International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

IAC-DIDAS-L Dynamic Interactive Decision Analysis and Support System Linear Version

Rogowski, T., Sobczyk, J. and Wierzbicki, A.P.

IIASA Working Paper

December 1988

WORKING PAPER

IAC-DIDAS-L DYNAMIC INTERACTIVE DECISION ANALYSIS AND SUPPORT SYSTEM LINEAR VERSION

T. Rogowski
J. Sobczyk
A.P. Wierzbicki

December 1988
WP-88-110

IIASA
International Institute
for Applied Systems Analysis

IAC-DIDAS-L DYNAMIC INTERACTIVE DECISION ANALYSIS AND SUPPORT SYSTEM LINEAR VERSION

T. Rogowski
J. Sobczyk
A.P. Wierzbicki

December 1988
WP-88-110

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

Foreword

This paper is one of the series of 11 Working Papers presenting the software for interactive decision support and software tools for developing decision support systems. These products constitute the outcome of the contracted study agreement between the System and Decision Sciences Program at IIASA and several Polish scientific institutions. The theoretical part of these results is presented in the IIASA Working Paper WP-88-071 entitled *Theory, Software and Testing Examples in Decision Support Systems.* This volume contains the theoretical and methodological bacgrounds of the software systems developed within the project.

This paper presents user documentation for two versions of decision analysis and support systems of DIDAS family: IAC–DIDAS–L1 (pilot version) and IAC–DIDAS–L2. These programs can be used for supporting decision problems when the model of the decision situation can be described using the linear programming framework.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

IAC–DIDAS–L
Dynamic Interactive Decision Analysis and Support System
Linear version

T. Rogowski, J. Sobczyk, A. P. Wierzbicki

Institute of Automatic Control, Warsaw University of Technology

Contents

1 Introduction

Both packages (IAC–DIDAS–L1 and L2) are designed to help in analysis based on multiobjective linear programing models. Both belong to the class of decision support system prototypes, that is when supplemented by a model of substantive aspects of a decision situation from the above mentioned class, they can be used either as tools for detailed model analysis or as decision support systems by an user that is an experienced decision maker in the given substantive field but not necessarily computer specialist. Both are implemented on professional microcomputers compatible with IBM–PC–XT (with a hard disk, Hercules or color graphics card and, preferably, a co–processor). However, the first version, IAC–DIDAS–L1, is written in FORTRAN, has support multiobjective linear programming solver that is relatively fast in execution of optimization runs during interactive analysis, can support multiobjective analysis of dynamic problems, but requires the edition of the model of substantive aspects of decision situation in the MPS–format that is well–known to linear programing specialists but not to an average user. This version is developed to the level of a pilot software system and is documented only as an Appendix D to this paper. The paper concentrates on the version IAC–DIDAS–L2 that is developed to the level of scientific transferable software, that is documented and tested to be used widely in research. IAC–DIDAS–L2 is written in PASCAL and supports also an interactive definition and edition of the substantive model by the user, in a user–friendly format of a spreadsheet; however it is limited to essentially static linear programming models. It also has an user friendly interface, supports graphical representation of results in interactive analysis and has a data base for models, formulations of multiobjective analysis problems and results of analysis.

2 Extended introduction

In many situations of complex decisions involving economic, environmental and technological decisions as well as in the cases of complex engineering design, the decision maker needs help of an analyst, or a team of analysts, to learn about possible decision options and their predicted results. The team of analysts frequently summarizes its knowledge in the form of a *model of subtantive aspects* of the decision situation that can be formalized mathematically and computerized.

While such a model can never be perfect and cannot encompass all aspects of the problem, it is often a great help to the decision maker in the process of learning about novel aspects of the decision situation and of gaining expertise when 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, 1985) — the interaction of a decision maker with the team of analysts and the substantive models prepared by them can be of great value.

In organizing such interaction, many techniques of optimization, multicriteria decision analysis and other tools of mathematical programming can be used. To be of value for a holistically thinking decision maker, however, all such techniques must be used as supporting tools of interactive analysis rather than as means for proposing unique optimal decisions and thus replacing the decision maker. The decision analysis and support systems of DIDAS family — that is, Dynamic Interactive Decision Analysis and Support systems, see e.g. (Lewandowski et al., 1984) — are especially designed to support interactive work with a substantive model while using multicriteria optimization tools, but they stress the learning aspects of such work, such as the right of a decision maker to change his priorities and preferences when learning

new facts. DIDAS systems can be used either by analysts who want to 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 require special technical means of support. The IAC–DIDAS–L1 and –L2 versions are designed to support models of linear programming type; specifically, multiobjective linear programming models, often with dynamic structure. If a model has a multiobjective dynamic structure, the *objectives* (called also criteria, outcomes, results, etc.) of decisions form *trajectories*, which might be interpreted as graphs of the dependence of an objective on time or another variable of similar type; these trajectories are evaluated by the user as a whole, complex objective. The decisions can also have the form of trajectories.

Models of multiobjective linear programming type specify, first, the bounds on admissible *decision variables*, in the form of linear equations or inequalities called *constraints* (including, for models of dynamic type, also special constraints called state equations of the model) and, secondly, the attainable decision outcomes, in the form of linear equations for *outcome variables* among which the user can select his objectives. Actually, the distinction between constraints and outcome variables is not necessarily sharp (if the value of a constraint can be changed, it becomes an outcome variable) and the user might select his objectives also among constraint variables.

