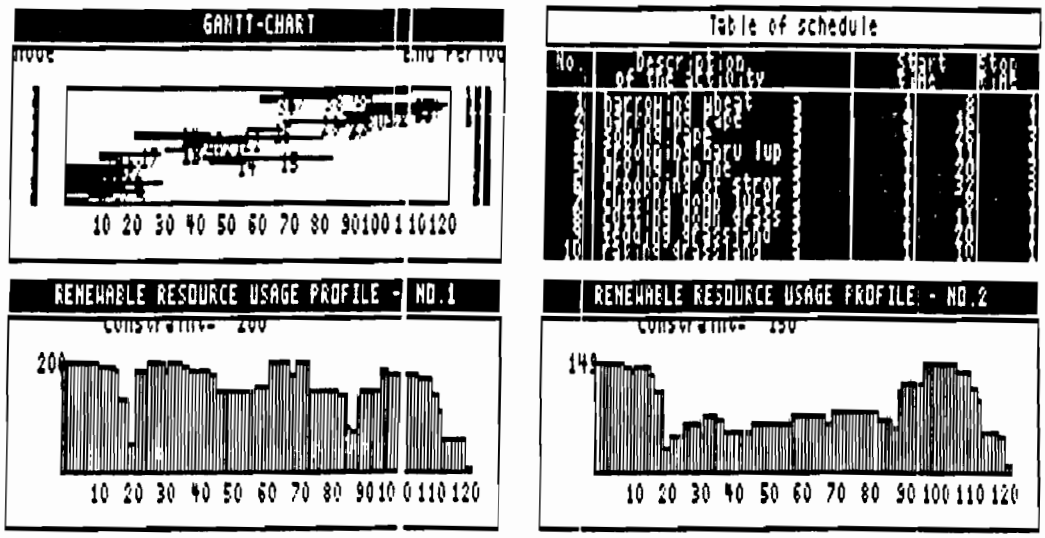
NNDS= 24

Objectives	CT	MWJL	TNTJ	MWFT	NPV1	WRC1
Ideal point	118	23	3	788	17808	10131
Nadir point	161	303	16	1300	17313	10829
Previous but one	122	59	7	788	17313	10307
Previous	122	59	7	788	17313	10307
Obtained	122	45	7	826	17498	10282
Relax level						
Relaxed level		60		850		

N ND SOLUTION WITH IMPROVED VALUE OF A CHOSEN CRITERION
i ANOTHER ND SOLUTION
-/+ CHOOSE THE FIELD ENTER ACCEPT D DELETE CHANGING V VIEW

If you relaxed at least one field, you continue the calculation

Fig. 15



PgDn PgUp - SCROLL WINDOW
O - OPTION ↑↓→← - MOVE WINDOW E - ENLARGEMENT H - HARDCOPY Q - QUIT

Fig. 16

MIPS : a DSS for Multiobjective Interactive Project Scheduling

Tomasz Ryś, Rafał Stanek and Wiesław Ziembla
Joint Systems Research Department
of Institute for Control and Systems Engineering,
Academy of Mining and Metallurgy, Cracow and of
Industrial Chemistry Research Institute, Warsaw

Abstract

In this paper we describe MIPS, a prototype version of a screen oriented graphics software package for interactive resource constrained project scheduling with multiple objectives. This system has been designed as a menu driven screen oriented Decision Support System. All the decision situation terms, choice and consequences are graphically represented through special window system. While working in the interactive mode, the user gets an approximate schedule based on a heuristics selected from list offered by the system and may further improve it by moving a task or a complex of tasks. At the same time the impact of the schedule modification on the quality measures and the resource profile utilization are reported on diagrams and tables. The criteria of economic type (total income, total project cost, net present value) as well as classical measures in project scheduling such as penalties for earliness and tardiness, completion time are also incorporated in the MIPS model.

1 Introduction

MIPS is a prototype version of a screen oriented graphic software package for interactive multiobjective project scheduling. Initially it was devised as a decision analysis and decision support system in the field of investment scheduling (Rys and Ziembla, 1988; Kopytowski and Zebrowski, 1989) viewed as a problem of setting start-up dates of plants selected for investment program under constraints on capital distribution over the time horizon. This process was considered as a multiobjective task with the objectives such as: completion time, net present value, total investment cost, total delay of construction etc. The prototype version of MIPS was modified in the context of the international exercise coordinated by IIASA (Antonisse and all, 1988), thus enabling its application for more general decision problems. The main area of the system's operation may be classified as resource constrained project scheduling.

The decision situation, multiobjective project scheduling, can be described by a set of tasks, each to be processed by a set of resources in a certain time interval. Tasks can be grouped into complexes that are to be completed simultaneously. For each task a release time, deadline and due dates which define a possible time interval of task performance and its best completion time, are determined. The weight parameters may also be defined

corresponding to the unit profit, penalty for earliness and/or tardiness. These parameters are used to assess the quality of a schedule. A task in the project can be processed in alternative ways which differ in processing times and resource requirements (upon the condition however that no interruptions are allowed and the mode of performance remains the same).

Three classes of resources may be considered in the decision problem:

- renewable resources (renewable over the time horizon e.g. machines in job shop scheduling problem),
- unrenewable resources (that are being used up in the course of time e.g. capital, energy),
- double constrained resources (total and intensity constrained).

The same resource unit cannot be allocated to two tasks at the same time. For each resource the cost of its utilization is specified and this enables to assess the project performance cost being one of the objectives. Another assumption is that the resources are available in certain time intervals.

Other elements of the decision situation may be defined by the relations of partial order type described below. For any two tasks precedence constraints with minimum and/or maximum waiting time between the start of the first one and the completion of the second one can be specified. Similarly the dates in the case of simultaneous task schedule requirements refer to the completion time of the first and the second task.

A schedule may be regarded as the complete one when a proper completion mode described by resource set requirement is assigned to each task, start up and completion times are calculated and all the above constraints are satisfied. The following criteria can be used to assess the quality of a schedule:

- profit obtained in the project,
- net present value,
- total cost of resource utilization over the project,
- project completion time,
- total tardiness and earliness of task completion

The first three criteria are of cash flow type and additionally the system enables to calculate them in a modified discounted form for the selected discount rate.

More detailed mathematical formulation of the above problem is presented in the section 2. Section 3 contains a description of an automatic solver procedure. In design of the prototype version of MIPS package, the search for the solution that is both accurate and best from the mathematical point of view was not of greatest concern. On the contrary an approach was developed based rather on linking of simple algorithms based on heuristic rules which quickly generate good schedules, to a system of easily applicable tools enabling the user for unsophisticated modification, processing and assessment of schedules. The user can, by means of a Gantt chart presented in one of the system's windows, move the given task or a complex of tasks in the project, he can change the mode of its realization and monitor the impact of these changes on chosen quality criteria of the schedules. This in detail is described in Section 4. Therefore MIPS in the actual

prototype version does not simply solve the multiobjective project scheduling problem. It rather helps the user to select interactively the solution which he considers to be the best for him.

In Section 5 MIPS implementation is shortly described. Finally in Section 6 an example of practical application for solving an investment scheduling problem is presented.

2 Mathematical model formulation

Now we may present the mathematical formulation of resources constrained project scheduling. The symbols are categorized as follows:

$k, l \in K$ set of tasks to be scheduled,

$< k, l > \in R_1$ predecessorship relation,

$< k, l > \in R_2$ simultaneous completion relation,

$i \in I$ resources,

$t \in [T_0, T]$ time horizon ,

due_k due date for k-th task,

d_k deadline for k-th task,

e_k release time for k-th task,

s_k starting time for k-th task,

p_{km} duration of k-th task performance by m-th mode,

c_i cost of one unit of i resource,

r_{ikt} requirement on i resource by task k in the moment t,

Wp_k weight of profits brought in by task k,

We_k weight of penalty for earliness – task k,

Wt_k weight of penalty for tardiness – task k,

f_k profit brought in by task k

$$f_k(t) = \begin{cases} 0 & \text{if } t \leq s_k + p_k - 1 \\ Wp_k(t - (s_k + p_k)) & \text{if } t \geq s_k + p_k \end{cases}$$

TP total profit brought by all tasks

$$TP = \sum_{k \in K} f_k(T), \tag{1}$$

TC total cost

$$TC = \sum_{k \in K} \sum_{i \in I} \sum_{t \in [T_0, T]} r_{ikt} c_i, \tag{2}$$

NV difference between profit and costs,

$$NV = TP - TC, \tag{3}$$

TE penalty for earliness

$$TE = \sum_{k \in K} e_k, \tag{4}$$

where

$$e_k = \begin{cases} We_k(due_k - (s_k + p_k)) & \text{if } due_k \geq s_k + p_k \\ 0 & \text{if } due_k \leq s_k + p_k \end{cases}$$

TT penalty for tardiness

$$TT = \sum_{k \in K} tt_k, \tag{5}$$

where

$$tt_k = \begin{cases} 0 & \text{if } due_k \leq s_k + p_k \\ Wt_k(s_k + p_k - due_k) & \text{if } due_k \geq s_k + p_k \end{cases}$$

Next, the constraints can be specified:

- **investment (resource) time balance**

$$\sum_{k \in K} g_{kit} \leq R_{it}, \quad t \in [T_0, T] \tag{6}$$

where

$$g_{kit} = \begin{cases} 0 & \text{if } t < s_k \text{ or } t > s_k + p_k \\ r_{kit-s_k} & \text{if } t \in [s_k, s_k + p_k] \end{cases}$$

where R_{it} is calculated depending on kind of the resource.

1. renewable resources – R_{it} is equal to availability value of the resource i.
2. unrenewable resources – R_{it} is in each time unit t decreased by amount of consumption $\sum_{k \in K} r_{it}$
3. double constrained – R_{it} is taken as lesser value of the one calculated for constrained resources and given constraint on intensity of resource utilization.

- **deadline constraints**

$$s_k + p_k \leq d_k \quad k \in K \tag{7}$$

- **release date constraints**

$$s_k \geq e_k \quad k \in K \tag{8}$$

- **constraint on predecessorship of task realization:**

$k, l \in K$ two tasks being in relation,

$minwait_{kl}$ minimal waiting time between tasks realization,

$maxwait_{kl}$ maximal waiting time between tasks realization,

1. predecessorship relation,

$$s_k + p_k + minwait_{kl} \geq s_l \tag{9}$$

$$s_k + p_k + maxwait_{kl} \leq s_l \tag{10}$$

2. simultaneous completion relation,

$$s_k + p_k + minwait_{kl} \geq s_l + p_l \tag{11}$$

$$s_k + p_k + maxwait_{kl} \leq s_l + p_l \tag{12}$$

The following objective can be formulated :

- **Maximum total profit**

$$T_p = \sum_{k \in K} f_k(T - s_k - p_k) \tag{13}$$

$$\max TP = \min \sum_{k \in K} f_k s_k \tag{14}$$

- **Minimum completion time**

$$CT = \max(s_k + p_k) \tag{15}$$

- **Minimum sum of penalty for earliness** in completion time with respect to due date

$$MinTE = \sum_{k \in K} e_k, \tag{16}$$

- **Minimum sum of penalty for tardiness** in completion time with respect to due date

$$MinTT = \sum_{k \in K} tt_k, \tag{17}$$

The above criterion is useful and should be applied when relaxation of deadlines is either acceptable or indispensable due to their violation (infeasibility of a problem).

According to the fact that different modes of task processing in the model are considered, also minimization of total project performance cost can be used as one of the criterion:

$$IN = \min \sum_{t=T_0}^{T} \sum_{k \in K} \sum_{i \in I} c_i \dot{g}_{kit} \tag{18}$$

$$g_{kit} = \begin{cases} 0 & \text{if } t < s_k \text{ or } t > s_k + p_k \\ r_{kijt} & \text{if } t \in [s_k, s_k + p_k] \\ & \text{and } x_{jk} = 1 \end{cases}$$

Binary variable x_{jk} is used for assigning the task to the mode of its completion:

$$x_{jk} = \begin{cases} 1 & \text{task } k \text{ is completed in the way } j \\ 0 & \text{otherwise} \end{cases}$$

3 Automatic problem solution

The problem formulated above is of mixed integer linear programming type. Unfortunately the problems of this kind, even for a single criterion, belong to the NP-complete class (Garey and Johnson, 1979) due to their computational complexity. The accurate method for solving such problems is based on implicit enumeration combined with problem oriented elimination rules (e.g. Talbot and Paterson 1978). In the case of project scheduling presented here, a methodology based on heuristic and approximate algorithms was developed in order to achieve practically applicable decision support. The results obtained so far confirm practical applicability of the approach. This comes merely from the fact that the necessity of evaluation and ranking of various schedules is the most important aspect of decision making with respect to scheduling while costly and laborious enumeration of alternatives is not so crucial. Similar approach is presented by other authors (Davis and Patterson, 1975; Slowinski, 1978; Russell, 1986; Dumond and Marbert, 1988).

Below heuristic rules are given based on identification and practical experience in dealing with single criteria problems. They can be presented as follows:

1. Profit - $P(k) = f_k(T)$; descending order of respective individual task income,

2. Resources - descending order of respective "complexity of resources requirements and duration",

$$P(k) = C_k + \sum_{j \in S_k} C_j \tag{19}$$

where

- $C_k = \sum_{i \in I}(r_{ki}/R_i)(U_i/U_{max})$
- $U_i = U_{pi} + \sum i/R_i$
- $U_{max} = \max_{i \in I} U_i$
- $r_{ki} = \sum_{t \in [T_0, T]} r_{ikt}$
- Up_i - the latest term where a non-zero demand for resource i occurs,
- $\sum i$ - area of the profile of i resource utilization above the R_i constraint,
- S_k - a set of successors of task k

3. Profit/Cost - $P(k) = f_k(T)/(\sum_{i \in I} \sum_{t \in [T_o, T]} r_{ikt} p_k$; descending order of respective ratio of individual income over individual cost of completion,

4. Shortest proc. time - $P(k) = 1/(p_k + 1)$; descending order of respective individual task duration,

5. Min LF time - $P(k) = 1/(LF_k + 1)$, LF_k ; ascending order of respective possible latest task completion time - based on PERT analysis,

6. Least TF - $P(k) = 1/(TK_k + 1)$, TF_k ; ascending order of total task float - based on PERT analysis

7. Most successors - $P(k) = |S_k|$; descending order of respective power of successors set,

8. Least TF per succ. – $P(k) = (|S_k| + 1)/(TF_k + 1)$; ascending order of respective float per power of successors set,

9. Max rem. path length – $P(k) = p_k + \sum_{j \in S_k} p_j$; descending order of respective remaining path length,

10. Max res. demand – $P(k) = p_k \sum_{j \in I} r_{kj}$; descending order of respective resource demand

11. Can start first – $P(k) = 1/(t_k+1)$, where t_k denominates the earliest term of starting the task performance with a given schedule; can start first considering individual resource requirement,

12. Random – $P(k)$ = Random value ; descending order of respective random value for tasks.

As each task may be performed in several ways differing by time of completion and utilization of resources, the mode of task performance is selected prior to definition of priorities for some of the heuristic rules:

1. Profit – according to shortest realization times,

2. Resources – according to smallest profiles of resource utilization,

3. Profit/Cost – according to lowest costs,

4. Shortest proc. time – according to shortest realization times,

5. Max rem. path length – according to shortest realization times,

6. Max res. demand – according to smallest profiles of resource utilization

7. Random – the mode is selected at random,

For the remaining heuristic rules, the mode of task realization is defined by the user.

The scheduling algorithm based on the values calculated along the heuristic rules described above may be presented as follows:

1. selection of modes of tasks realization,

2. preliminary scheduling based on PERT analysis with determination of earliest start-up terms and determination of parameters related to critical path analysis,

3. determination of a set of such tasks Q, predecessors of which are already scheduled; in case the set is empty – the calculations are terminated; set Q^{-1} denominates the set of the tasks already scheduled;

4. calculation of priorities for tasks of set Q and selection of the task X with the highest priority;

5. checking the relation between task X and the tasks of the set Q^{-1}; if the relations are erroneous, then increasing of earliest terms of start-up of the tasks, so as to make them proper, then going to point 3;

6. an attempt of scheduling of task X (i.e. finding such a start-up term to satisfy the resources constraint); if it fails then increasing the earliest start-up terms of the last task from the set Q^{-1}, deleting it from set Q^{-1} and going to point 3; if the set Q is empty then termination of calculations without success takes place; if the task was scheduled then linking it to set Q^{-1}; if there are any tasks left to be scheduled go to point 3.

7. termination of calculations.

As a result of sequencing according to the rules H1 - H12 and their combination, the schedule obtained provides de facto priorities for the considered activities. Additionally the decision maker is also in position to impose (interactively) his own preferences. This gives him a possibility to modify the priorities resulting from the above rules.

4 Interactive work with a schedule

As it was mentioned above since no efficient algorithm for solving the resource constrained scheduling problem exists, even according to a single criterion which is of NP hard type, certain heuristic procedures are designed in such a way as to enable for a quick build up of automatic schedules which are good or practically acceptable rather than optimal. These schedules can be now examined and/or improved. All terms, conditions and features of the decision situation are clearly graphically represented by a system of windows.

On the first screen MIPS offers by default two main window diagrams. The first presents a schedule as task allocation (y - axis) over time (x - axis), and the second shows the quality of the schedule by a bar chart i.e. :

- total profit obtained during project horizon,
- total project cost,
- net present value,
- total tardiness as penalty for task completion after due date,
- total earliness as penalty for task completion before due date.

(using two different kinds of grids that represent the values in nominal and discounted form).

The systems provides the user with simple tools for schedule modification. The user can, using the mouse or the keyboard, select any task and move it to any place within the schedule which is equivalent to change its in terms of realization. The user is informed both about exceeding of time limits (due dates, deadlines, release time) or violation the predecessorship relation and on the impact of these changes on quality measures of the schedule, is simultaneously visible on the second diagram.