There are many examples of decision problems that can be analyzed by means of a substantive model of multiobjective linear programming type; for example, DIDAS–type systems with multiobjective, dynamic linear programming models have been used in planning energy policies (see Strubegger, 1985, Messner, 1985), agricultural policies (see Makowski and Sosnowski, 1983) as well as in analyzing various environmental or technological problems (see Kaden, 1985, Gorecki et al., 1983). As a demonstrative or tutorial example, IAC–DIDAS–L1 and –L2 use a multiobjective linear programming model for a problem of diet composition, where the decision variables correspond to various dishes and the constraints or outcomes correspond to the amount of vitamins, minerals, the cost and subjectively defined taste and stimulus of the diet. IAC–DIDAS–L1 uses as well as a demonstrative example a dynamic multiobjective linear programming model for flood control with several tributaries of a river and several reservoirs, where the decisions are time sequences — trajectories — of outflows of reservoirs and the outcomes are trajectories of flows in various points on the river. As more advanced example.

IAC–DIDAS–L2 uses also another demonstrative example from agricultural economics. The user can also define substantive models of multiobjective (possibly dynamic) linear programming type for his own problems and analyze them with the help of IAC–DIDAS–L1 or –L2.

A typical procedure of working with a DIDAS–type system consists of several phases.

In the first phase, a user — typically, an analyst — defines the substantive model and edits it on the computer. In earlier versions of DIDAS–type systems (which were mostly implemented on bigger mainframe computers) this phase has not been explicitly supported in the system and the user had to separately prepare (define and edit) his model in the MPS format. This is a typical format for single–objective linear programming problems and can be also used for multiobjective problems; however, working with MPS format requires some knowledge of linear programming and thus limits the use of such DIDAS systems to rather experienced analysts. On the other hand, there are many existing linear programming models in the MPS format that could be analyzed multiobjectively with the help of a DIDAS system.

Therefore the version IAC–DIDAS–L1 has been designed to work with substantive models in the MPS format while the user–friendliness of professional microcomputers compatible with IBM–PC–XT is exploited only in the graphical representation of results of multiobjective analysis.

The second version: IAC–DIDAS–L2, exploits the user–friendliness of such microcomputers also by supporting the definition and edition of a substantive model in an easy format of a spreadsheet, where the decision variables (and, possibly, some model parameters) are represented by the columns, the constraints and outcome variables — by the rows of the spreadsheet, and the coefficients of all linear functions defining the model are entered in the corresponding cells of the spreadsheet. Therefore, the user can define, review and edit his model easily; when analyzing his model in further phases of work with IAC–DIDAS–L2, he can also return to the model definition phase and modify his model if necessary. The user of IAC–DIDAS–L2 can also have several substantive models recorded in a special model directory, use old models from this directory to speed up the definition of a new model, etc., while the system supports automatically the recording of all new or modified models in the directory. The easiness of model definition and edition has, however, its price: models defined in the spreadsheet format should not be too large and the number of their variables (decision variables, constraints and outcome variables, while counting separately variables for each time instant in dynamic models) should not be too large (not greater than a hundred).

In the second phase of work with DIDAS–type systems, the user — here typically an analyst working together with the decision maker — specifies a multiobjective analysis problem related to his substantive model and participates in an initial analysis of this problem. There might be many multiobjective analysis problems related to the same substantive model: the specification of a multiobjective problem consists in designating outcome and constraint variables in the model that become objectives (or objective trajectories in a dynamic case) and defining whether an objective (or objective trajectory) should be minimized or maximized, or kept close to a given level. For a given definition of the multiobjective analysis problem, the decision and outcomes in the model are subdivided into two categories: those that are *efficient* with respect to the multiobjective problem (that is, such that no objective can be improved without deteriorating some other objective) and those that are inefficient. It is assumed that the user is interested only in efficient decisions and outcomes (this assumption is reasonable provided that the user has listed all objectives of his concern; if he has not, or if some objectives of his concern are not represented in the model he can still modify the sense of efficiency by adding new objectives, or by requiring some objectives to be kept close to given levels, or by returning to the model definition phase and modifying the model).

One of the main functions of a DIDAS–type system is to compute efficient decisions and outcomes — following interactively various instructions of the user — and to present them for analysis. This is done by solving a special parametric linear programming problem resulting from the specification of the multiobjective analysis problem; for this purpose, IAC–DIDAS–L contains a specialized linear programming algorithm called *solver*.

Usually, however, the definition of a multiobjective problem admits many efficient decisions and outcomes; therefore the user should first learn about ***bounds on efficient outcomes***. This is the main function of IAC–DIDAS–L in the initial analysis phase. The user can request the system to optimize any objective separately; however, there are also two special commands in the system, related to this function. The first, called "utopia", results in subsequent computations of the best possible outcomes for all objectives treated separately (such outcomes are practically never attainable jointly, hence the name "utopia" for the point in outcome space composed of such outcomes; in dynamic cases, only approximate joint bounds

for entire trajectories are computed). The second, called "nadir", results in an estimation of the worst possible among the efficient outcomes (defining precisely the worst possible efficient outcome is a very difficult computational task; in some simple cases, the "utopia" computations give enough information to determine the worst possible among the efficient outcomes, but for more general cases this information is not reliable and a more reliable way of estimating the worst possible efficient outcome is implemented in IAC–DIDAS–L).

The "utopia" and "nadir" computations give important information to the user about reasonable ranges of decision outcomes; in order to give him also information about a reasonable compromise efficient solution, a ***neutral efficient solution*** can be also computed in the initial analysis phase following a special command. The neutral solution is an efficient solution situated "in the middle" of the range of the efficient outcomes, while the precise meaning of being "in the middle" is defined by the distances between the utopia and the nadir point. After analyzing the utopia point, the nadir point and a neutral solution (which all can be represented graphically for the user), the initial analysis is completed and the user has already learned much about the ranges of the attainable efficient objectives and the possible trade–offs between these objectives. Each change of the definition of the substantive model or of the multiobjective analysis problem, however, necessitates actually a repetition of the initial analysis phase; on the other hand, the user can omit this repetition if he judges that the changes in the model or in multiobjective analysis definition have been small.

The third phase of work with DIDAS–type systems consists in interactive scanning of efficient outcomes and decisions, guided by the user through specifying ***aspiration levels*** for each objective (or aspiration trajectories, in a dynamic case; called also ***reference points*** or trajectories). The user has already reasonable knowledge about the range of possible outcomes and thus he can specify the aspiration levels that he would like to attain. IAC–DIDAS–L utilizes the aspiration levels as a parameter in a special achievement function, coded in the system, uses its solver to compute the solution of a linear programming problem, equivalent to maximizing this achievement function, and responds to the user with an attainable efficient solution and outcomes (or outcome trajectories) that strictly correspond to the user–specified aspirations.

If the aspirations are "too high" (better than attainable), then the response of the system is a solution with attainable, efficient outcomes that are uniformly as close to the aspirations as possible. If the aspirations are, by a chance ***efficient and precisely attainable***, the response of the system is a solution (decisions and outcome variables) with objective variables, that precisely matches the specified aspirations. If the aspirations are "too low" (if they correspond to attainable but inefficient outcomes that can be improved), then the response of the system is a solution with outcomes that are uniformly better than the aspirations. The precise meaning of the uniform approximation or improvement depends on ***scaling units*** for each objective that can be either specified by the user or defined automatically in the system as the differences between the utopia point and the current aspiration point. This second, automatic definition of scaling units has many advantages to the user who is not only relieved of specifying scaling units but also has a better control of the selection of efficient outcomes by changing aspiration levels in such a case.

After scanning several representative efficient solutions and outcomes controlled by changing aspirations, the user usually learns enough to select either an actual decision, subjectively, (which needs not to correspond to the decisions proposed in the system, since even the best substantive model might differ from real decision situation) or an efficient decision and outcome proposed in the system as a basis for actual decisions.

Rarely, the user might be still uncertain about what decision to choose; for such a case,

several additional options can be included in a system of DIDAS type. Such options include two more sophisticated scanning options: ***multidimensional scanning***, resulting from perturbing current aspiration levels along each coordinate of objective space, ***directional scanning***, resulting from perturbing current aspiration levels along a direction specified by the user (see Korhonen, 1985). Another option is ***forced convergence***, that is, such changes of aspiration levels along subsequent directions specified by the user that the corresponding efficient decisions and outcomes converge to a final point that might represent the best solution for the preferences of the user. However, not all these additional options are implemented in IAC–DIDAS–L, since the experience of working with DIDAS–type systems shows that these options are rarely used.

IAC–DIDAS–L systems have special data bases for models, multiobjective analysis problems and analysis results. The data structure reflects the stage of analysis.

3 Theoretical manual

The standard form of a multiobjective linear programming problem is defined as follows:

$$\textit{maximize} \qquad (q = Cx)\,; \qquad X = \{\, x \in R^n \,:\, Ax = b\,, \qquad x \geq 0\,\} \tag{1}$$

where $x \in R^n$, $b \in R^p$, A is a $m \times n$ matrix, C is a $p \times n$ matrix and the maximization of the vector q of p objectives is understood in the Pareto sense: $\hat{x}$, $\hat{q}$ are solutions of (1) if $\hat{q} = C\hat{x}$, $\hat{x} \in X$ and there are no such x, q, with $q = Cx$, $x \in X$ that $q \geq \hat{q}$, $q \neq \hat{q}$. Such solutions, $\hat{x}$ and $\hat{q}$, of (1) are called an ***efficient decision*** $\hat{x}$ and the corresponding ***efficient outcome*** $\hat{q}$, respectively. If, in the above definition, it were only required that there would be no x and q, with $q = Cx$, $x \in X$, such that $q > \hat{q}$, then the solutions x, q would be called ***weakly efficient***. Equivalently, if the set of all attainable outcomes is denoted by

$$Q = \{\, q \in R^p \,:\, q = Cx\,, \qquad x \in X\,\} \tag{2}$$

and so called ***positive cones*** $D = R^p_+$, $\tilde{D} = R^p_+ \setminus \{O\}$ and $\tilde{\tilde{D}} = int\ R^p_+$ are introduced (thus, $q \geq \hat{q}$ can be written as $q - \hat{q} \in D$, $q \geq \hat{q}$, $q \neq \hat{q}$ as $q - \hat{q} \in \tilde{D}$, and $q > \hat{q}$ as $q - \hat{q} \in \tilde{\tilde{D}}$), then the sets of efficient outcomes $\hat{Q}$ and of weakly efficient outcomes $\hat{Q}^w$ can be written as:

$$\hat{Q} = \{\, q \in Q \,:\, (\hat{q} + \tilde{D}) \cap Q = \emptyset\,\} \tag{3}$$

$$\hat{Q}^w = \{\, \hat{q} \in Q \,:\, (\hat{q} + \tilde{\tilde{D}}) \cap Q = \emptyset\,\} \tag{4}$$

The set of weakly efficient outcomes is larger and contains the set of efficient outcomes; in many practical applications, however, the set of weakly efficient outcomes is decisively too large. For multiobjective linear programming problems, the efficient outcomes are always ***properly efficient***, that is, they have bounded ***tradeoff coefficients*** that indicate how much an objective outcome should be deteriorated in order to improve another objective outcome by a unit.