The user may also in a simple way cause a parallel shift of several tasks grouped in a complex indicated by a chosen task (i.e. relative terms of tasks realization in the complex remain unchanged). It is also possible for the user to change the way of realization of any task and checks the impact of this change on the quality of the schedule.

Of course all the above actions lead to the change of profile of resource utilization over the time horizon. The user can monitor these changes directly on the diagrams shown in separate windows. Other windows show time diagrams of quality measures of the schedule

i.e. profit, cost of the project or net present value. Due to different grids applied, nominal and discounted values may be monitored.

The windows described above are offered by the system "by default". However on consecutive screens the user can put together in arbitrary way the diagrams that are interesting to him. Moreover a set of graphical functions is available to facilitate the analysis a.o.:

- zoom – zooming on interesting fragments of the diagram,
- value – reading out the values from the diagram using cross-cursor,
- table – presentation of diagram contents in a form of a table.

The examples of the windows contents offered by the system for the case of investment scheduling problem in the petrochemical industry are shown in Appendix.

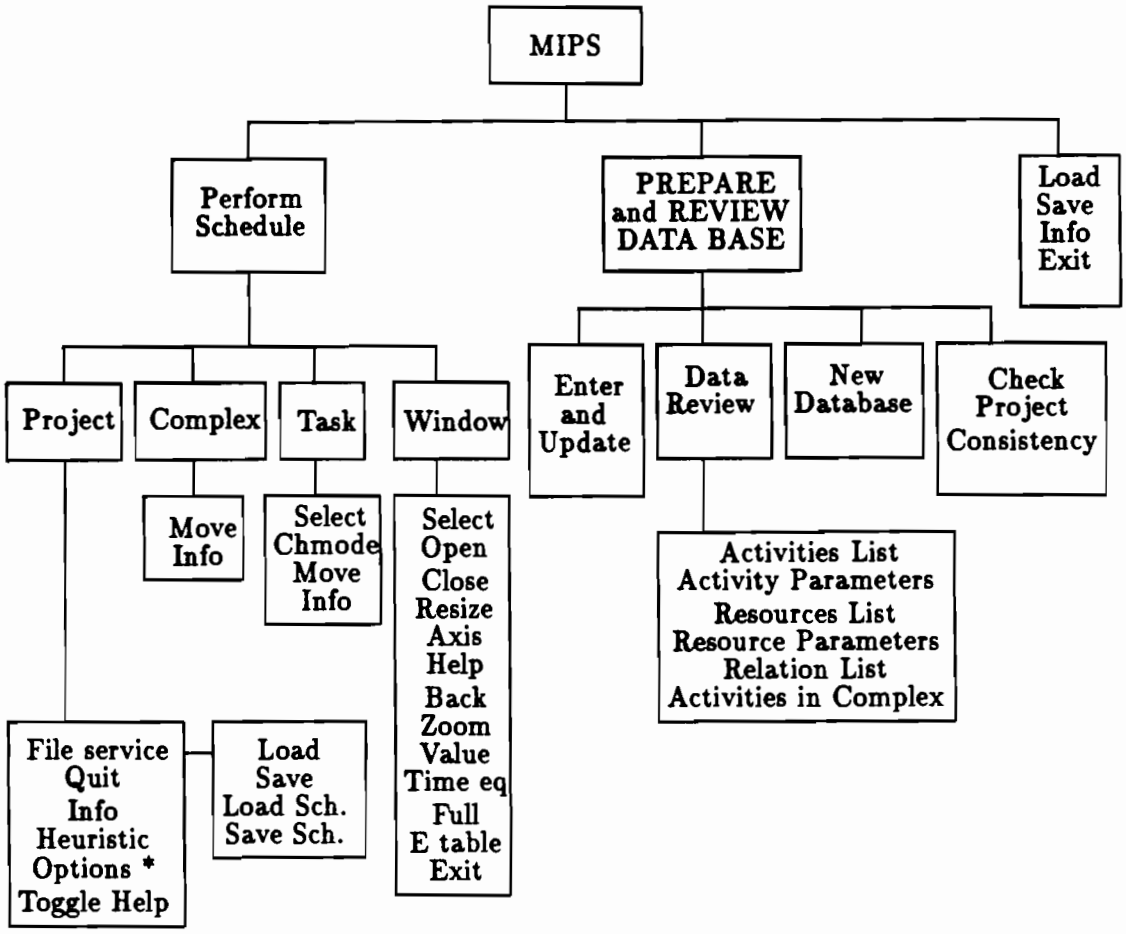


Figure 1: Menus tree of the MIPS system.

5 Short description of MIPS implementation

The system of windows as well as the whole MIPS have been designed as a menu driven system. It means that everybody who operates the system is guided by a hierarchically organized collection of menus that contain all the functional options offered by the system (Fig. 1.).

Prepare and review data

Using the first option of the menu, a user gets access to MIPS relational database management system (DBMS) which enables him to enter, store, manipulate and retrieve the information organized into the database files. The hierarchy of these files is presented in Fig. 2. The user communicates with DBMS of MIPS by means of screen forms and once a form is displayed he can browse through the records and files using `QUERY`, `NEXT` and `PREVIOUS` commands. He can also `ADD`, `DELETE` and `UPDATE` the data. Each time a tree illustrating the structure of the database is displayed at the bottom of the screen,

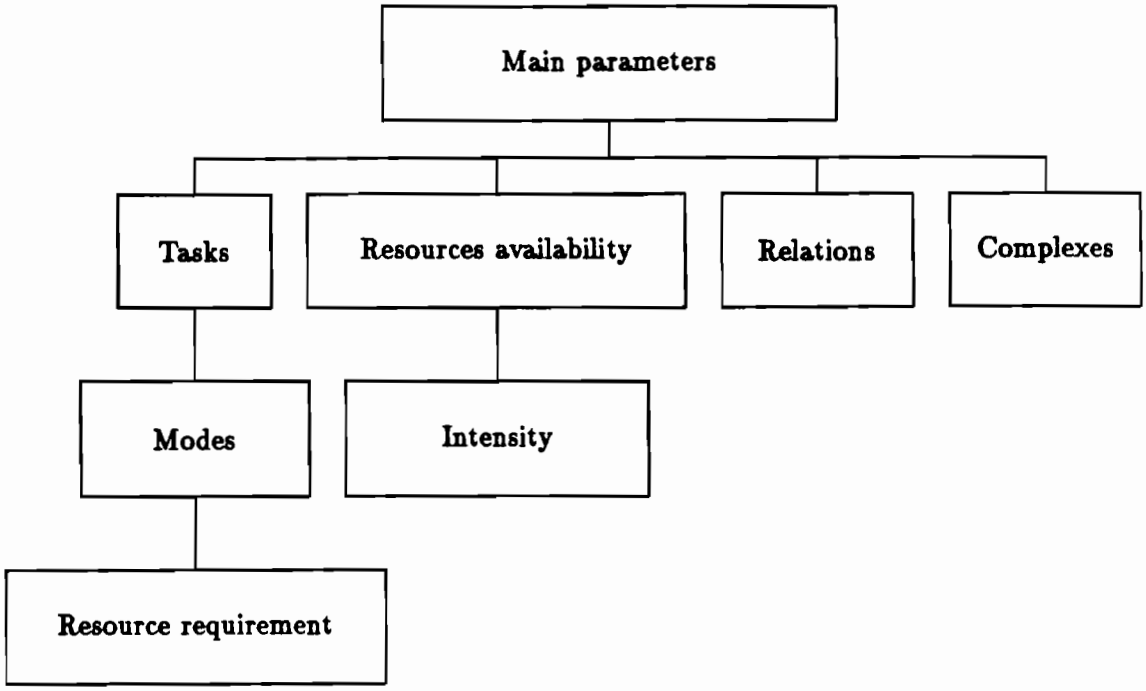


Figure 2: Hierarchy of database files in the MIPS system.

below the form of entering and manipulation with the data. The actual level on which the operations are performed, is highlighted.

Contents, structure and references in the data files are designed according to the problem structure described verbally in the Introduction and formally in section 2.

Taking into account the specific application of MIPS data files also a selection of data reviewing and consistency checking option (e.g. circle in graph) are provided by the system.

Perform schedule

By selection of this option the user activates a system of windows procedure which provides:

- automatic list scheduling based on collection of heuristic rules for priority calculation,
- manual interactive development of a schedule from scratch,
- interactive modification of a schedule and analysis of the impact of this modification on the quality of the schedule,
- relaxation of some constraints and analysis of the impact of this modification on the quality of the schedule.

All the results can be presented in the form of graphical diagrams. The option has for branches:

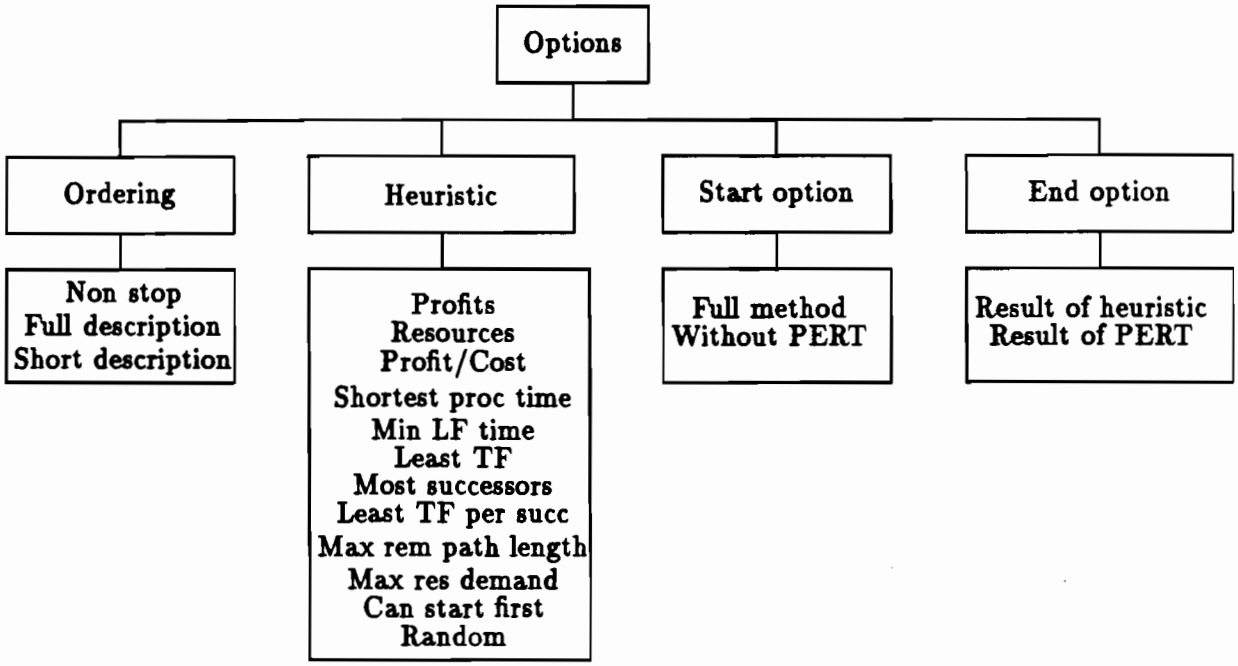


Figure 3: Menus tree of the MIPS system - cont.

Project	provides options dealing with the whole project
Complex	refers only to the selected complex
Task	refers only to the selected task
Window	provides window type activities

All the above options are presented in Fig. 1.

In the submenu **Project** the user sets up the option on MIPS automatic solution. He can select (submenu **OPTION**, see Fig. 3) one of the heuristic rules described in Section 3, mode of exit from automatic scheduling procedure in case no feasible solution can be found and also the mode of interactive reporting on the calculations. He can obtain by means of **Info** main characteristics of the currently elaborated schedule.

The submenu **Task** enables the user to select any task for manual completion and/or tuning. The user may also change, using this option, the mode of task performance and obtain a short characteristics of the task in the actually constructed schedule.

Similarly, the submenu **Complex** is to be used for parallel moving of a task in the complex. Prior to that operation a task must be selected which defines the complex in question. A short description of the selected complex in the schedule is also available through this option.

Now let us describe the **Window** option. **Window** management options are following:

Select	select an active window
Open	open a new window
Close	close the active window
Resize	resize or move the active window
Axis	toggle the time axis for the active window
Help	put the help sub-window inside the active window
Pack	leave only the active window on the screen
Zoom	zoom a part of the active window
Value	show values of figures in the active window
Time/_eq	use the same time domain for the selected windows
Full	dezoom and restore initial sizing of all windows
E/_table	present of the active window items in the form of tables
Exit	leave this submenu

Examples of the diagrams on the windows corresponding to the practical application in the area of investment scheduling are shown in Appendix.

Compare schedules

The results for the stored schedules as well as the schedules obtained automatically for the different heuristic rules can be compared using this option. Overall penalty criterion is formulated as a weighted sum of each single penalty criterion (i.e. total earliness, total tardiness, maximum earliness, maximum tardiness). Also total profile, total project cost, net present value and completion time are reported on the screen.

Auxiliary options

The last three options of the main menu **Load, Save** and **Info** are of the auxiliary type. The user can choose the database from the presented list or change the current working directory to find the appropriate database.

Similarly the database as well as the project data (including window layouts) can be stored on the disk. The user can also overwrite the existing database from the presented list or enter a new name. By using the **Info** system information and copyright notice is displayed.

Hardware and software requirements

MIPS runs under Disc Operating System (DOS version 3.00 or later) on IBM PC XT/AT with a graphic adapter (CGA,EGA, VGA, HERCULES) with or without PC MOUSE. A hard disc is recommended but not indispensable. Graphic procedures of MIPS are built using GSS*CGI software package therefore proper drivers for this graphic software must be installed in the computer system before running MIPS. However MIPS is provided with its own installing procedures for any of the configurations mentioned above.

6 An example of practical application – investment scheduling problem

The problem of investment scheduling can be viewed as a problem of setting start-up dates of plants selected for investment program (Kopytowski and Zebrowski, 1989; Skocz,

Ziembla and Zebrowski, 1989; Rys and Ziembla, 1988). It is to be accomplished in such a way so as to maximize the profit from investment with respect to demand distributed over a given time horizon. This is also to be done under constraints on availability of capital (distributed over time) and other resources such as energy or construction potential. In quite common situation such as in the case of a developing country the above problem has also more general dimension. This is the necessity stemming from imperative of minimization of import and maximization of export in order to improve trade balance and acquire means for repayment of investment loans. Even the above simple verbal description shows that the investment scheduling is in fact a multiobjective optimization decision problem.

In order to formulate a mathematical model a basic identity was introduced namely an industrial complex or shortly a complex. The complex is a set of technologies (installations) which are closely technologically interrelated and generally utilize a common infrastructure. In a specific case the complex may consist of only one installation. Each installation in the complex constitutes an investment task which is subject to scheduling. For a set of installations a predecessorship relation of a partial order type is defined.

The relation may be imposed by indispensable material flows between installations as well as demand distribution or none salability of certain semiproducts (which therefore have to be consumed by other installations). In this case however, notwithstanding to the common scheduling theory, the constraints are imposed through the completion dates.

An important time type of parameters, which can act as constraints or be involved in penalty criteria, are so called release and due dates. They reflect necessary time coordination with other industrial branches. For example a release date, may be imposed by date of availability of electricity from a power station or a due date may be the starting date of the production for which rubber must be supplied from a rubber plant or otherwise it would have to be imported. The value of this import represents monetary equivalent of the relaxation of the constraints imposed by release and/or due dates.

The necessity of coordination of this kind is present in any type of economy but is more critical, due to the scarcity of resources and lack of convertible currency in developing countries. In the process of investment scheduling basic constraints come from availability of resources. In the model presented here two basic constraints of this type are considered: investment level and construction potential. First one is of the nonrenewable type (during the construction period) second is a renewable one. Both types of constraints are made additive since they are expressed in monetary terms. The model allows however for differentiating various categories of construction potential (which than could be expressed in terms of man-shifts or construction machinery potential etc should this factors be critical). Differentiation of construction cost may come also from the financing mode. In case such as "turn key " type of plant contracting a construction period as a rule is shorter but the plant is more expensive. Another situation arises from changing debt/equity ratio when a higher ratio may increase availability of the capital on the expense of a higher loan. Impacts steming from all such factors should be evaluated with respect to the whole schedule while global investment can be minimized and/or subject to constraints. As an example the results of a development program of the petrochemical industry elaborated for a developing country is presented.

Half year interval was assumed as a time unit and medium term planning horizon is 14 – 15 years. The program contains 8 industrial complexes comprising 24 installations to be arranged in that time. These are different plants, starting from feedstock production, through production of plasticizers, octane enhancers, components for paints and varnishes,

Task id	Name	Start[1]	Duration[1]	Cost[2]
1.	Ethylene cracker	0	8	320.0
2.	PE HD	2	6	62.4
3.	PE LD	8	6	55.2
4.	Polypropylene	2	6	78.0
5.	Butene - 1	12	4	17.6
6.	Chloromethanes	12	4	31.2
7.	Methanol	8	7	80.5
8.	MTBE	10	5	45.0
9.	Butadiene	7	4	17.2
10.	Ethylbenzene	15	6	30.0
11.	Styrene monomer	15	6	33.0
12.	SB rubber	14	7	80.5
13.	Acetaldehyde	15	6	34.3
14.	Acetic acid	16	5	31.2
15.	Vinyl acetate	16	5	32.0
16.	Chlorine & caustic soda	13	7	73.0
17.	Vinyl chloride	23	6	42.0
18.	PVC	23	6	28.0
19.	VC/VA Copolymer	15	6	30.0
20.	Benzene	18	5	42.0
21.	ABS/SAN resins	21	6	70.0
22.	PS block polymer	21	5	60.0
23.	PS expandable	21	6	30.0
24.	Acrylonitrile	16	6	60.6

[1] in half year's intervals
[2] in mln USD

Table 1: A petrochemical investment schedule – time and cost table

Measure	Unit	Nominal	At 10% rate
Total profit	MM USD	6915	2843
Total cost	MM USD	1490	904
NPV	MM USD	5412	1939
Completion time	half year intervals	29	—

Table 2: A petrochemical investment schedule – main characteristics

to production of plastics and rubbers and raw materials for artificial fibers production.