The ***abstract problem of multiobjective linear programming*** consists in determining the entire sets $\hat{Q}$ or $\hat{Q}^w$, or at least all vertices or basic solutions of the linear programming problem that corresponds to efficient decisions and outcomes.

The ***practical problem of multiobjective decision support***, using linear programming models, is different and consists in computing and displaying for the decision maker (or, generally, for the user of the decision support system) some selected efficient decisions and outcomes.

This selection of efficient decisions and outcomes should be easily controlled by the user and should result in any efficient outcome in the set Q he might wish to attain, in particular, also in efficient outcomes that are not necessarily basic solutions of the original linear programming problem; moreover, weakly efficient outcomes are not of practical interest for the user.

Before turning to some theoretical problems resulting from these practical requirements, observe first that the standard formulation of multiobjective linear programming is not the most convenient for the user. Although many other formulations can be rewritten to the standard form by introducing proxy variables, such reformulations should not bother the user and should be automatically performed in the decision support system. Therefore, we present here another basic formulation of the multiobjective linear programming problem, more convenient for typical applications.

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

$$y = Ax\,; \qquad x^{\mathrm{lo}} \le x \le x^{\mathrm{up}}\,; \qquad y^{\mathrm{lo}} \le y \le y^{\mathrm{up}} \tag{5}$$

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

$$y^c = A^c x = b^c\,; \qquad y^{c,\mathrm{lo}} \le b^c \le y^{c,\mathrm{up}} \tag{6}$$

where A^c is the corresponding submatrix of A. The outcome variables corresponding to equality constraints will be called ***guided outcomes*** here. Some other outcome variables can be also chosen as optimized objectives or ***objective outcomes***. Denote the vector of$p$ objective outcomes by $q \in R^p \subset R^m$ (some of the objective variables might be originally not represented as outcomes of the model, but we can always add them by modifying this model) to write the corresponding objective equations in the form:

$$q = Cx \tag{7}$$

where C is another submatrix of A. Thus, the set of attainable objective outcomes is again $Q = CX$, but the set of admissible decisions X is defined by:

$$X = \{\, x \in R^n \,:\, x^{\mathrm{lo}} \le x \le x^{\mathrm{up}}\,;\;\; y^{\mathrm{lo}} \le Ax \le y^{\mathrm{up}}\,;\;\; A^c x = b^c \,\} \tag{8}$$

Moreover, the objective outcomes are not necessarily minimized; some of them might be minimized, some maximized, some stabilized or kept close to given ***aspiration levels*** (that is, minimized if their value is above aspiration level and maximized if their value is below aspiration level). All these possibilities can be summarized by introducing a different definition of the positive cone D:

$$\begin{aligned} D = \{\, q \in R^p \,:\;\; & q_i \ge 0\,, && i = 1, \ldots, p'\,; \\ & q_i \le 0\,, && i = p'+1, \ldots, p''\,; \\ & q_i = 0\,, && i = p''+1, \ldots, p \,\} \end{aligned} \tag{9}$$

where the first p' objectives are to be maximized, the next, from $p'+1$ to p'', are to be minimized, and the last, from $p''+1$ to p, are to be stabilized. Actually, the user needs only to define what to do with subsequent objectives; the concept of the positive cone D is used here only in order to define comprehensively what are efficient outcomes for the multiobjective problem. Given some aspiration levels for stabilized objectives and the requirement that these objectives should be minimized above and maximized below aspiration levels, the set of efficient outcomes can be defined only relative to the aspiration levels.

However, since the user can define aspiration levels arbitrarily, of interest here is the union of such relative sets of efficient outcomes. Let $\tilde{D} = D \setminus \{\emptyset\}$; then the outcomes that might be efficient for arbitrary aspiration levels for stabilized objectives can be defined, as before, by the relation (3). The weakly efficient outcomes are of no practical interest in this case, since the cone D, typically, has empty interior which implies that weakly efficient outcomes coincide with all attainable outcomes.

The stabilized outcomes in the above definition of efficiency are, in a sense, similar to the guided outcomes; however, there is an important distinction between these two concepts. Equality constraints must be satisfied; if not, then there are no admissible solutions for the model. Stabilized objective outcomes should be kept close to aspiration levels, but they can differ from those levels if, through this difference, other objectives can be improved. The user of a decision support system should keep this distinction in mind and can modify the definition of the multiobjective analysis problem by taking, for example, some outcomes out of the guided outcome category and putting them into the stabilized objective category.

By adding a number of proxy variables and changing the interpretation of matrix A, the substantive model formulation (5), (6), (7), (8) together with its positive cone (9) and the related concept of efficiency could be equivalently rewritten to the standard form of multiobjective linear programming (1); this, however, does not concern the user. More important is the way of user–controlled selection of an efficient decision and outcome from the set (3). For stabilized objective outcomes, the user can change the related aspiration levels in order to influence this selection; *it is assumed here that he will use, for all objective outcomes, the corresponding aspiration levels in order to influence the selection of efficient decisions*. The aspiration levels are denoted here $\bar{q}_i$ or, as a vector, $\bar{q}$ and called also, equivalently, *reference points*.

A special way of parametric scalarization of the multiobjective analysis problem is utilized for the purpose of influencing the selection of efficient outcomes by changing reference points. This parametric scalarization is obtained through maximizing the following *order–approximating achievement function* (see Lewandowski et al. 1983; Wierzbicki, 1986):

$$s(q, \bar{q}) = \min\Big[\min_{1\le i\le p} z_i(q_i, \bar{q}_i),\ \ (\frac{1}{\rho p})\sum_{i=1}^{p} z_i(q_i, \bar{q}_i)\Big] + (\frac{\varepsilon}{p})\sum_{i=1}^{p} z_i(q_i, \bar{q}_i) \qquad (10)$$

where the parameter ε should be positive, even if very small; if this parameter would be equal to zero, then the above function would not be order–approximating any more, but *order–representing*, and its maximal points could correspond to weakly efficient outcomes. The parameter ρ should be $\rho \ge 1$; the interpretation of both these parameters is given later.

The functions $z_i(q_i, \bar{q}_i)$ are defined as follows:

$$z_i(q_i, \bar{q}_i) = \begin{cases} (q_i - \bar{q}_i)/s_i, & \text{if} \quad 1 \le i \le p', \\ (\bar{q}_i - q_i)/s_i, & \text{if} \quad p'+1 \le i \le p'', \\ \min(z_i', z_i''), & \text{if} \quad p''+1 \le i \le p, \end{cases} \qquad (11)$$

where

$$z_i' = (q_i - \bar{q}_i)/s_i', \qquad z_i'' = (\bar{q}_i - q_i)/s_i'' \tag{12}$$

The coefficients s_i, s_i' and s_i'' are scaling units for all objectives, either defined by the user (in which case $s_i' = s_i''$, the user does not need to define two scaling coefficients for a stabilized objective outcome) or determined automatically in the system (see further comments).

The achievement function $s(q, \bar{q})$ is maximized with $q = Cx$ over $x \in X$; its maximization in the system is converted automatically to an equivalent linear programming problem, different than the original one, and having more basic solutions that depend on the parameter $\bar{q}$. If the coefficient $\varepsilon > 0$, then the achievement function has the following properties (see Wierzbicki, 1986):

a) For an arbitrary aspiration level or reference point $\bar{q}$, not necessarily restricted to be attainable or not attainable, each maximal point $\hat{q}$ of the achievement function $s(q, \bar{q})$ with $q = Cx$ over $x \in X$ is a D_ε–efficient solution, that is, a properly efficient solution with tradeoff coefficients bounded approximately by ε and $1/\varepsilon$.

b) For any properly efficient outcome $\hat{q}$ with trade–off coefficients bounded by ε and $1/\varepsilon$, there exist such reference points $\bar{q}$ that the maximum of the achievement function $s(q, \bar{q})$ is attained at the properly efficient outcome $\hat{q}$. In particular, if the user (either by chance or as a result of a learning process) specifies a reference point $\bar{q}$ that in itself is such properly efficient outcome, $\bar{q} = \hat{q}$, then the maximum of the achievement function $s(q, \bar{q})$, equal zero, is attained precisely at this point.

c) If the reference point $\bar{q}$ is 'too high' (for maximized outcomes; 'too low ' for minimized outcomes), then the maximum of the achievement function, smaller than zero, is attained at an efficient outcome that approximates the reference point uniformly best, in the sense of scaling units s_i. If the reference point $\bar{q}$ is 'too low ' (for maximized outcomes; 'too high' for minimized outcomes and it can happen only if there are no stabilized outcomes), then the maximum of the achievement function, larger than zero, is attained at an efficient outcome that is uniformly 'higher' than the reference point, in the sense of scaling units s_i.

d) By changing his reference point $\bar{q}$, the user can continuously influence the selection of the corresponding efficient outcomes $\hat{q}$ that maximize the achievement function.

The parameter ε in the achievement function sets 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 it does not differ, for the decision maker, from weakly efficient outcomes — some of its components could be improved without practically deteriorating other components. Another interpretation of this parameter is that it indicates how much an average overachievement (or underachievement) of aspiration levels should correct the minimal overachievement (or maximal underachievement) in the function (10).

The parameter $\rho \geq 1$ can influence the shape of this achievement function only if $\rho > 1$. If $\rho = 1$, then the middle term of this function can be omitted since it is never active in this case. If $\rho > 1$, then this term becomes active only if the achievement function is positive (that is, if the reference point $\bar{q}$ is 'too low ' for maximized outcomes, 'too high' for minimized outcomes and there are nostabilized outcomes). In such a case, the piece–wise linear achievement function (10) has a piece on its positive level–sets that corresponds to the sum of overachievements $(q_i - \bar{q}_i)/s_i$ and not to the minimal overachievement (for maximized outcomes, with corresponding changes for minimized outcomes). This modification becomes

stronger for larger ρ, but always occurs only for positive values of the achievement function; it is useful when the user wants to select efficient outcomes that maximize the sum of positive overachievements.