For each of the time intervals the constraints on the investment capital to be spent were imposed. One of the elaborated schedules is presented in Table 1.

In Table 2. the main characteristics of the schedule are given (for economic measure in nominal and discounted values).

In the Appendix the same schedule is presented in the graphic form and also the characteristics of resource utilization and dynamics of quality measures are displayed on other diagrams.

7 Conclusions

As it was mentioned earlier, MIPS was created as a prototype version of an extension of the decision support system devised for solving investment scheduling problems. The extension concerned mainly the range of system's applications. It was practically shown, that the scheduling problem which is a complex one both theoretically and numerically, can be effectively solved by means of this system due to linking the simple algorithms based on heuristic rules to unsophisticated procedures for interactive scheduling, correction and tuning. The decision maker may interactively monitor the impact of schedule changes, using diagrams of criteria and characteristics of resources utilization.

Basically in the prototype version of MIPS, the term "Multiobjective" may be brought to parallel monitoring of several criteria. Neither a procedure of *sensu stricto* selection of compromise alternatives (overall criterion in option **Compare Schedules** is only a substitute of such a procedure), nor any method of construction of such a solution occurs in the system.

Although the mathematical aspect of the presented problem remains very interesting, however at the present stage of the work the attention was paid mainly to design and testing of the system of graphical presentation of items describing the decision situation and its effective utilization. In such a way the mechanisms for interactive project scheduling were created which enable for practical application of the system. Moreover in the situation when no clearly feasible solutions can be obtained or if the process of finding the solution is very laborious and time consuming, a decision maker is offered a number of solutions not always feasible however, but certainly useful. The flexibility of the system

enables also for quick analysis of the impact of modification of constraints (changes in decision situation) on the quality of the solutions obtained.

Despite the fact that the MIPS system proved its usefulness in practical applications, at the next stage of design process the stress will be put by the authors on improvement of automatic planning algorithms and on implementation of the procedure of multicriterial selection of the alternatives from among the generated ones.

Acknowledgment

This study is sponsored by The Chemical Industry in Poland but co-sponsorship of the theoretical part of this work by IIASA *International Institute of Applied Systems Analysis, Laxemburg, Austria*) is gratefully acknowledged.

References

Antonisse J.M. and K.M. van Hee (1988) Resource Constrained Project Scheduling : an International Exercise in DSS Development. *Decision Support Systems* Vol 4, No 2, pp. 249-258.

Davis E.W. and J.H. Paterson (1978) A Comparison of Heuristic and Optimal Solution in Resource Constrained Project Scheduling. *Management Sci.* Vol 21, No 8, pp. 944-955.

Dumond J. and V. Marbert (1988) Evaluating Project Scheduling and Due Date Assignment Procedures. An Experimental Analysis. *Management Sci.* Vol 34, No 1, pp. 101-118.

Garey M.R. and D.S. Johnson (1979) Computers and Intractability. A Guide to the Theory. of NP-Completeness, Freeman, San Francisco.

Kopytowski J. and M. Żebrowski (1989) MIDA : Experience in Theory, Software and Application of DSS in the Chemical Industry. In: Aspiration Based Decision Support Systems, Lewandowski A. and A.P.Wierzbicki eds., *Lecture Notes on Economics and Mathematical Systems.* Vol.331, pp. 271-286, Springer-Verlag.

Russell R.A. (1986) A Comparison of Heuristic for Scheduling Projects with Cash Flows and Resource Restriction. *Management Sci.* Vol 32, pp. 1291-1300.

Rys T., W.Ziembla (1988) Multiobjective Investment Scheduling Problem. Paper presented in Yalta Conference on Multiple Criteria Decision Making, to appear in: *Lecture Notes on Economics and Mathematical Systems.*

Skocz M., M.Zebrowski and W.Ziembla (1989) Spatial Allocation and Investment Scheduling. In: Aspiration Based Decision Support Systems, Lewandowski A. and A.P.Wierzbicki eds., *Lecture Notes on Economics and Mathematical Systems.* Vol.331, pp. 322-338, Springer-Verlag.

Slowinski R.(1978) A Node ordering heuristics for network scheduling under resource constraints. *Foundation of Control Engineering* Vol 3, No. 1 pp. 19-27.

Slowinski R.(1981) Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European J. of Operational Research* Vol 7, pp. 265-273.

Talbot F.B. and J.H. Paterson (1978) An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource Constrained Scheduling Problem. *Management Sci.* Vol 24, No 11, pp. 1163-1174.

Appendix

A brief description of sample windows offered by MIPS is presented below.

Figure 4. Basic screen of PERFORM SCHEDULE option

Window: "Petrochemical Demo Example" contains the time schedule of the tasks of the project.

Window: "Criteria" presents a bar chart of basic quality measures of the schedule displayed on the left diagram (i.e. P – total profit, C – total cost, NPV – net present value, E – total earliness, T – total tardiness).

Figure 5. The dynamics of profit, cost and NPV

Brighter colors refer to nominal values, darker to the discounted ones.

Figure 6. The dynamics of resource utilization

On the first diagram utilization of resources is marked with darker color on the background of their availability in a given time unit. Second diagram represents total utilization of a given resource. Both diagrams are built with reference to the time schedule shown on the third diagram.

Figure 7. Net Present Value

NPV is a economic criterion applied in the project scheduling. The darker diagram represents the discounted values while the brighter one represents nominal values. It should be noticed that NPV nominal value equal to zero occurs much earlier i.e. in 21st period of the schedule. Additionally the submenu "Window" offering the window type of MIPS operation, is presented on the diagram.

Figure 8. Task submenu operation

On the upper diagram, the chosen task is marked with darker color. This task may be moved or the mode of its realization may be changed. The requirements on resources for the chosen task are marked on the lower diagram.

Figure 9. MIPS task data file form

An example of filled data file form is presented on the diagram. Below the hierarchy of database files is presented together with the currently chosen operation level of the database management system.

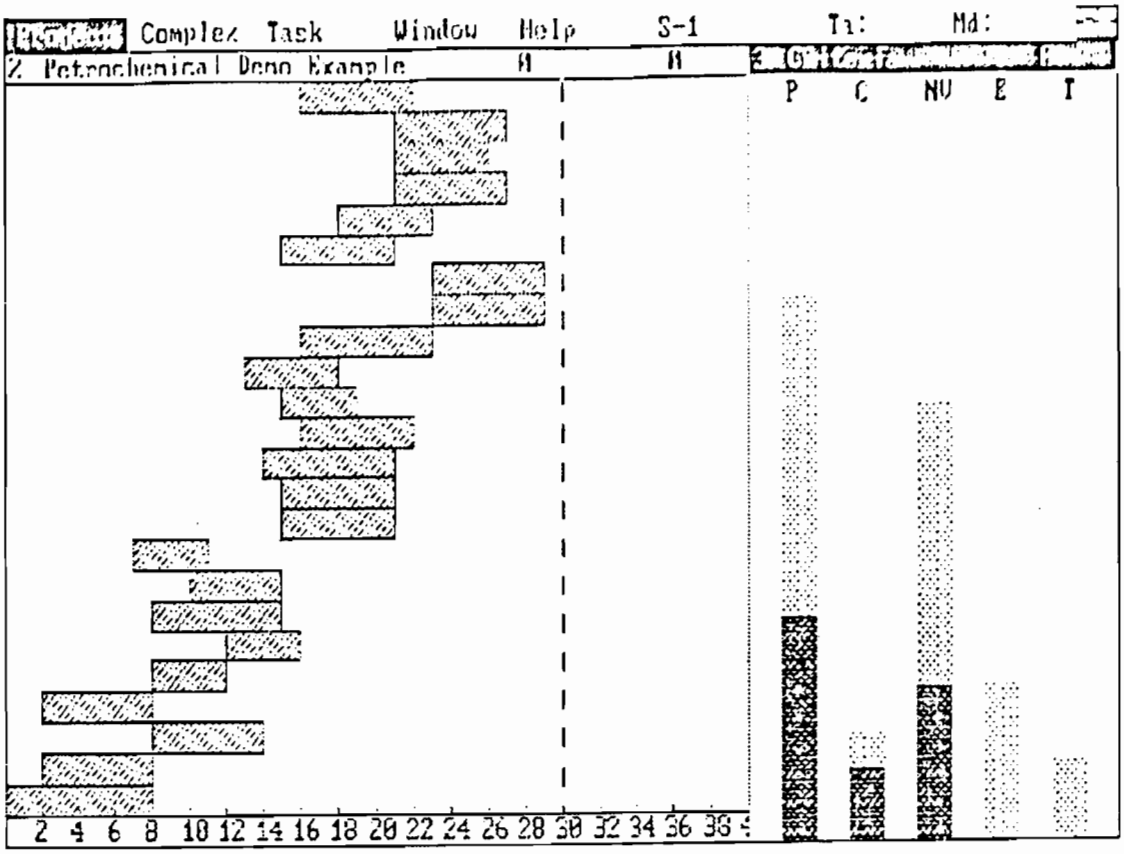


Figure 4: Basic screen of PERFORM SCHEDULE option.

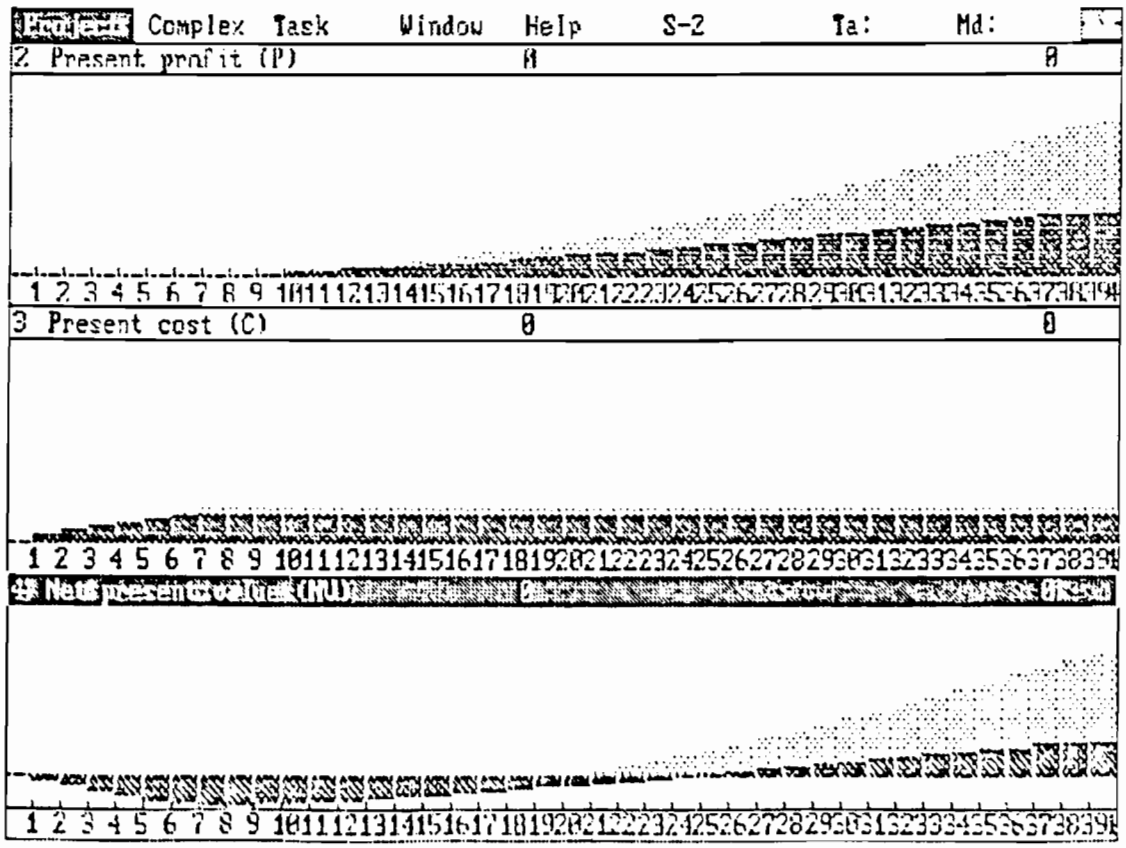


Figure 5: The dynamics of profit, cost and NPV.

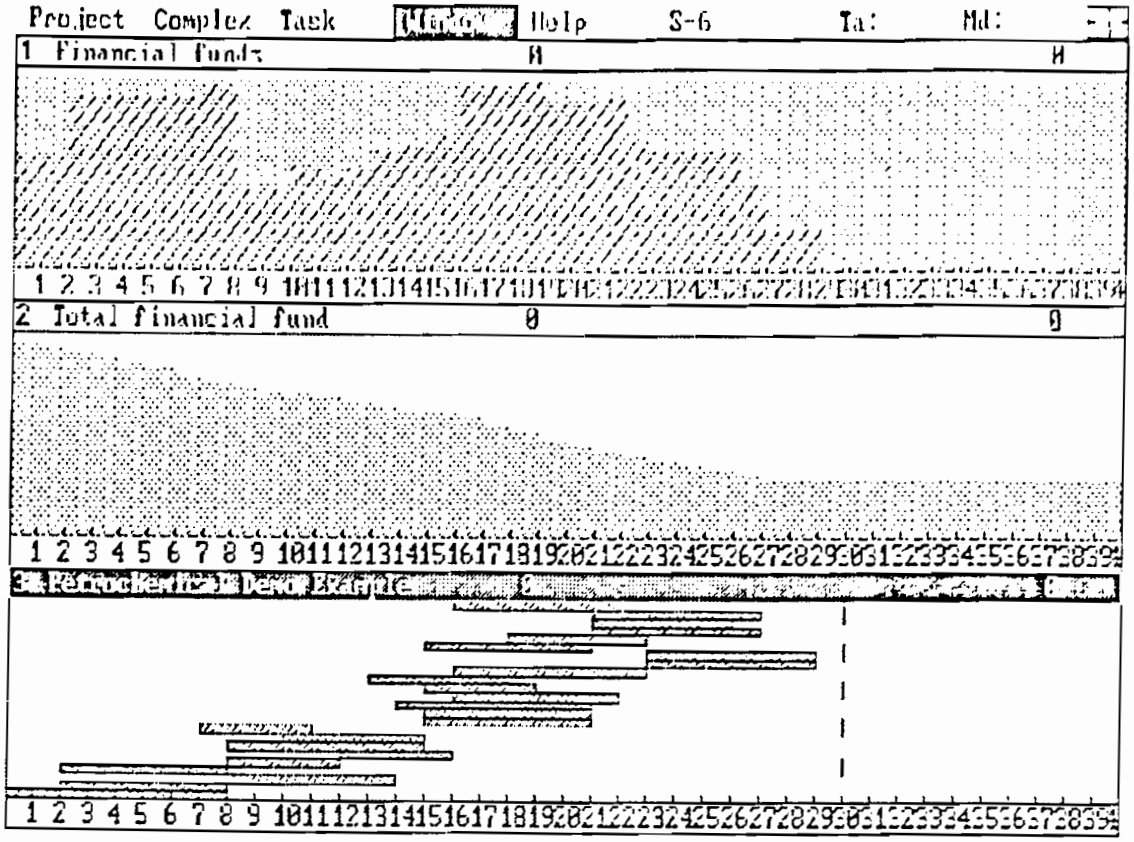


Figure 6: The dynamics of resources utilization.

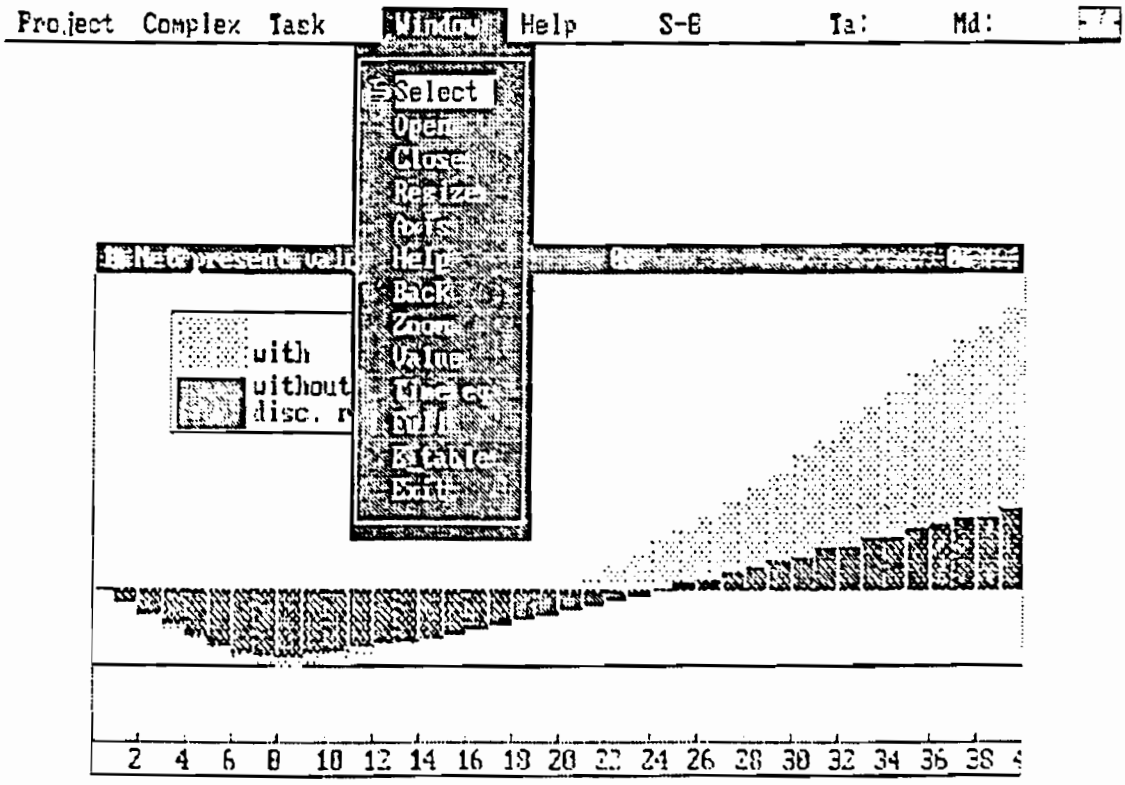


Figure 7: Net Present Value.

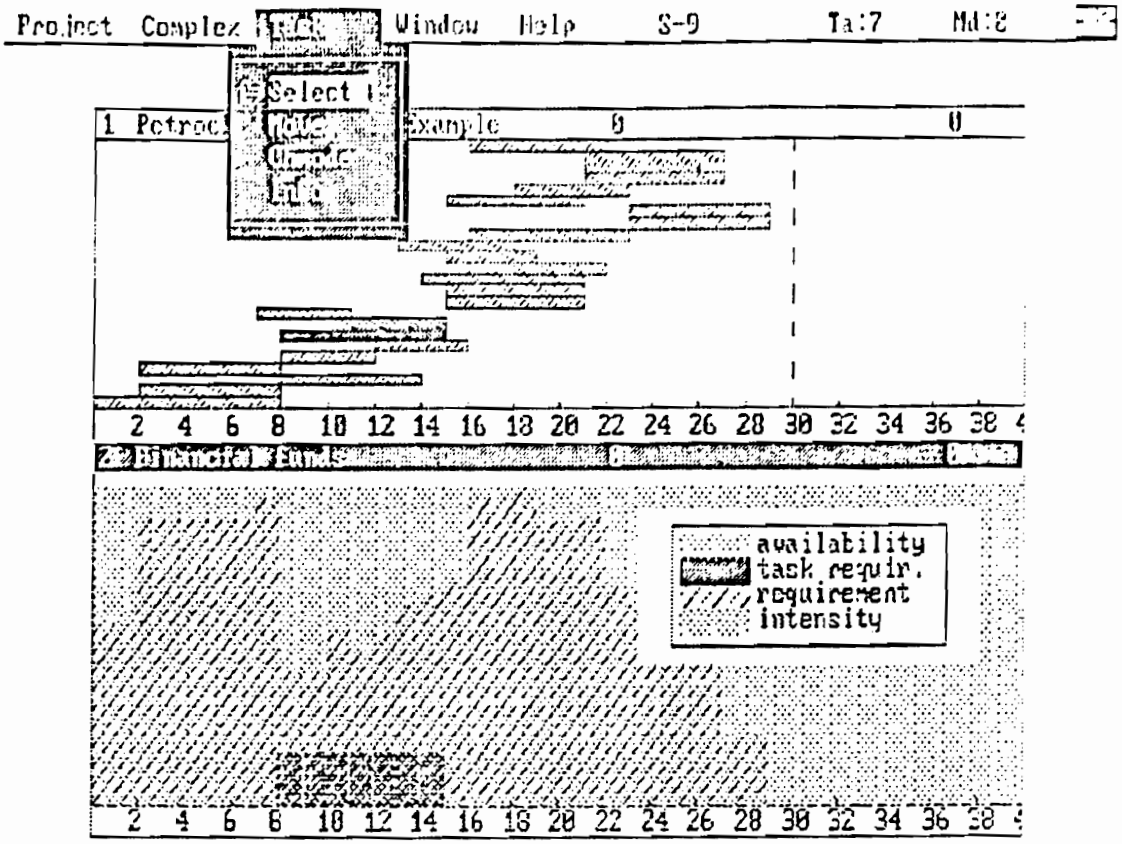


Figure 8: Task submenu operation.

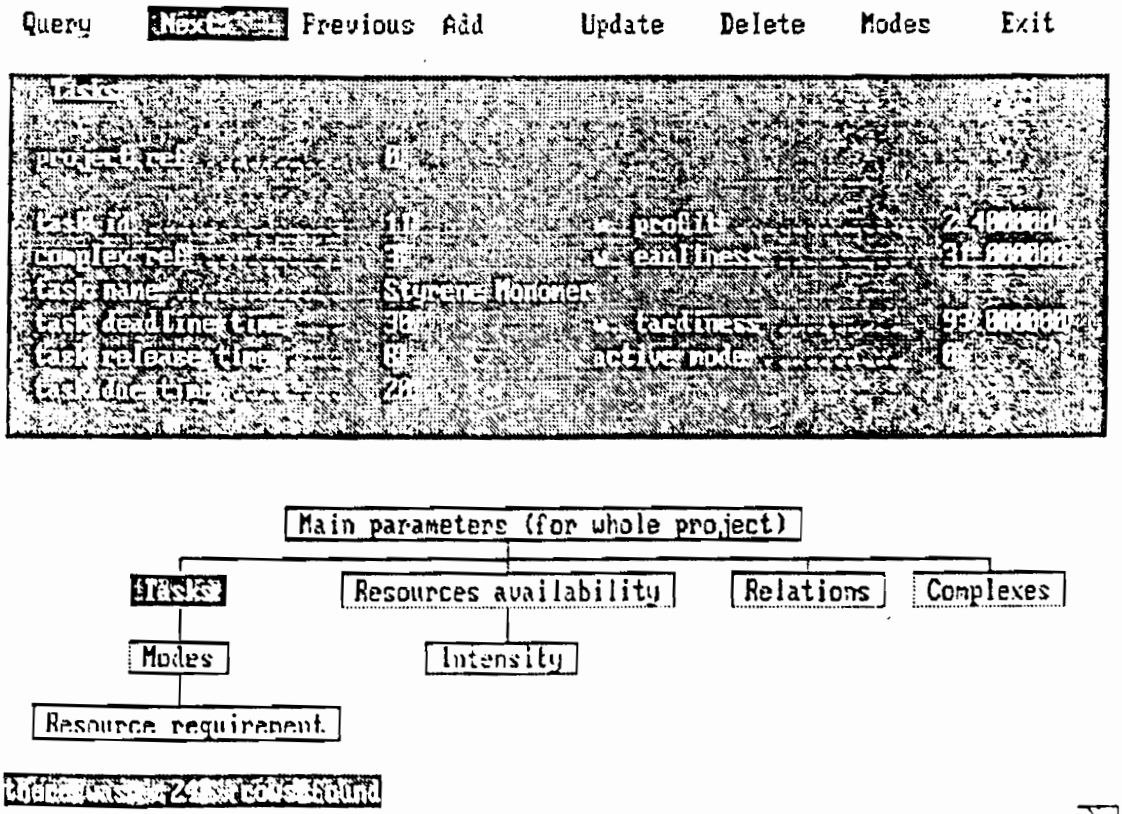


Figure 9: MIPS task data file form.

Application of IAC-DIDAS-L2 to Optimization of Water Releases From a Retention Reservoir During Flood

Andrzej Karbowski
Institute of Automatic Control
Warsaw University of Technology

1 Introduction

The aim of the discussed research was to analyze the possibilities of the application of IAC-DIDAS-L2 package to optimization of the operation of a middle size water reservoir during flood. The special attention was paid to analysis of features of DIDAS while working with on-line data, such as stochastic forecasts of the inflow to the reservoir. As an introduction a simpler, deterministic problem was solved for several sets of data taken from a historical data base. Afterwards, more complicated, 2-stage stochastic problem was formulated and solved.

This article contains the mathematical models of the studied problems, the evidence gathered and conclusions concerning usefulness of the present version of DIDAS to solve the presented problems together with some recommendations for future improvements of this package.

2 Deterministic formulation of the flood control problem

The main objective of the flood control systems is to protect the dam and to reduce damages evoked by high water levels downstream the reservoir. These damages usually depend on the culminant flow in an area situated below the reservoir. With some simplification we may assume that they are proportional to the peak of the release from the reservoir.

Another objective is connected with other functions of the reservoir, such as water supplying and energy production. From their point of view, after flood the reservoir should have as much water as possible.

Denoting: w — storage of the reservoir (in [m^3]), d — inflow to the reservoir (in [m^3/s]) and u — release from the reservoir (also in [m^3/s]) (see Fig. 1), k —current period of time and T — the control horizon, we can formulate the problem as follows:

$$q_1 = \max_{k=1,\ldots,T} u(k) \longrightarrow \min \tag{1}$$

$$q_2 = w(T) \longrightarrow \max \tag{2}$$

$$w(k+1) = w(k) + d(k) - u(k) \tag{3}$$

$$w(1) = w_1 \tag{4}$$

$$w_{\min} \leq w(k) \leq w_{\max}, \qquad k = 1, \ldots, T \tag{5}$$

$$u_{\min} \leq u(k) \leq u_{\max}(w(k)), \quad k = 1, \ldots, T \tag{6}$$

In this formulation equality (3) is simply a continuity (mass balance) equation. Inequalities (5), (6) express the necessity to keep water level between minimal and maximal value and, respectively, to consider water releases from an admissible range. It is worthy to underline, that the upper release constraint in (6) is a function of storage, that comes from gravitational outflow with fully open reservoir gates. For the sake of simplicity we assume that it has the linear form:

$$u_{\max}(w(k)) = \beta \cdot w(k) \tag{7}$$

A formal analysis of the problem (1)–(6), where outcome (2) was replaced with the final condition

$$w(T) = w_{\max} \tag{8}$$

and the continuous-time dynamics was used, can be found in (Karbowski, 1989a, b).

In order to apply DIDAS to solve problem (1)–(6) some transformations were necessary. First of all untypical, minimax outcome was replaced with an additional, artificial variable z and T constraints:

$$u(k) \leq z, \qquad k = 1, \ldots, T \tag{9}$$

Thus, using the matrix notation and denoting decision variables as x, objective outcomes as q (compare (Rogowski and others, 1988)), we can express the problem in the following, DIDAS-acceptable form (we assume that T equals 10):

$$x = [u(1), u(2), \ldots, u(9), w(2), w(3), \ldots, w(10), z]^T \tag{10}$$

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = Cx = \begin{bmatrix} 0, 0, \ldots, 0, 0, 0, 1 \\ 0, 0, \ldots, 0, 0, 1, 0 \end{bmatrix} \tag{11}$$

STATUS: q_1 — min
q_2 — max

constraints:

$$x^{\text{lo}} \leq x \leq x^{\text{up}}$$

$$x^{\text{lo}} = [u_{\min}, u_{\min}, \ldots, u_{\min}, w_{\min}, w_{\min}, \ldots, w_{\min}, 0]^T \tag{12}$$

$$x^{\text{up}} = [\infty, \infty, \ldots, \infty, w_{\max}, w_{\max}, \ldots, w_{\max}, \infty]^T \tag{13}$$

$$y^{\text{lo}} \leq Ax \leq y^{\text{up}}$$

$$A = \left[\begin{array}{cccccccccccccccccc}
\alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & \vdots & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & & & & \vdots & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\beta & 0 & 0 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
 & & & & & & & & & & \vdots & & & & & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}\right] \tag{14}$$

$$y^{\text{lo}} = [\gamma_1, \gamma_2, \ldots, \gamma_9, -\infty, -\infty, \ldots, -\infty]^T \tag{15}$$

$$y^{\text{up}} = [\gamma_1, \gamma_2, \ldots, \gamma_9, \delta, 0, 0, \ldots, 0]^T \tag{16}$$

where:

$$\alpha = 43200 \text{ (twelve hours discretization was taken)}$$

$$\gamma_1 = w_1 + 43200 \cdot d(1) \tag{17}$$

$$\gamma_i = 43200 \cdot d(i), \quad i = 2, \ldots, 9 \tag{18}$$

$d(i)$ is understood here as the mean over 12hours instant inflow (in [m^3/s])

$$\delta = \beta \cdot w_1 \tag{19}$$

In this way we have obtained a two-criteria linear programming problem with 19 decision variables and 27 constraints.

The presented problem was solved for data concerning one of the biggest Polish reservoirs — Rożnów, located in Southern part of Poland on Dunajec, side inflow to the Vistula river. Its parameters are the following: $w_{\min} = 40.3$ mln m^3, $w_{\max} = 171.2$ mln m^3, $\beta = 15 \cdot 10^{-6}$. The initial storage w_1 was assumed to be equal 84.9 mln m^3. As inflow $d(k)$ to the reservoir, several historical floods were taken, namely that of 1958, 1965, 1970, 1972, 1973, 1983 and 1985. One optimization took 10–15 seconds on IBM PC/AT compatible computer with 8 MHz clock.

The results of optimization (for automatic scaling) are presented in Table 1 and Figs. 2–4.

year	culminant inflow	$\hat{q}_1$	$\hat{q}_2$	activity of constraints			
	$[m^3/s]$	$[m^3/s]$	$[m^3]$	u_{min}	u_{max}	w_{min}	w_{max}
1958	2481	971.761	171.2e6	-	+	-	+
1965	1161	470.775	171.2e6	-	-	-	+
1970	3237	1906.401	171.2e6	-	+	-	+
1972	1367	514.66	171.2e6	-	-	-	+
1973	2836	1504.463	171.2e6	-	+	+	+
1983	1401	187.599	171.2e6	-	-	-	+
1985	1463	372.222	171.2e6	-	-	-	+

Table 1

Table 1 shows the optimal solutions for any reference points, which means that in all cases the 'utopia' points belonged to the Pareto sets, which were single-pointed. That should not be surprising, because there is no conflict between criteria (1) and (2). It can be also seen, that the optimization reduced considerably the peaks of the inflows to the reservoir ('cut' the inflows) causing filling up the reservoir at the end of the floods.

The above mentioned degenerated, single-pointed form of the Pareto set caused some problems at the beginning of the work with DIDAS. As it was described in (Rogowski and others, 1988), while automatic scaling the program uses (among other parameters) coordinates of the nadir point. Unfortunately, this point is determined quite inaccurately, as the point of the worst coordinates occured during optimization. In this particular case, nadir point, Pareto set and utopia point were overlapping and the obtained estimates of the nadir points were completely wrong. Because the implemented algorithm is very sensitive to scaling coefficients, in some cases, i.e., for some reference points, the solutions calculated by the program were different than the utopia points, that means that they were wrong. It encouraged authors of the DIDAS program to remove the nadir point from the algorithm and to change the scaling formulas. After these changes the program started to work properly.

3 Stochastic, one-stage formulation of the flood control problem

In actual flood control systems the releases from reservoirs are calculated basing upon current forecasts of the inflow. For example, in the Upper Vistula river system hydro-meteorological services provide during the whole control horizon stochastic, two-variant forecasts:

- so-called synoptic forecasts, which are calculated under the assumptions, that in the nearest future there will be some rainfalls
- pure-hydrologic forecasts, which are determined with the exclusion of rainfalls

These forecasts could be applied in various suboptimal reservoirs' control rules (Malinowski and Karbowski, 1986, Karbowski and others, 1988, Karbowski, 1989c) based on POLF concept (Bertsekas, 1986). They can be also applied in less classical control schemes. In this section, as well as in the next one, two such schemes, making use of multicriteria optimization, will be presented.

The first approach uses two-variant forecasts of the inflow in the framework of 1-stage stochastic control scheme. Let us denote as

d_1 — synoptic forecast of the inflow to the reservoir

d_2 — pure-hydrologic forecast of the inflow to the reservoir

u — proposal of the release trajectory

$$WU = \{(w,u) : w_{\min} \le w \le w_{\max}, \quad u_{\min} \le u \le u_{\max}(w)\} \tag{20}$$

— area of admissible storages and releases (control field)

u_i^r, $i = 1,2$ — realized release trajectory being the result of the projection of the proposed release pattern u on WU when the actual inflow is equal to the forecast d_i (see formula (29))

w_i, $i = 1,2$ — storage of the reservoir when its inflow equals d_i and the release u

Now we can formulate multicriteria 1-stage stochastic flood control problem as follows:

$$q_1 = \max_{k=1,\ldots,T} u_1^r(k) \longrightarrow \min \tag{21}$$

$$q_2 = w_1(T) \longrightarrow \max \tag{22}$$

$$q_3 = \max_{k=1,\ldots,T} u_2^r(k) \longrightarrow \min \tag{23}$$

$$q_4 = w_2(T) \longrightarrow \max \tag{24}$$

$$w_1(k+1) = w_1(k) + d_1(k) - u(k) \tag{25}$$

$$w_1(1) = w_1 \tag{26}$$

$$w_2(k+1) = w_2(k) + d_2(k) - u(k) \tag{27}$$

$$w_2(1) = w_1 \tag{28}$$

$$u_i^r(k) = \chi_{wu}(w_i(k)) \cdot u(k) + [1 - \chi_{wu}(w_i(k))] \cdot d_i(k), \quad i = 1,2 \tag{29}$$

where κ is the characteristic set function, defined as:

$$\chi_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \tag{30}$$

The typical results of projection of the release u on control field WU due to formula (29) are presented in Fig. 5.

Unfortunately, the projection operator (29) is nonlinear and the problem (21)–(29) cannot be solved with the help of linear IAC-DIDAS-L2 system. Probably it could be solved with using another system from DIDAS family — nonlinear IAC-DIDAS-N (see (Kręglewski and others, 1988)).