The maximization of the achievement function is a convenient way of organizing interaction between the model and the user. Before the interactive–analysis phase, however, the user must firstly define the substantive model, then define the multiobjective analysis problem by specifying outcome variables that should be maximized, minimized, stabilized, guided or *floating* (that is, displayed for the users' information only, but not included as optimized or guided objectives; various decision variables of interest to the user can be also included into one of these categories). Before the initial analysis phase, the user should also define some reasonable lower and upper bounds for each optimized (maximized, minimized or stabilized) variable, and some reasonable scaling units s_i for these variables. In further phases of analysis, a special automatic way of setting scaling units s_i can be also applied; this, however, requires an approximation of bounds on efficient solutions. Such an approximation is performed in the initial analysis phase.

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

However, this way of computing the 'upper' bound for efficient outcomes is not always practical, particularly for problems of dynamic structure (see further comments); thus, IAC–DIDAS–L1 and –L2 use a different way of estimating the utopia point (see Rogowski et al., 1987). This way consists in subsequent maximizations of the achievement function $s(q, \bar{q})$ with suitably selected reference points. If an objective should be maximized and its maximal value must be estimated, then the corresponding component of the reference point should be very high, while the components of this point for all other maximized objectives should be very low (for minimized objectives — very high; stabilized objectives must be considered as floating in this case that is, should not enter the achievement function). If an objective should be minimized and its minimal value must be estimated, then the corresponding component of the reference point should be very low, while other components of this point are treated as in the previous case. If an objective should be stabilized and both its maximal and minimal values must be estimated, then the achievement function should be maximized twice, first time as if for a maximized objective and the second time as if for minimized one. Thus the entire number of optimization runs in utopia point computations is $p'' + 2(p - p'')$. It can be shown that, for problems with static structure (no trajectory objectives), this procedure gives a very good approximation of the utopia point $\hat{q}^{\mathrm{uto}}$, whereas the precise meaning of 'very high' reference should be interpreted as the upper bound for the objective plus, say, twice the distance between the lower and the upper bound, while the meaning of 'very low' is the lower bound minus twice the distance between the upper and the lower bound.

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

maximized and minimized objectives, this procedure can give misleading results, while an accurate computation of the nadir point becomes a very cumbersome computational task.

Therefore, IAC–DIDAS–L1 and –L2 offer an option of improving the estimation of the nadir point in such cases. This option consists in additional p'' maximization runs for achievement function $s(q, \bar{q})$ with reference points $\bar{q}$ that are very low, if the objective in question should be maximized, very high for other maximized objectives, and very low for other minimized objectives, while stabilized objectives should be considered as floating. If the objective in question should be minimized, then the corresponding reference component should be very high, while other reference components should be treated as in the previous case. By recording the lowest efficient outcomes that occur in subsequent optimizations for maximized objectives (and are lower than the previous estimation of nadir component) and the highest efficient outcomes for minimized objectives (higher that the previous estimation of nadir component), a better estimation $\hat{q}^{\text{nad}}$ of the nadir point is obtained.

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

$$s_i = | \hat{q}_i^{\text{uto}} - \hat{q}_i^{\text{nad}} |, \qquad 1 \leq i \leq p \tag{13a}$$

for maximized or minimized outcomes, and:

$$\begin{aligned} s_i' &= \bar{q}_i - \hat{q}_i^{\text{nad}} - 0.01(\hat{q}_i^{\text{uto}} - \hat{q}_i^{\text{nad}}), \\ s_i'' &= \hat{q}_i^{\text{uto}} + 0.01(\hat{q}_i^{\text{uto}} - \hat{q}_i^{\text{nad}}) - \bar{q}_i, \quad p'' + 1 \leq i \leq p \end{aligned} \tag{13b}$$

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

In further interactive analysis, an important consideration is that the user should be able to influence easily the selection of the efficient outcomes q by changing the reference point $\bar{q}$ in the maximized achievement function $s(q, \bar{q})$. It can be shown (see Wierzbicki, 1986) that best suited for this purpose is the choice of scaling units determined by a difference between the slightly displaced utopia point and the current reference point:

$$s_i = \begin{cases} \hat{q}_i^{\text{uto}} + 0.01(\hat{q}_i^{\text{uto}} - \hat{q}_i^{\text{nad}}) - \bar{q}_i, & \text{if} \quad 1 \leq i \leq p', \\ \bar{q}_i - \hat{q}_i^{\text{uto}} - 0.01(\hat{q}_i^{\text{nad}} - \hat{q}_i^{\text{uto}}), & \text{if} \quad p' + 1 \leq i \leq p'', \end{cases} \tag{14a}$$

for maximized or minimized outcomes. For stabilized outcomes, the scaling units are determined somewhat differently than in (13b):

$$\begin{aligned} s_i' &= \hat{q}_i^{\text{uto}} + 0.01(\hat{q}_i^{\text{uto}} - \hat{q}_i^{\text{nad}}) - \bar{q}_i, \\ s_i'' &= \bar{q}_i - \hat{q}_i^{\text{nad}} - 0.01(\hat{q}_i^{\text{uto}} - \hat{q}_i^{\text{nad}}), \quad p'' + 1 \leq i \leq p. \end{aligned} \tag{14b}$$

It is assumed now that the user selects the reference components in the range $q_i^{\text{nad}} \leq q_i \leq q_i^{\text{uto}}$ or $q_i^{\text{uto}} \leq q_i \leq q_i^{\text{nad}}$ (if he does not, the system automatically projects the reference component on these ranges) for all objectives. Observe that, similarly as in the case of the neutral solution, the scaling units are determined automatically once the utopia, nadir and reference points are known; the user is not bothered by their definition. The interpretation of the above way of setting scaling units is that the user attaches implicitly more importance to reaching a reference component if he places it close to the known utopia component; in such a case, the corresponding scaling unit becomes smaller and the corresponding objective component is weighted stronger in the achievement function $s(q, \bar{q})$. Thus, this way of scaling, ***relative to utopia–reference difference***, is taking into account the implicit information, given by the user, involved in the relative position of the reference point.