4 Stochastic, two-stage formulation of the flood control problem

In this formulation we assume, that the control procedure consists of two stages. At the first stage we know that the future inflow is uncertain, more precisely, in our case it may take one of two values d_1 or d_2, but while making the decision we do not know which of them. We assume that this will be known at the second stage, while making decision concerning the remaining part of the control horizon. Hence, at the first stage we

optimize the release for only one control period, while at the second stage for the others. This scheme can be expressed in the following form:

$$q_1 = \max_{k=1,\ldots,T} u_1(k) \longrightarrow \min \tag{31}$$

$$q_2 = w_1(T) \longrightarrow \max \tag{32}$$

$$q_3 = \max_{k=1,\ldots,T} u_2(k) \longrightarrow \min \tag{33}$$

$$q_4 = w_2(T) \longrightarrow \max \tag{34}$$

$$w_1(k+1) = w_1(k) + d_1(k) - u_1(k) \tag{35}$$

$$w_1(1) = w_1 \tag{36}$$

$$w_2(k+1) = w_2(k) + d_2(k) - u_2(k) \tag{37}$$

$$w_2(1) = w_1 \tag{38}$$

$$w_{\min} \le w_1(k) \le w_{\max}, \quad k = 1,\ldots,T \tag{39}$$

$$u_{\min} \le u_1(k) \le u_{\max}(w_1(k)), \quad k = 1,\ldots,T \tag{40}$$

$$w_{\min} \le w_2(k) \le w_{\max}, \quad k = 1,\ldots,T \tag{41}$$

$$u_{\min} \le u_2(k) \le u_{\max}(w_2(k)), \quad k = 1,\ldots,T \tag{42}$$

$$u_1(1) = u_2(1) \tag{43}$$

If the joining condition (43) were absent, the problem (31)–(42) would simply correspond to two independent problems of (1)–(6) type formulated and solved for two different forecasts of inflow d_1 and d_2. Due to this condition we obtain 2-stage stochastic, but other than based on discrete probability density functions (besides, usually it is rather difficult for the operator of the reservoir to determine the probability of a particular inflow forecast) control scheme.

Because the matrix, DIDAS — acceptable form of the discussed problem is very similar to that of the deterministic case presented above, we will omit it here.

As for the deterministic case, this problem was solved for the Roznow reservoir. The same reservoir parameters and data were used. On the same IBM PC/AT-compatible computer one optimization took 20–25 seconds.

The results of optimization are presented in Table 2 and Figs. 6–8.

No.	Reference point				Solution			
	q_1	q_2	q_3	q_4	$\hat{q}_1$	$\hat{q}_2$	$\hat{q}_3$	$\hat{q}_4$
	[m^3/s]	[mln m^3]	[m^3 /s]	[mln m^3]	[m^3 /s]	[mln m^3]	[m^3 /s]	[mln m^3]
1	264.1	171.2	102.4	171.2	290.9	171.2	130.1	171.2
2	270	160	120	170	291.2	171.2	128.2	171.2
3	265	160	120	170	290.9	171.2	130.0	171.2
4	265	160	130	170	289.0	171.2	139.3	171.2
5	265	120	130	165	289.0	171.2	139.3	171.2
6	265	120	140	165	287.1	171.2	148.6	171.2
7	265	120	200	165	276.0	171.2	204.7	171.2
8	265	120	265	165	264.1	171.2	264.1	171.2

Table 2

The first point of the table corresponds to utopia and neutral points (standard projection of IAC-DIDAS-L2 on the Pareto set).

It can be seen from the table, that the outcomes q_2 and q_4, corresponding to maximizing the final storage of the reservoir, have the same value over the whole Pareto set. They have also no influence on the solution, that is, even if they are changed while keeping the reference outcomes q_1 and q_3 constant, the solution (projection on the Pareto set) remains unchanged. The only 'active' outcomes are q_1 and q_3 (culminations of the releases due to various forecasts). By deteriorating q_3 we could approach very close to the optimal (i.e., that of the 'utopia point') value of q_1.

The last experiment was performed to check what is the relation between the usual, Euclidean notion of distance and the distance implied by the reference point methodology. As the starting point the neutral point was used with slightly modified first coordinate ($q_1 = 292.0$ instead of 290.9, $q_2 = 171.2e6$, $q_3 = 130.1$, $q_4 = 171.2e6$) . The solution was (independently of scaling, both for automatic and manually entered scaling coefficients equal 1.):

$$\hat{q}_1 = 292.0, \quad \hat{q}_2 = 171.2e6, \quad \hat{q}_3 = 124.4, \quad \hat{q}_4 = 171.2e6$$

It should be noted, that although the coordinate q_1 of the reference point was bigger on only 1.1, it caused the reduction of the coefficient q_3 as much as on 5.7. It means, that the reference point methodology yields other points than the closest to the reference points in the Euclidean sense (because the neutral point was closer to the reference point than the solution).

5 Conclusions

IAC-DIDAS-L2 is a very interesting and convenient software tool for solving small size multicriteria linear programming problems. In particular, it is very easy to learn and use the basic functions of the program (entering model, problem, reference points, looking over the previous solutions, reviewing their neighbourhood, etc.). However, at present its application to more complicated, real-world engineering problems, such as real-time flood control problem, is rather limited. The reasons are the following:

(i) the optimal control problems are dynamic; they usually use difference state equations to describe the dynamics. In these equations time is the index variable, that is all equality constraints stemming from one state equation (i.e. corresponding to subsequent time instants) can be described by one equation with time as index variable. Unfortunately, IAC-DIDAS-L2 has not such possibilities. To introduce all constraints stemming from the single state equation it was necessary to write down this equation for all time periods separately. Of course it was very cumbersome, time-consuming and exposed to mistakes.

(ii) in engineering problems parameters of the model or other kinds of data (for example current storage of the reservoir, current forecasts of the inflow in our problem) are imported from external ASCII files. The present version of DIDAS does not allow for it (there is a certain indirect possibility to make it by preparing files in IAC-DIDAS-L2 `*.MOD` and `*.PRO` files format, but it is too complicated). The only possibility to enter the data is by keyboard. It is convenient only in these cases when we enter a few numbers.

(iii) in optimal control problems we are mostly interested in the course of control and state variables in time. A simple graphical representation of these functions is always very useful (see (Loucks and others, 1985)). The discussed version of DIDAS has not such possibility because it does not display graphically the decision variables.

(iv) the program failed to solve the problem (1)–(6) with more realistic discretization, which was 4 times bigger than applied in above, both deterministis and stochastic models (e.g. periods were not 12 but 3 hours long)

Of course, it is a matter of discussion, if implemented in DIDAS method of projection the reference points onto Pareto set, which often yields points different than the closest ones in Euclidean sense (see the last part of the previous section) is substantiated. Author has no competence to evaluate it. In any case, the potential user should understand this difference as well as the whole methodology.

In spite of these critical remarks, the most of which stem from the fact that IAC-DIDAS-L2 was projected for other purposes than real-time management of complicated dynamic systems, in the author's opinion it is a very valuable tool, also for engineers and scientists. Usually at the beginning of the design stage simplified problems are solved to understand the nature of the process or the algorithm. In such situations, as an user-friendly optimization tool IAC-DIDAS-L2 can be most helpful. Moreover, it proved to be very easy to extend the graphical possibilities of the program. The specialized version of the program, which realized graphical functions mentioned at point (iii) was prepared in only two days by IAC-DIDAS-L2 authors (see screen figures). Probably other missing functions mentioned above are also easy implementable.

References

Bertsekas, D. P. (1976). Dynamic Programming and Stochastic Control, Academic Press, New York and London.

Karbowski, A., Malinowski, K. and Niewiadomska, E. (1988). A stochastic model for real-time flood control. 33 *Intern. Wiss. Koll.* TH Ilmenau, Germany, October.

Karbowski, A. (1989a). Synthesis of structures and mechanisms of flood control in multireservoir systems, Ph.D. thesis, Warsaw University of Technology.

Karbowski, A. (1989b). Optimal Control of Single Retention Reservoir During Flood; Analytical Solution of Deterministic, Continuous-Time Problems. *Journal of Optimization Theory and Applications*, accepted for publication.

Karbowski, A. (1989c). FC-ROS — Decision Support System for reservoir operators during flood. *Environmental Software*, accepted for publication.

Kręglewski, T., Paczynski, J., Granat, J. and Wierzbicki, A. P. (1988). IAC-DIDAS-N: A Dynamic Interactive Decision Analysis and Support System for Multicriteria Analysis of Nonlinear Models with Nonlinear Model Generator Supporting Model Analysis. Working Paper WP 88-112 of IIASA.

Loucks, D. P., Kindler J. and Fedra, K. (1985). Interactive Water Resources Modeling and Model Use: An Overview. *Water Resources Research*, vol. 21, no. 2, pp. 95–102.

Malinowski, K. and Karbowski, A. (1986). On-line decision support for reservoir operators during flood period. UNESCO/IHP Symposium: DSS and related methods in water resources planning, Oslo.

Rogowski, T., Sobczyk J. and Wierzbicki, A. P. (1988). IAC-DIDAS-L Dynamic Interactive Decision Analysis and Support System Linear Version. Working Paper WP-88-110 of IIASA.

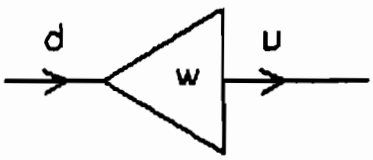


Figure 1: Single reservoir system - notation

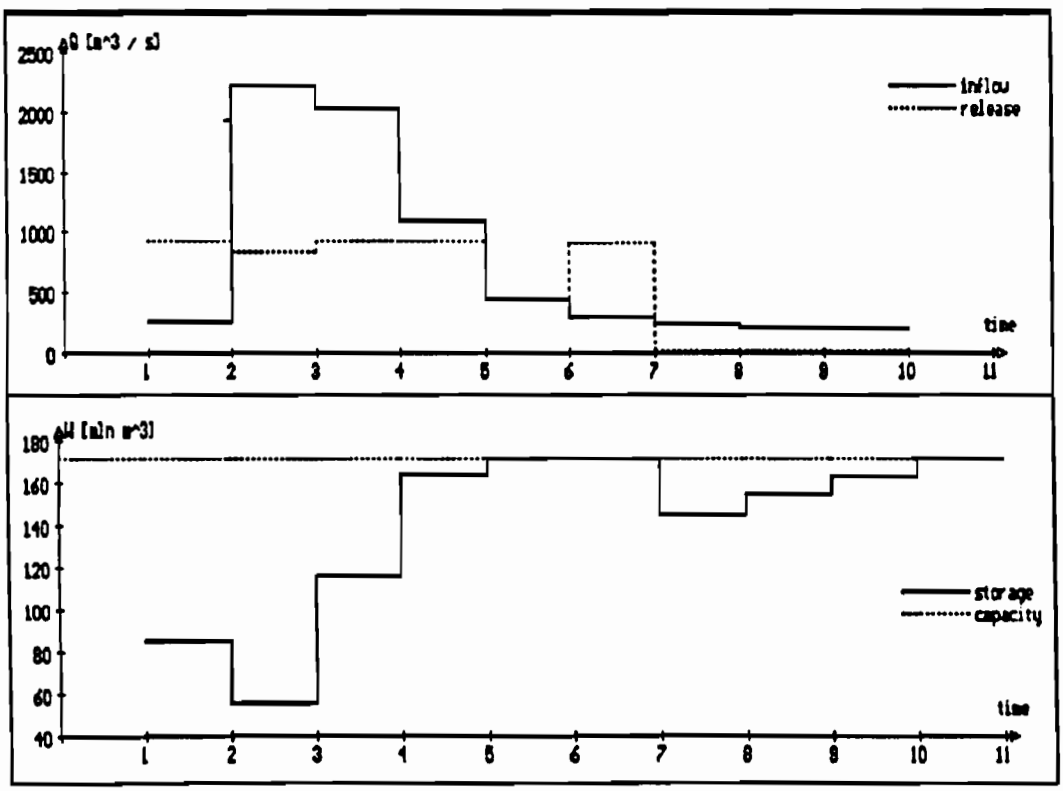


Figure 2: Results of optimization for 1958 year's flood data

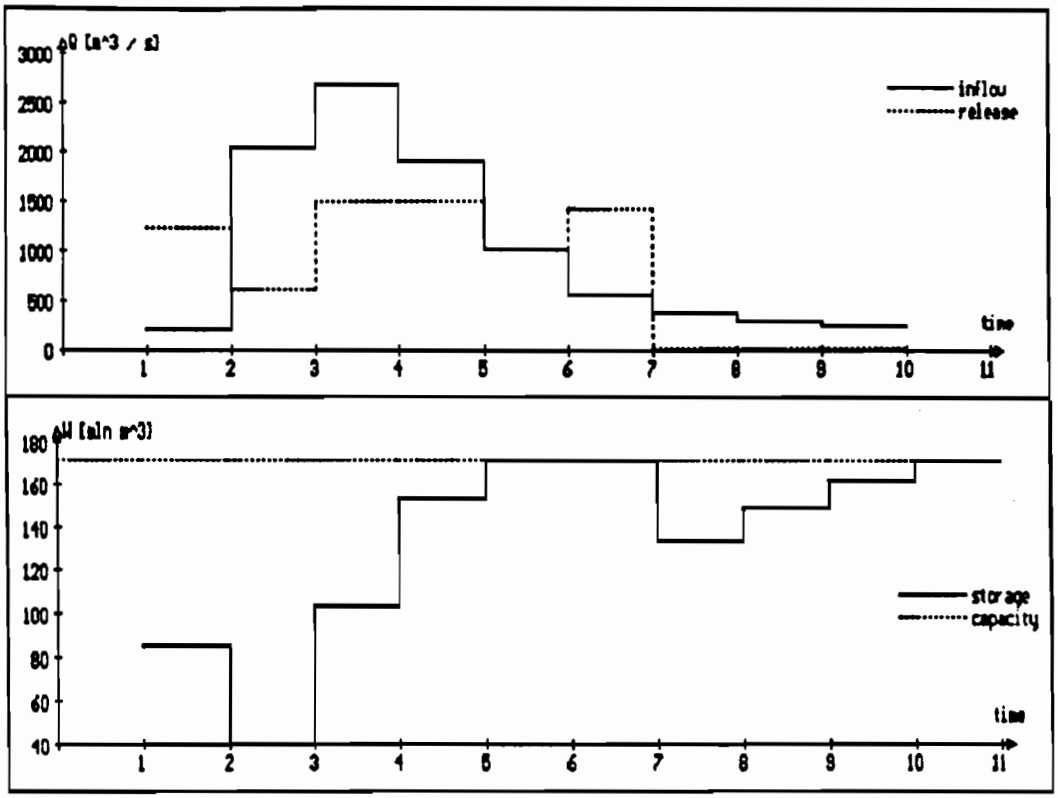


Figure 3: Results of optimization for 1973 year's flood data

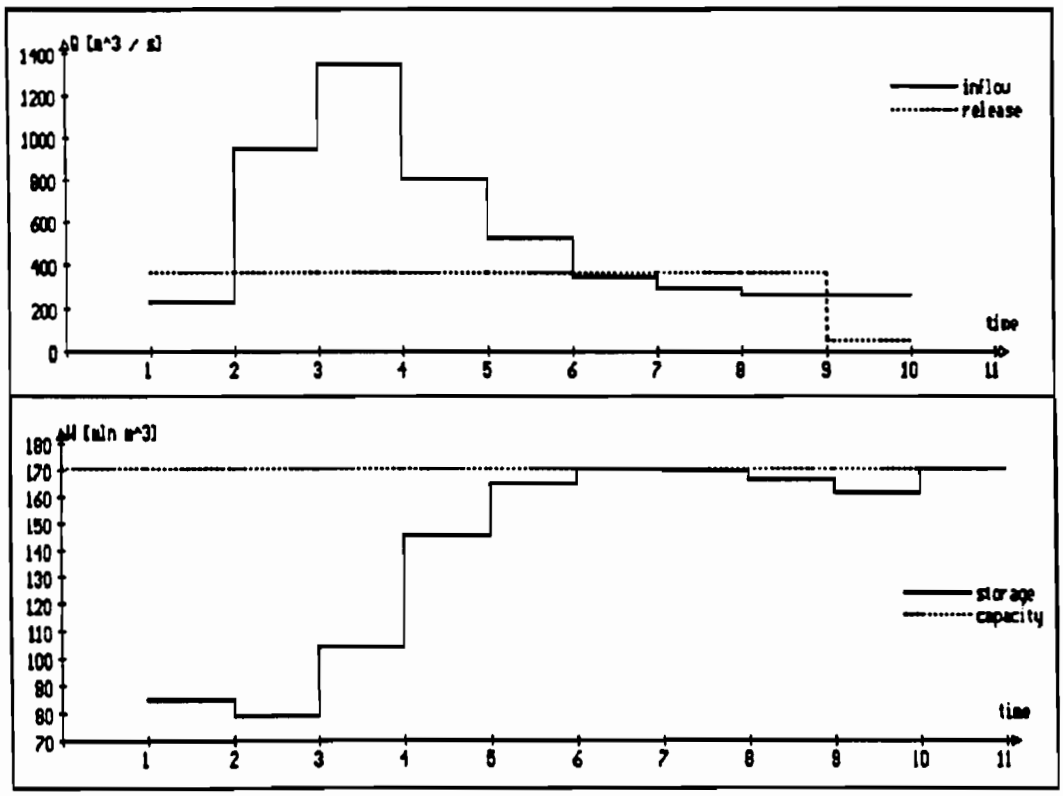


Figure 4: Results of optimization for 1985 year's flood data

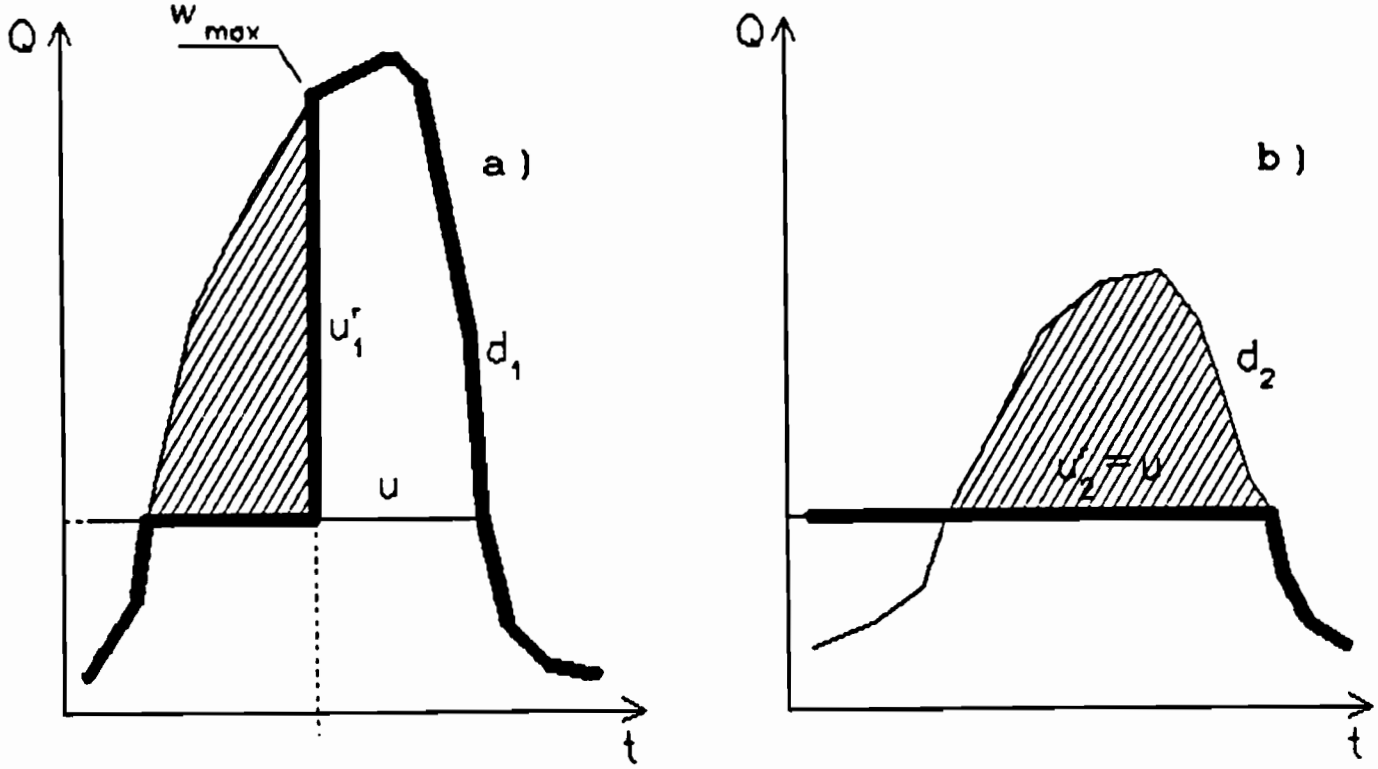


Figure 5: The examples of the projection of the release proposal u on the reservoir control field a)release too small - the reservoir is filled up too early b)release too big - the reservoir storage is not entirely used

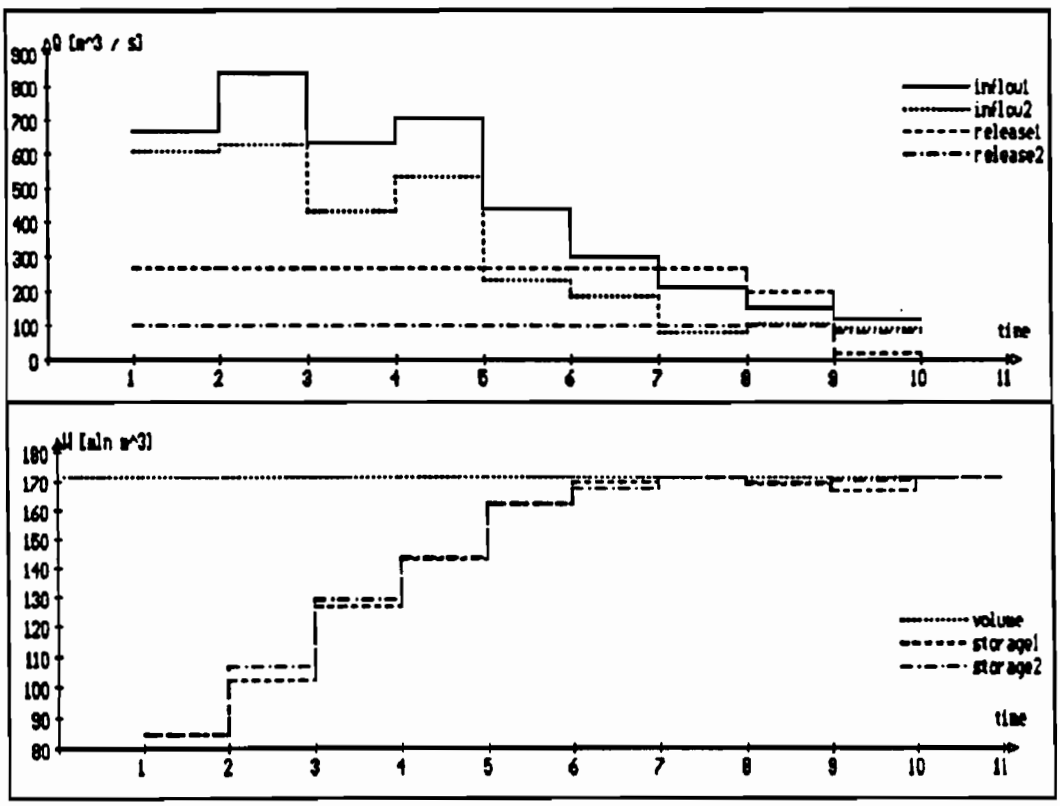


Figure 6: Solution of two independent optimization problems (1) - (6) for two different forecasts of inflow

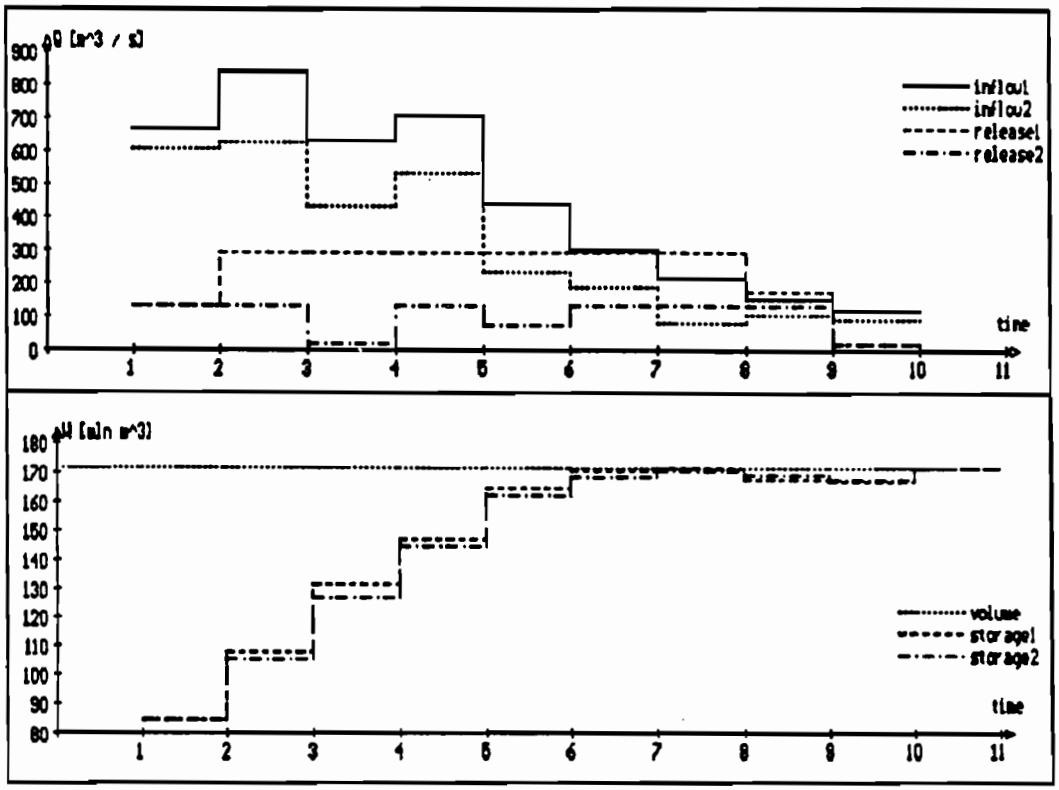


Figure 7: Neutral solution in two-stage stochastic flood control problem (point 1 of Table 2)

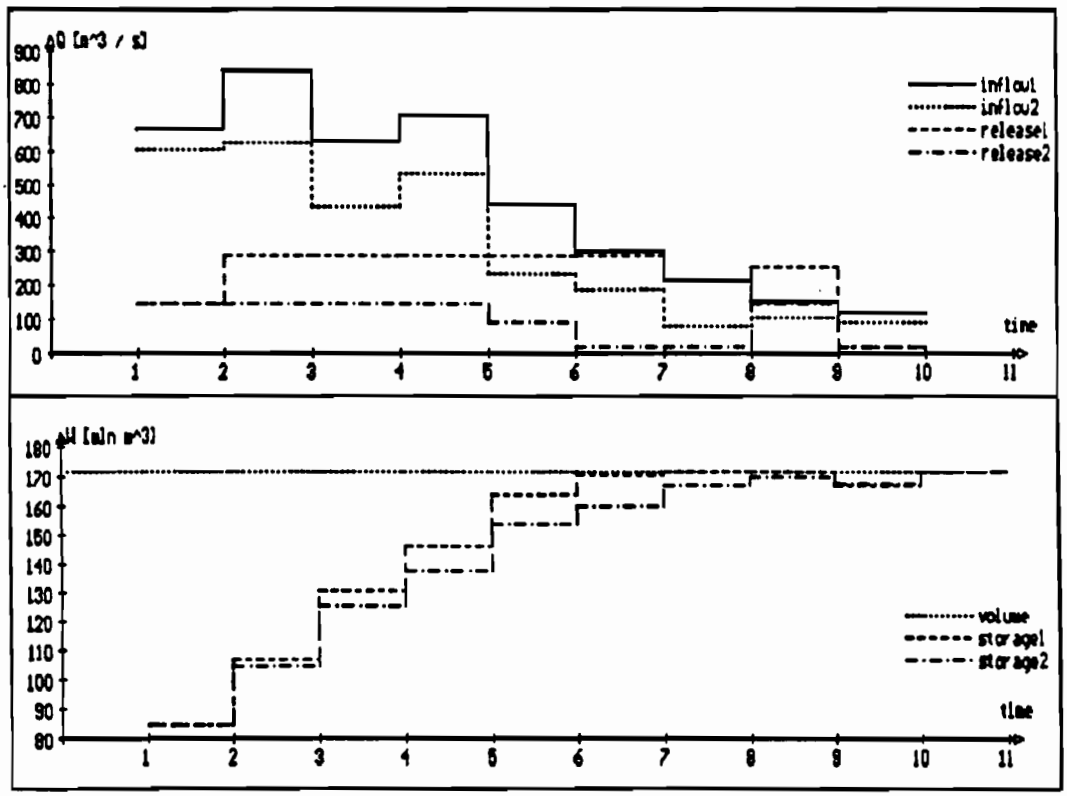


Figure 8: Typical solution of two-stage stochastic flood control problem (point 6 of Table 2)

Multiobjective Design of Multi–Purpose Batch Plant

Tomasz Ryś
Joint System Research Department
of the Institute for Control and Systems Engineering,
Academy of Mining and Metallurgy, Cracow,
and the Industrial Chemistry Research Institute, Warsaw.

Abstract

In the chemical industry a batch plant is understood as a production system in which production process is carried out on "relatively" standardized types of equipment (usually chemical reactors) which can be easily adapted and, if necessary, reconfigured to produce many other different products. It is particularly suitable for low volume products such as pharmaceuticals or other fine chemicals.

In the multi-purpose batch plant, unlike the single or multi-product plant, two or more products can be produced simultaneously.

The problem of the multi-purpose batch plant design can be formulated as follows: a production structure (schedule) and equipment parameters are to be determined according to given production requirements, with respect to chosen criteria set and other technological constraints.

In this paper the following particular assumptions were made:

- demand for products is defined by their range,
- each stage of production process is performed by one type of reactor,
- production strategy is of no-wait type,
- overlapping cycles are allowed,
- parallel reactors work out–of–phase,
- reactor volume is bounded,
- equipment can't be shared by different products processed simultaneously.

For the above case the mixed–integer, nonlinear, bi–criteria model was developed. The investment cost and production profit were selected as objectives.

The sub–optimal and branch and bound solution strategies were proposed for the model solving and satisfactorily tested on theoretical examples published in literature.

1 Introduction

In the chemical industry, two kinds of processes may be distinguished with respect to the chemical reaction type and time conditions. They are:

- continuous processes which usually occur in production of heavy chemicals,

- batch (periodical) processes used in production of fine chemicals.

It would be surprising to find that a significant proportion of the world's chemical production by volume and much larger proportion by value is still performed in batch plants that seem - old–fashioned, in comparison with modern continuous chemical processes. Fine chemicals are used in such fields as:

- animal and human health care,
- photography and reprography,
- flavours and fragrances,
- electronics,
- food additives,
- pesticides.

Batch production is usually carried out on "relatively" standardized types of equipment (chemical reactors, filters, tray or fluidbed driers, storages, etc.) which can easily be adapted and if necessary reconfigured to produce many other fine chemicals for which the actual demand occurs on the market.

The great interest in fine chemicals will call for increased efficiency in the following areas related to the batch plants:

1. **Design Problem** which should determine the equipment sizes and other parameters, production structure, production cycle parameters for the given production requirements, time horizon and technological constraints to satisfy assumed criteria set (e.g. investment cost, profit, etc.).
2. **Production Planning Problem** which determines when, and in what amounts the various products should be manufactured for the given plant specification and some production requirements.
3. **Allocation and Scheduling Problem** decides in which equipment modules the individual tasks of the production processes of the products should be carried out, at what precise moment they should start and when they will be completed for the given production plan.
4. **Retrofitting Problem** which can be understood as a special variant of the first problem, i.e. to answer the question of how to modernize a plant to increase plant efficiency or to satisfy other objectives.

Batch plants may be divided into the following groups with respect to their structure and the way of production:

- **Single-product Batch Plant** which produces only one product. Such plant is developed only in the case of great and stable demand for a product.
- **Multi-product Batch Plant** in which all the products follow "essentially" the same path in the process and use the same equipment (engineering *flow–shop*). Normally in such plant only one product is produced at a time. This plant is usually designed for a fixed set of products.

- **Multi-purpose Batch Plant** which is rather like an engineering *job–shop* where jobs are done on machines as they become available on completion of earlier jobs. Thus, there is no common pattern of movement for the products (successive batches of the same product may take different routes through the process. In such a plant two or more products may be produced simultaneously. The product set for this plant is usually not fixed.

This paper is related to the problem of optimal design of the multipurpose batch chemical plant. Such a problem was formulated for the first time in the paper (Grossmann and Sargent, 1979) in an arbitrary form as a nonlinear program (NLP) but the proposed model can't be easily used (due to its very general formulation) for solving of a particular example. Neither does it discuss all possible items of the real–life batch plant problem.

The general description of the above problem can be presented as follows:

1. Given is a set of batch equipment types. A number of equipment of the particular type can be limited. The equipment types can be grouped into the two classes: batch equipment (reactors) and semicontinuous equipment (pumps, heat exchangers).
2. Given is a set of products with respective demands.
3. Production is realized as multi-stage process
4. To each stage alternative types of reactors can be assigned.
5. Performance of a stage can be limited by time–type conditions.
6. Duration of a stage can depend on batch size
7. A stage may be realized with parallel reactors working in– or out–of– phase.
8. Reactor volumes are standardized.
9. Inter–stage storages are allowed.
10. Equipment can be shared by different products.
11. Production horizon is established.

The problem is to determine production structure and equipment parameters with respect to criteria set (e.g.):

- equipment investment cost,
- production profit,
- energy cost,
- waste treatment cost,
- storage cost,
- buildings investment and use cost,
- supplementary equipment cost,
- employment cost,

- economic efficiency.

Such a general case of the design of multi-purpose batch plant was neither solved nor even formulated as a multicriteria problem. It comes from the complexity of the *job–shop* like problems. In turn, the case of multi-product batch plant design (*flow-shop*) was examined much more deeply (Birewar and Grossmann, 1989; Karimi and Reklaitis, 1985; Ku and Karimi, 1988; Kuriyan, Reklaitis and Joglekar, 1987; Lázaro and Puigjaner, 1985; Rippin, 1983; Sparrow, Rippin and Forder, 1974; Sparrow, Forder and Rippin, 1975; Suhami and Mah, 1981; Takamatsu, Hashimoto and Hasebe, 1979; Wellons and Reklaitis, 1989; White, 1989).

At the current stage all the proposed methods (Klossner and Rippin, 1984; Kopytowski, 1986; Suhami and Mah, 1982; Vaselenak, Grossmann and Westerberg, 1987) of the optimal design of multi-purpose batch plant are related to the plants in which the types of equipment used for each product are specified. In this case products do not use the same equipment in the plant and, therefore, can be separated into partitions for parallel processing with restriction that all items of each batch stage are devoted only to one product at a time. Additionally, the following hidden assumptions were also made:

- production strategy is of no-wait type,
- overlapping cycles are allowed,
- parallel reactors work out–of–phase,
- equipment volume is bounded,
- storage problems are omitted,
- processing time as well as demand for products are constant,
- equipment investment cost is only one objective which should be minimized.

In this paper an extension of the above type of problem is proposed to the case of bi-criteria analysis. The investment cost and production profit were selected as objectives.

In the next parts of the paper an adequate mathematical model is formalized and then solved on the basis of a particular example.