When the relative scaling is utilized, the user can easily obtain — by moving suitably reference points — efficient outcomes that are either situated close to the neutral solution, in the middle of efficient outcome set Q, or in some remote parts of the set Q, say, close to various extreme solutions.

Typically, several experiments of computing such efficient outcomes give enough information for the user to select an actual decision — either some efficient decision suggested by the system, or even a different one, since even the best substantive model cannot encompass all aspects of a decision situation. However,there might be some cases in which the user would like to receive further support — either in analyzing the sensitivity of a selected efficient outcome, or inconverging to some best preferred solution and outcome.

For analyzing the sensitivity of an efficient solution to changes in the proportions of outcomes, a ***multidimensional scan*** of efficient solutions is implemented in IAC–DIDAS–L1 and –L2. This operation consists in selecting an efficient outcome, accepting it as a base $\bar{q}^{\text{bas}}$ for reference points, and performing p'' additional optimization runs with the reference points determined by:

$$\begin{aligned} \bar{q}_j &= \bar{q}_j^{\text{bas}} + \gamma(\hat{q}_j^{\text{uto}} - \hat{q}_j^{\text{nad}}), \\ \bar{q}_i &= \bar{q}_i^{\text{bas}}, \qquad\qquad i \neq j, \quad 1 \leq j \leq p'', \end{aligned} \tag{15}$$

where γ is a coefficient determined by the user, $-1 \leq \gamma \leq 1$; if the relative scaling is used and the reference components determined by (15) are outside the range $\hat{q}_j^{\text{nad}}$, $\hat{q}_j^{\text{uto}}$, they are projected automatically on this range. The reference components for stabilized outcomes are not perturbed in this operation (if the user wishes to perturb them, he might include them, say, in the maximized category). The efficient outcomes, resulting from the maximization of the achievement function $s(q, \bar{q})$ with such perturbed reference points, are typically also perturbed, mostly along their subsequent components, although other their components might also change.

For 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 is implemented in IAC–DIDAS–L1 and –L2. This operation consists again in selecting an efficient outcome, accepting it as a base $\bar{q}^{\text{bas}}$ for reference points, selecting another reference point $\bar{q}$, and performing a user–specified number K of additional optimizations with reference points determined by:

$$\bar{q}(k) = \bar{q}^{\text{bas}} + \frac{k}{K}(\bar{q} - \bar{q}^{\text{bas}}), \qquad 1 \leq k \leq K \tag{16}$$

The efficient solutions $\hat{q}(k)$, obtained through maximizing the achievement function $s(q, q(k))$ with such reference points, constitute a cut through the efficient set $\hat{Q}$ when moving

approximately in the direction $q - \bar{q}^{bas}$. If the user selects one of these efficient solutions, accepts it as a new q^{bas} and performs the next directional scans along some new directions of improvement, he can converge eventually to his most preferred solution (see Korhonen, 1985). Even if he does not wish the help in such convergence, the directional scans can give him valuable information.

Another possible way of helping in convergence to the most preferred solution is choosing reference points as in (16) but using a harmonically decreasing sequence of coefficients (such as $1/j$, where j is the iteration number) instead of user–selected coefficients k/K. This results in convergence even if the user makes stochastic errors in determining next directions of improvement of reference points, or even if he is not sure about his preferences, and learns about them during this analysis (see Michalevich, 1986). Such a convergence, however, is rather slow and is thus not implemented in IAC–DIDAS–L1 and –L2.

A separate problem is multiobjective decision analysis and support based on substantive models of dynamic structure. A useful standard of defining a substantive model of multiobjective linear dynamic programming type is as follows.

The model is defined on $T + 1$ discrete time periods t, $0 \leq t \leq T$ (where t is a discrete time variable counted in days, years or any other time units; models of dynamic structure can also have other interpretations of the variable t, such numbers of subsequent operations, etc). The decision variable x, called in this case *control trajectory*, is an entire sequence of decisions:

$$x = \{ x(0), \ldots, x(t), \ldots, x(T-1) \} \in R^{nT}, \qquad x(t) \in R^{n} \tag{17a}$$

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

$$w = \{ w(0), \ldots, w(t), \ldots, w(T-1) \} \in R^{m'(T+1)} \tag{17b}$$

is actually one time period longer than x; the initial state $w(0)$ must be specified as given data, while the decision $x(T)$ in the final period is assumed to influence the state $w(T+1)$ only, thereby of no interest for the interval $\{0, \ldots, T\}$. This is because the fundamental equations of a substantive dynamic model have the form of *state equations*:

$$w(t+1) = A(t)w(t) + B(t)x(t); \qquad t = 0, \ldots, T-1, \qquad w(0) \ -- \text{ given} \tag{18a}$$