2 Model Formulation

2.1 What is a "product configuration" ?

At it was mentioned in the previous section, the problem considered of a multi-purpose batch plant optimal design refers to the situation when products can be separated into groups for parallel processing. This separation determines the production structure (schedule, manufacturing order) and is known as a *product configuration* (Suhami and Mah, 1982). The grouping process of the products is a central feature of the design of this type of plant because, as it will be shown in the next sections, it determines some of time-type model constraints. The number of possible product configurations for a given set of products and set of equipment types can be large, which for the optimal design leads to a difficult combinatorial problem.

To solve the above problem generally two approaches were proposed. In the first concept a two-stage solving algorithm is assumed:

1. At the first stage feasible *product configuration* is generated

2. At the second stage a mixed-integer nonlinear program is solved

The above procedure examines all possible product configurations (Klossner and Rippin, 1984) or applies a heuristic method to pre-eliminate some of them (Suhami and Mah, 1982). The main disadvantage of these procedures is that many different alternatives may have to be examined (each requires the solution of a mixed-integer nonlinear program). The bi-criteria approach to the multi-purpose batch plant design based on this procedures was presented in (Ryś, 1989) where the economic efficiency was used as categorization coefficient of feasible *product configurations.*

The second approach (Vaselenak, Grossmann and Westerberg, 1987) is based on the *superstructure* and *multiperiod* model concept. The superstructure embeds all of the groups of products that are candidates for the optimal *product configuration* and with it a multiperiod model is formulated and then condensed into a merged formulation. This procedure requires a single mixed-integer nonlinear program to be solved to obtain the plant design with the minimum equipment investment cost.

For the purpose of bi-criteria optimal multi-purpose batch plant design the above approach will be adapted and extended.

2.2 Superstructure Determination

To generalize the algorithm of the superstructure determination it will be assumed that the multi-purpose batch plant consists of:

1. a set M of the batch equipment types: $M = \{1, 2, \ldots, m\}$.

2. a set N of the products which will be produced in the plant: $N = \{1, 2, \ldots, n\}$.

3. a technology, given as a relation $\Omega \subset N \times N$. If $< i \in N, j \in M > \in \Omega$ it means that equipment type j is necessary for manufacturing of the product i. This relation determines the following sets C_i:

$$\forall_{i \in N} \ \exists \ C_i = \{j : < i, j > \in \Omega\}$$

$$|C_i| = c_i \leq m$$

If $c_i = m$ it means that product i can't be grouped with any other. In the considered type of the batch plant the following assumption must be also satisfied:

$$\exists_{i_1, i_2 \in N} \ C_{i_1} \cap C_{i_2} = \emptyset$$

Our task is to determine groups $G_t \subset N$ and number of periods T ($t = 1, 2, \ldots, T$) of possible product configurations, that is, the superstructure $S = \{G_1, G_2, \ldots, G_T\}$ of groups of the products which can be manufactured simultaneously. To find S the following algorithm is to be proposed:

Step 1 Set $l = 2$.

Determine all binary combinations (groups G_t^l and T^l) of products i_1, i_2 which do not share the same equipment types, that is:

$$G_t^l = \{i_1, i_2\}, C_{i_1} \cap C_{i_2} = \emptyset, \quad t = 1, 2, \ldots, T$$

The one-product groups can be excluded from the analysis because sequential production of all products forces longer production time when equipment set is fixed or larger equipment requirement for a fixed production horizon.

Step 2 $l = l + 1$.

$\forall\ G_t^{l-1}$ determine:

$$G_t^l = \{G_t^{l-1} \cup \{i^l \in N\}\} : i^l \notin G_t^{l-1},$$

$$\forall_{i^{l-1} \in G^{l-1}}\ \ C_{i^{l-1}} \cap C_{i^l} = \emptyset$$

If no G_t^l can be created set $l = l - 1$ and follow **Step 4**.

Step 3 Delete each $G_t^{l-1} \subset G_t^l$, re-numerate G_t indexing G_t^l and all undeleted G_t^{l-1} sequentially (in any order) and determine current number of periods T^l. Go to **Step 2**.

Step 4 Set $T = T^l$ and $S = \{G_1^l, G_2^l, \ldots, G_T^l\}$. l represents maximal number of elements in G_t groups.

The superstructure S allows to formulate the multiperiod model the idea of which assumes that production is realized in T periods of time. The length of each t period as well as the length of the total processing time of each product which belongs to the group G_t, are the model variables. If any of these lengths has the zero value it means that the period is eliminated from the production structure or the product will be manufactured in an other period.

2.3 Example of the Superstructure Determination

The above algorithm will be illustrated by the well-known example problem of (Suhami and Mah, 1982) with the data as follows:

$n = 7$ products, $m = 10$ batch equipment types.
Technology: $C_1 = \{8\}$, $C_2 = \{3,4,7,9\}$, $C_3 = \{3,8,9,10\}$, $C_4 = \{4,5,7\}$, $C_5 = \{1,5,9\}$, $C_6 = \{3,6,7,10\}$, $C_7 = \{1,2,5,8\}$.

Run of the algorithm:

Step 1 $l = 2$. The two-elements groups:

$$G_1^2 = \{1,2\},\ G_2^2 = \{1,4\},\ G_3^2 = \{1,5\},\ G_4^2 = \{1,6\},\ G_5^2 = \{2,7\},$$
$$G_6^2 = \{3,4\},\ G_7^2 = \{5,6\},\ G_8^2 = \{7,8\}.$$

The current number of periods $T^2 = 8$.

Step 2 $l = 3$. Only one three-element group can be created: $G_1^3 = \{G_3^2 \cup \{6\}\}$

Step 3 The following groups are to be deleted: $G_3^2 \subset G_1^3, G_4^2 \subset G_1^3, G_7^2 \subset G_1^3$.

The groups after re-numeration: $G_1^3 = \{1,2\}$, $G_2^3 = \{1,4\}$, $G_3^3 = \{1,5,6\}$, $G_4^3 = \{2,7\}$, $G_5^3 = \{3,4\}$, $G_6^3 = \{7,8\}$. $T^3 = 6$.

Step 2 $l = 4$. No G_t^4 can be created. $l = 3$.

Step 4 $T = T^3 = 6$ and $S = \{G_1^3, G_2^3, G_3^3, G_4^3, G_5^3, G_6^3\}$.

2.4 Bi-Criteria, MINLP Model Formulation

In this section the bi-criteria, mixed integer nonlinear program (MINLP) model for optimal design of multi-purpose batch plant is formulated. First, the symbols are categorized and defined and then model equation and criteria are presented.

SETS AND INDEXES:

$j \in M$	Batch equipment type. $\|M\| = m$.
$i \in N$	Product to be manufactured in the plant. $\|N\| = n$.
$C_i \subset M$	Set of equipment types which are necessary for i product manufacturing (see section 2.2).
$U_j \subset N$	Set of products which are using equipment type j in the production process. $$\forall_{j \in M} \ \exists \ U_j \ = \ \{i : \ j \in C_i\}$$
S	The superstructure determined by algorithm presented in the section 2.2. To short cut notation, the index l (level) of the groups G_t^l will be omitted what allows to define S as: $$S = \{G_1, G_2, \ldots, G_T\}$$
$t \in \{1,2,\ldots,T\} \Leftarrow S$	time periods determined by S.
s_i	Set of time periods t associated with i defined as follows: $$s_i \ = \ \{t : \ i \in G_t\} \ , \ \ G_t \in S \ , \ \ t = 1,2,\ldots,T$$

VARIABLES:

q_i^t	The quantity of product i that is manufactured in each time period t.
b_i^t	The number of batches of product i in the period t.
B_i	The batch size of the product i.
Q_i	The total amount of product i (production goal).
v_{ij}	The capacity needed for one batch of product i in an equipment type j.
δ_{ij}	The processing time of a batch of product i on an equipment type j.
V_j	The volume of an equipment type j.
$1 \leq P_j \in Z$	The number of units operating independently in parallel and out of phase associated with an equipment type j (production stage j). P_j is restricted to positive integer values (Z set).

T_{Li}	The time required for the manufacture of a batch of product i.
T_t	The length of production period t.

PARAMETERS:

P_j^L, P_j^U	Lower and upper bound for a number of units type j operating in parallel.
V_j^L, V_j^U	Lower and upper bound of an equipment type j.
Δ_{ij}, c_{ij}, γ_{ij}	Data defining processing time function δ_{ij} (see equation (9)).
S_{ij}	The product size factor. Allows to calculate the capacity necessary for one batch of product i in an equipment type j based on nominal batch size.
Q_i^L, Q_i^U	Lower and upper bound of production goal for the product i.
α_j, β_j	Parameters for equipment investment function of type j.
p_i	Unit price of a product i.
r_i	A price of raw materials used for manufacturing of unit of a product i.
a	Investment cost amortization coefficient.
o	Annual batch equipment operating cost coefficient given for total plant capacity unit.
H	Production horizon.

CONSTRAINTS AND EQUATIONS:

The quantity of product that is made in each production time period:

$$q_i^t = b_i^t B_i , \quad i \in N, \quad t \in s_i \tag{1}$$

The total amount of product that is manufactured must meet its production goal range:

$$\sum_{t \in s_i} q_i^t = \sum_{t \in s_i} b_i^t B_i \geq Q_i , \quad i \in N \tag{2}$$

$$Q_i^L \leq Q_i \leq Q_i^U , \quad i \in N \tag{3}$$

The capacity needed for one batch of a product on a particular equipment type:

$$v_{ij} = S_{ij} B_i , \quad i \in N, \; j \in C_i \tag{4}$$

$$B_i \geq 0 , \quad i \in N \tag{5}$$

The volume of any equipment type should accommodate the maximum processing volume for all the products using it:

$$V_j = max\{v_{ij}\} , \quad j \in M, \; i \in U_j$$

This equation can be written as the following constraint (because of modeling purposes):

$$V_j \geq v_{ij} , \quad j \in M, \; i \in U_j \tag{6}$$

The volume of particular equipment must be selected from specified range:

$$V_j^L \leq V_j \leq V_j^U , \quad j \in M \tag{7}$$

The number of parallel units is also both side limited:

$$P_j^L \leq P_j \leq P_j^U , \quad j \in M \tag{8}$$

The processing time of a product batch on particular equipment type is given as the following function (refer to (Grossmann and Sargent, 1979)):

$$\delta_{ij} = \Delta_{ij} + c_{ij} B_i^{\gamma_j} , \quad i \in N, \; j \in C_i \tag{9}$$

The time required to produce a product batch is the maximum stage (associated with specified equipment type) cycle time of the stages in the technology path. In fact it represents the time between successive batches and from time to time is called a limiting cycle. The below equation was introduced by (Sparrow, Forder and Rippin, 1975) and is a good approximation of the limiting cycle for constant processing time ($c_{ij} = 0$):

$$T_{L_i} = max\{\frac{\delta_{ij}}{P_j}\} , \quad i \in N, \; j \in C_i$$

This equation can be replaced for modeling purposes by the following set of inequalities:

$$T_{L_i} \geq \frac{\delta_{ij}}{P_j} , \quad i \in N, \; j \in C_i \tag{10}$$

The production time period is defined as follows:

$$T_t = max\{b_i^t T_{L_i}\} , \quad t = 1,2,\ldots,T , \; i \in G_t$$

and it can be rewritten as:

$$T_t \geq b_i^t T_{L_i} , \quad t = 1,2,\ldots,T , \; i \in G_t \tag{11}$$

The sum of the lengths of above periods must not exceed available production horizon:

$$\sum_{t=1}^{T} T_t \leq H \tag{12}$$

CRITERIA:

The equipment investment cost[1]:

$$min \mapsto f_1 = \sum_{j=1}^{m} \alpha_j V_j^{\beta_j} P_j. \tag{13}$$

The production profit:

$$max \mapsto \sum_{i=1}^{n} (p_i - r_i)Q_i - af_1 - o \sum_{j=1}^{m} V_j P_j, \tag{14}$$

The model described by equations (1)–(14) corresponds to a bi-criteria MINLP formulation. In the general case, because of its large size, this model can't be solved easily and the existence of unique optimal value is not necessarily guaranteed. Therefore, so called *merged* formulation of such model type was proposed by (Vaselenak, Grossmann and Westerberg, 1987) in which the total processing times and total number of batches for a product were used as variables instead variables associated with each production time period.

The total number of batches of each product can be defined as follows:

$$b_i = \sum_{t \in s_i} b_i^t, \quad i \in N$$

and then constraint (1) comes to a new form:

$$b_i B_i \geq Q_i, \quad i \in N \tag{15}$$

For the merged model formulation the new variable T_i, the total product i processing time is introduced and constraint (11) is replaced by:

$$T_i \geq b_i T_{L_i}, \quad i \in N \tag{16}$$

and variables T_i are neglected. The horizon constraint (12) is replaced by equivalent constraints expressed in terms of the new variables T_i. The general algorithm of the horizon constraints derivation for the merged model formulation, based on a determined problem superstructure, is presented in (Vaselenak, Grossmann and Westerberg, 1987).

To complete this section the new model formulation will be presented what allows to describe the problem in question as:

Choose: $b_i,\ B_i,\ Q_i,\ V_j,\ P_j,\ T_{L_i},\ T_i$

Subject to:

$$b_i B_i \geq Q_i, \quad i \in N \tag{17}$$

$$B_i \geq 0, \quad i \in N \tag{18}$$

$$Q_i^L \leq Q_i \leq Q_i^U, \quad i \in N \tag{19}$$

$$V_j \geq S_{ij} B_i, \quad j \in M,\ i \in U_j \tag{20}$$

$$V_j^L \leq V_j \leq V_j^U, \quad j \in M \tag{21}$$

[1]This formula introduced by (Bauman, 1964) is commonly used as investment cost estimate

$$T_{L_i} \geq \frac{\Delta_{ij} + c_{ij} B_i^{\gamma_j}}{P_j}, \quad i \in N,\ j \in C_i \tag{22}$$

$$P_j^L \leq P_j \leq P_j^U, \quad j \in M,\ 1 \leq P_j^L,\ P_j^U,\ P_j \in Z \tag{23}$$

$$T_i \geq b_i T_{L_i}, \quad i \in N \tag{24}$$

$$S \mapsto f_k(T_i) \leq H, \quad i \in N,\ k = 1, 2, \ldots, K \tag{25}$$

where: K symbolizes number of horizon constraints generated from the problem superstructure.

To satisfy: criteria defined through equations (13) and (14).

At it was shown (Grossmann and Sargent, 1979) for a single-criteria case, the continuous relaxation of the above formulation can be transformed to a geometric program. It proves the existence of a unique optimal solution.

3 The Model Solving Technique

3.1 Multiobjective Problem

The well known (Hwang and Masud, 1979) *bounded objective* method will be used to solve multicriteria problem described by equations (17) – (25). Because the criteria set is composed of two elements ((13),(14)) only the method allows also to examine easily the Pareto set.

If f_2 (14) will be chosen as the objective the set of constraints (17)–(25) should be completed by the following inequality:

$$f_1 \leq RHS(f_1) \tag{26}$$

where: RHS defines right hand side value for the above constraint.

In the opposite case, one should define the following constraint:

$$f_2 \geq RHS(f_2) \tag{27}$$

The nonlinear problem will be solved with the use of the computer package MINOS (Murtagh and Saunders, 1983) which is bases on a *reduced-gradient* algorithm in conjunction with a *quasi-Newton* algorithm with respect to nonlinear objective function and *projected augmented Lagrangian* algorithm with respect to nonlinear constraints.

3.2 Mixed Integer Problem

It is not possible with the MINOS package to solve problems with integer restrictions on variables. Therefore additional procedures must be implemented to solve the model of multi-purpose batch plant design. Three procedures which may be applied to solve a mixed integer problem are proposed below.

FTI Procedure

This procedure uses a simple mathematical "trick" — forcing to integer (FTI) the P_j model variables. It consists of the following steps:

Step 1 Take off the integer restrictions from the problem (17)-(25) that is:

$$\forall_{j \in M} \quad P_j \in \Re^+$$

Step 2 Solve the currently formulated problem. If the obtained solution $(*)$ is optimal and $\forall_{j \in M} \quad P_j^* \in Z$ stop the procedure, otherwise follow **Step 3**.

Step 3 Introduce the following constraints to the problem:

$$\sum_{P_j^* \notin Z,\ j \in M} (\lceil P_j^* \rceil - P_j)\,(P_j - \lfloor P_j^* \rfloor) = 0$$

$$\forall_{P_j^* \notin Z,\ j \in M} \quad \lfloor P_j^* \rfloor \leq P_j \leq \lceil P_j^* \rceil$$

where:

$\lceil \bullet \rceil$ – upper integer rand of a real number

$\lfloor \bullet \rfloor$ – lower integer rand of a real number

and go to **Step 2**.

The above procedure doesn't guarantee that the obtained solution is optimal but it provides an efficient way to locate a feasible integer solution which should be close to the relaxed solution. In most testing instances it was possible to get the integer solution with one run in addition to the relaxed solution.

This procedure can also fail to find a feasible solution. In this case the branch and bound (Reingold, Nievergelt and Deo, 1977) algorithm may be necessary to obtain an integer solution.

BB Procedure

Because a set of integer restricted variables is finite (P_j's are bounded) branch and bound (BB) procedure can be adapted without crucial meaning of a branch rule. If we assume that equation (14) will be maximized criteria and (13) will complete the constraints set (26) the following general BB rules can be formulated:

- breadth first search method is used to move through the BB tree,
- a vertex of BB search tree is closed if:
  1. the current solution (f_2^*) has infeasible status
  2. the known discrete solution (f_2^D) is better than the current solution, that is:

$$f_2^D \quad \geq \quad f_2^*$$

- a vertex with the best solution is selected to start a new branch on the current level of the BB tree.

BB procedure is an expensive alternative to FTI but it allows to find the optimal integer solution.

i	Q_i^L (kg)	Q_i^U (kg)
1	270000.0	330000.0
2	135000.0	165000.0
3	180000.0	220000.0
4	171000.0	209000.0
5	126000.0	154000.0
6	154800.0	189200.0
7	95400.0	116600.0

Table 1: Production Goals Range

BB-2 Procedure

This procedure is a conjunction of FTI and BB procedures. FTI is used to find a starting feasible integer solution for BB algorithm and that usually speeds-up branch and bound method.

The comparison of the above procedures based on a particular example is presentmulti-purpose batch plant and to compare the different solving techniques. In the literature this example is treated as corresponding to the large plant.

The superstructure S generated in section 2.3 allows to derive the following horizon constraints (equation (25)):

$$
\begin{aligned}
T_1 + T_3 + T_7 &\leq H \\
T_2 + T_3 + T_5 &\leq H \\
T_2 + T_3 + T_6 &\leq H \\
T_2 + T_4 + T_5 &\leq H \\
T_2 + T_4 + T_6 &\leq H \\
T_3 + T_5 + T_7 &\leq H \\
T_4 + T_5 + T_7 &\leq H
\end{aligned}
$$

The data for this problem are shown in tables 1 and 2. In addition, the following values were assigned to the rest of the model parameters:

$H = 6200$ h; $\forall_{j \in M}$: $\alpha_j = 250$, $\beta_j = 0.6$, $P_j^L = 1$, $P_j^U = 3$, $V_j^L = 250$, $V_j^U = 10000$; $\forall_{i \in N,\ j \in M}$ $c_i j = 0$; $\forall_{i \in N}(p_i - r_i) = 5$; $a = 1.53$, $o = 105$.

The MINLP type 26 model formulation involves 70 constraints (including 39 nonlinear) and 55 variables (including 41 nonlinear).

i	Δ_{ij} (h / batch) S_{ij} ($\frac{l}{kg}$ / batch)									
	1	2	3	4	5	6	7	8	9	10
1								7.456 5.404		
2			7.143 9.768	2.595 1.125			3.974 3.205		5.719 3.304	
3			4.36 8.065				2.554 4.62	7.318 4.529	2.397 8.163	
4				2.404 1.922	9.987 9.415		6.758 4.833			
5	6.534 9.422				5.516 2.653				1.932 5.982	
6			1.269 3.174			2.005 2.895	5.469 5.731			7.725 3.587
7	6.855 3.757	7.326 5.64			5.062 9.381			6.65 6.418		

Table 2: S_{ij} and Δ_{ij} for the example of (Suhami and Mah, 1982).

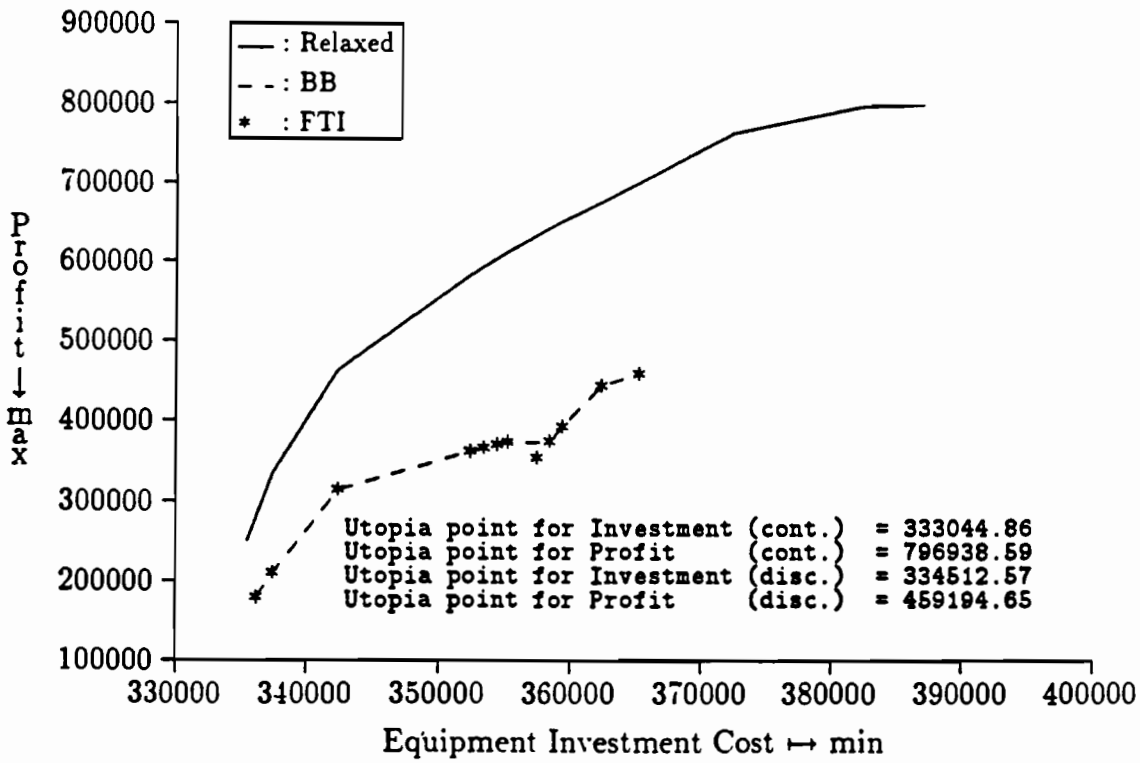


Figure 1: Pareto Set Approximation

j	P_j	V_j
1	1.0	3643.04021
2	1.0	2808.23731
3	1.0	6252.90955
4	1.0	1279.99311
5	1.0	6270.10152
6	1.0	1625.88267
7	1.0	3218.62991
8	1.0	5098.42099
9	1.0	2326.43176
10	1.0	4193.12486

Table 3: Solution for $RHS(f_1) = 335000$ (products)

Approximations of the Pareto set as well as the utopia points for relaxed and discrete problems are presented in the figure 1 prepared on the basis of 14 optimization experiments. It is obvious, that for the integer solution the Pareto set is composed of points. The curve on the figure for the results obtained with the use of BB procedure shows only point layouts (not the approximation). Analysis of this figure leads to a surprising remark that only in one case the results obtained through FTI procedure are different from calculated by BB algorithm. It means that in most cases FTI procedure brings in an optimal integer solution.

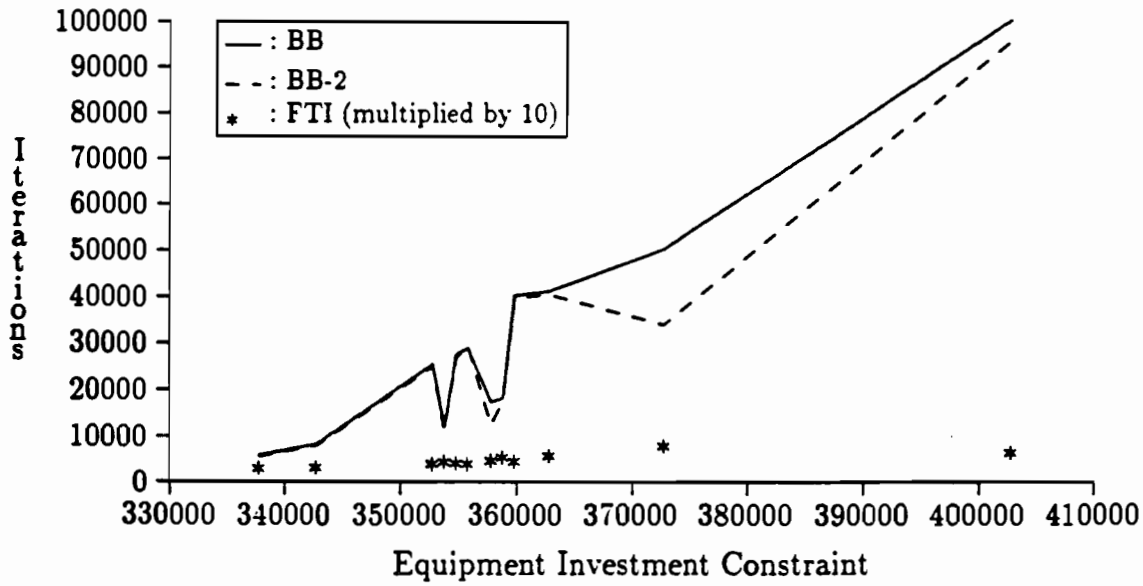


Figure 2: Total number of iterations for three solving procedures.

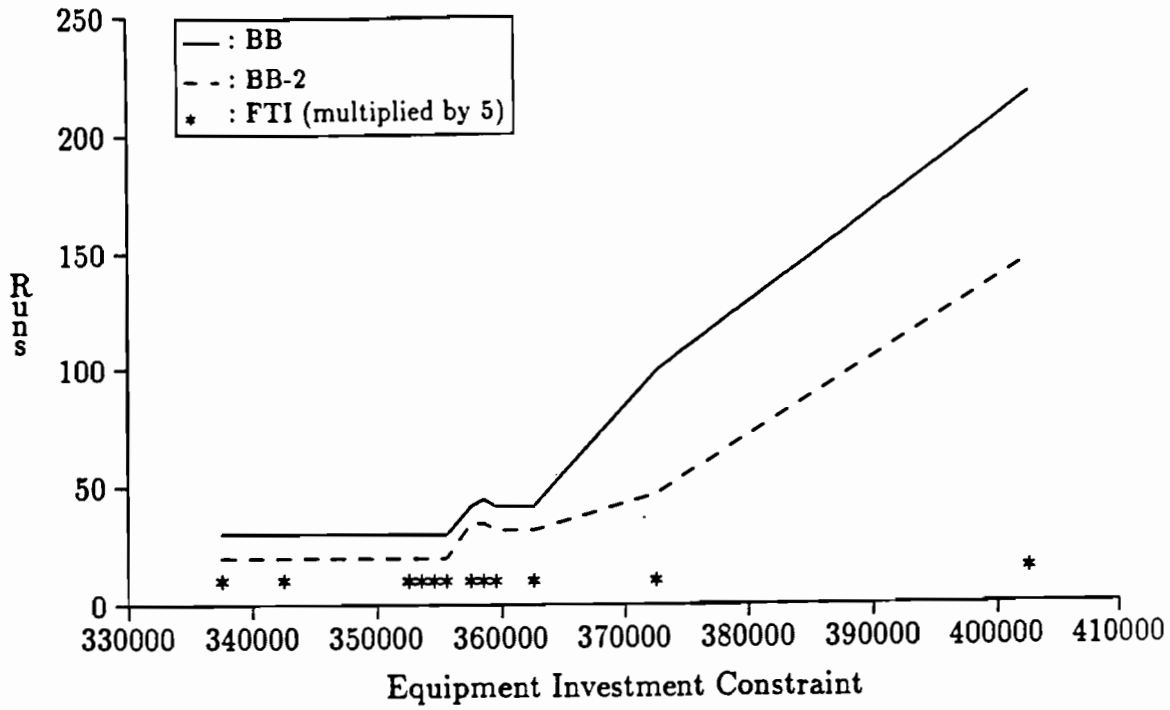


Figure 3: Number of optimization runs for three solving procedures.

The numerical expense of BB and BB-2 procedures, in comparison with FTI, is clearly visible in the figures 2 and 3. The figure 2 shows the total number of MINOS iterations and figure 3 the number of optimization runs for three solving procedures presented in the section 3. It is worth noting that the results of FTI were multiplied by a factor to make it visible on the pictures. It is interesting to note that excluding only one case, FTI procedure was terminated after only two optimization runs. The BB-2 procedure seems to be apparently more efficient than BB.

The detailed results for the value of $RHS(f_1) = 335000$ are given in tables 3 and 4. The production schedule can be determined through the analysis of the Ti values. Since products 1, 5, 6 belong to G_3^3 (see section 2.3) 2, 7 to G_4^3 and 3, 4 to G_5^4, periods 1,2 and 6 of the S superstructure have been deleted.

i	T_{L_i}	B_i	Q_I	T_i	b_i
1	7.4560	943.4532	282428.1520	2231.9966	299.3558
2	7.1430	640.1423	135000.0000	1506.3917	210.8906
3	7.3180	513.6745	180000.0000	2564.3477	350.4165
4	9.9870	665.9694	171000.0000	2564.3477	256.7686
5	6.5340	386.6525	126000.0000	2129.2606	325.8740
6	7.7250	561.6175	154800.0000	2129.2606	275.6324
7	7.3260	497.9144	95400.0000	1403.6557	191.5992

Table 4: Solution for $RHS(f_1) = 335000$ (products)

4 Conclusions

A bi-criteria approach to the optimal design of a multi-purpose batch plant has been presented in this paper. An adequate MINLP model was developed. The three solving procedures were proposed. The numerical results show that in most cases FTI (force to integer) procedure is computationally efficient and leads to achieving a good integer solution. The BB (branch and bound) procedure is easily adaptable for such a model and solves the problem in reasonable time but is still numerically expensive. The BB-2 procedure upgrades significantly this efficiency.

It can be therefore concluded that the model and solving procedures proposed in this paper may constitute a core of a computer DSS (*Decision Support System*).

Acknowledgment

This study is sponsored by The Chemical Industry in Poland but co-sponsorship of the theoretical part of this work by IIASA *International Institute of Applied Systems Analysis, Laxemburg, Austria*) is gratefully acknowledged.

References

Bauman, H. C. (1964) Fundamentals of Cost Engineering in the Chemical Industry. Reinhold, New York.

Birewar, D. B. and Grossmann, I. E. (1989) Incorporating Scheduling in the Optimal Design of Multiproduct Batch Plants. *Computer and Chemical Engineering* **13** (1/2), pp. 141–161.

Grossmann, I. E. and Sargent, R. W. H. (1979) Optimum Design of Multipurpose Chemical Plants. *Ind. Eng. Chem. Process Des. Dev.* **18** (2), pp. 343–348.

Hwang, C-L. and Masud, A. S. M. (1979). In Multiple Objective Decision Making. Methods and Applications, *Lect. Notes in Econ. and Math. Syst.* **164**, Springer-Verlag, New York–Heidelberg–Berlin.

Karimi, L. A. and Reklaitis, G. V. (1985) Variability Analysis for Intermediate Storage in Noncontinuous Processes: Stochastic Case. In PSE'85: The Use of Computers in Chemical Engineering, *The Institution of Chemical Engineers Symposium Series* **92**, pp. 663–674, Institution of Chemical Engineers; Pergamon Press, Oxford, New York, Toronto, Sydney, Paris, Frankfurt.

Klossner, J. and Rippin, D. W. T. (1984) Multipurpose Design Problem. Paper presented on the AIChE Annual Meeting, San Francisco.

Kopytowski, J. A. (1986) Multipurpose Batch Plant Establishment Project. United Nations Industrial Development Organization, Vienna.

Ku, H-ming and Karimi, I. A. (1988) Scheduling in Serial Multiproduct Batch Processes with Finite Interstage Storage: A Mixed Integer Linear Program Formulation. *Ind. Eng. Chem. Res.* **27**, pp. 1840–1848.

Kuriyan, K., Reklaitis, G. V. and Joglekar, G. (1987) Multiproduct Plant Scheduling Studies Using BOSS. *Ind. Eng. Chem. Res.* **26**, pp. 1551–1558.

Lázaro, M. and Puigjaner, L. (1985) Simulation and Optimization of Multi-Product Plants for Batch and Semibatch Operation. In PSE'85: The Use of Computers in Chemical Engineering, *The Institution of Chemical Engineers Symposium Series* **92**, pp. 209–222, Institution of Chemical Engineers; Pergamon Press, Oxford, New York, Toronto, Sydney, Paris, Frankfurt.

Murtagh, B. A. and Saunders, M. (1983) Minos 5.0 User's Guide. SOL, Standford University. Technical Report SOL 83-20, California.

Reingold, E. M., Nievergelt, J. and Deo, N. (1977). In Combinatorial Algorithms. Theory and Practice., Prentice-Hall, Englewood Cliffs, NJ.

Rippin, D. W. T. (1983) Review Paper. Simulation of Single- and Multiproduct Batch Chemical Plants for Optimal Design and Operation. *Computer and Chemical Engineering* **7** (3), pp. 137–156.

Ryś, T. (August 7–11, 1989) Multiobjective Design of Multi-Purpose Batch Plant. Paper presented on International Workshop on MCDS, Helsinki, Finland.

Sparrow, R. E., Rippin, D. W. T. and Forder, G. J. (1974) MULTI-BATCH: A Computer Package for the Design of Multi-Product Batch Plants. *The Chemical Engineer*, pp. 520–525.

Sparrow, R. E., Forder, G. J. and Rippin, D. W. T. (1975) The Choice of Equipment Sizes for Multiproduct Batch Plants. Heuristics vs. Branch and Bound. *Ind. Eng. Chem. Process Des. Dev.* **14** (3), pp. 197–203.

Suhami, I. and Mah, R. S. H. (1981) An Implicit Enumeration Scheme for the Flowshop Problem with no Intermediate Storage. *Computer and Chemical Engineering* **5**, pp. 83–91.

Suhami, I. and Mah, R. S. H. (1982) Optimal Design of Multi-Purpose Batch Plant. Department of Chemical Engineering, Northwestern Uniwersity, Evanston, Illinois.

Takamatsu, T., Hashimoto, I. and Hasebe, S. (1979) Optimal Scheduling and Minimum Storage Tank Capacities in a Process System with Paraller Batch Units. *Computer and Chemical Engineering* **3**, pp. 185–195.

Vaselenak, J. A., Grossmann, I. E. and Westerberg, A. W. (1987) An Embedding Formulation for the Optimal Scheduling and Design of Multipurpose Batch Plants. *Ind. Eng. Chem. Res.* **26**, pp. 139–148.

Wellons, H. S. and Reklaitis, G. V. (1989) The Design of Multiproduct Batch Plants Under Uncertainty With Staged Expansion. *Computer and Chemical Engineering* **13** (1/2), pp. 115–126.

White, C. H. (1989) Productivity Analysis of a Large Multiproduct Batch Processing Facility. *Computer and Chemical Engineering* **13** (1/2), pp. 239–245.