The model *otcome equations* have, then, the form:

$$\begin{aligned} y(t) &= C(t)w(t) + D(t)x(t), \quad t = 0, \ldots, T-1; \\ y(T) &= C(T)w(T) \in R^{m''} \end{aligned} \tag{18b}$$

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

$$y = \{ y(0), \ldots, y(t), \ldots, y(T-1), y(T) \} \in R^{m''(T+1)} \tag{17c}$$

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

$$x^{lo} \leq x \leq x^{up}, \qquad w^{lo} \leq w \leq w^{up}, \qquad y^{lo} \leq y \leq y^{up} \tag{18c}$$

The matrices $A(t)$, $B(t)$, $C(t)$ and $D(t)$, of appropriate dimensions, can dependent on — or can be independent of time t; in the latter case, the model is called *time invariant*

(actually, in a fully time–invariant model, the bounds should also be independent of time t, that is, they should be constant for all time periods). This distinction is important, in multiobjective analysis of such models, only in the sense of model edition: time–invariant models can be defined easier by automatic, repetitive edition of model equations and bounds for subsequent time periods.

Some of the outcomes might be chosen to be equality constrained, or ***guided*** along a given trajectory:

$$\begin{aligned} y^c(t) &= e^c(t) \in R^{m'''} \subset R^{m''}, \quad t = 0, \ldots, T; \\ e^c &= \{ e^c(0), \ldots, e^c(T) \} \end{aligned} \tag{19}$$

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

$$q = \{ q(0), \ldots, q(t), \ldots, q(T-1), q(T) \} \in R^{p(T+1)}, \qquad q(t) \in R^p \tag{20}$$

Various positive cones could be defined to specify the sense of efficiency of such objective trajectory; however, it is assumed here that the sense of efficiency cannot change along the trajectory, that is, a component $q_i(t)$ that will be maximized in one period t must be also maximized in other time periods, etc. (however, not necessarily in all time periods: if the user wishes to maximize, minimize or stabilize some outcome only in one or several time periods, he can always change suitably the definition of objective outcomes). Thus, assume that the first components $q_i(t)$, for $1 \le i \le p'$, are to be maximized, next, for $p'+1 \le i \le p''$, are to be minimized, and the last components, for $p''+1 \le i \le p$, are to be stabilized. The achievement function $s(q, \bar{q})$ in such a case takes the form:

$$\begin{aligned} s(q, \bar{q}) &= \min \left\{ \min_{0 \le t \le T} \min_{1 \le i \le p} z_i(t), \frac{1}{\rho(T+1)p} \sum_{t=0}^{T} \sum_{i=1}^{p} z_i(t) \right\} + \\ &\quad + \frac{\varepsilon}{(T+1)p} \sum_{t=0}^{T} \sum_{i=1}^{p} z_i(t) \end{aligned} \tag{21}$$

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

$$z_i(t) = \begin{cases} [\, q_i(t) - \bar{q}_i(t)]/s_i(t), & \text{if} \quad 1 \le i \le p', \\ [\, \bar{q}_i(t) - q_i(t) \,]/s_i(t), & \text{if} \quad p'+1 \le i \le p'', \\ \min [\, z_i'(t), z_i''(t) \,], & \text{if} \quad p''+1 \le i \le p, \end{cases} \tag{22}$$

where

$$z_i'(t) = \frac{q_i(t) - \bar{q}_i(t)}{s_i'(t)}, \qquad z_i''(t) = \frac{\bar{q}_i(t) - q_i(t)}{s_i''(t)} \tag{23}$$

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

The estimation of utopia and nadir points in the space of objective trajectories would create, in the dynamic case, major computational difficulties ($p(T+1)$ subsequent optimization runs) if exact estimates were needed; moreover, even if the utopia point in itself is not

attainable, it can be better interpreted if each of its components — in this case, each objective component trajectory — is attainable for the model. These considerations indicate that the way of estimating utopia point by p (or by $p'' + 2(p - p'')$, when stabilized objectives are included) subsequent maximizations of the achievement function (21) with suitably 'very high' or 'very low' components of *reference trajectories*:

$$\bar{q} = \{ \bar{q}(0), \ldots, \bar{q}(t), \ldots, \bar{q}(T-1), \bar{q}(T) \} \in R^{p(T+1)}, \qquad \bar{q}(t) \in R^p \tag{24}$$

is much more adequate for the dynamic case than an exact computation of the utopia point. Denote the results of such maximizations with subsequent reference trajectories $\bar{q}^{(i)}$ by $\hat{q}^{(i)}$, $i = 1, \ldots, p$ (we do not include here stabilized outcomes for the simplicity of denotations); then the components of an approximate utopia trajectory can be determined as:

$$\hat{q}_i^{\mathrm{uto}}(t) = \hat{q}_i^{(i)}(t), \qquad t = 0, \ldots, T; \qquad i = 1, \ldots, p \tag{25a}$$

whereas the components of an approximate nadir trajectory (in the case of maximized trajectories, with obvious modifications in the minimized case) should be determined as:

$$\hat{q}_i^{\mathrm{nad}}(t) = \min_{1 \le j \le p} \hat{q}_i^{(j)}(t), \qquad t = 0, \ldots, T; \qquad i = 1, \ldots, p \tag{25b}$$

Unfortunately, the components of such nadir approximation cannot be interpreted as attainable trajectories for the model (since the minimization in (25b) can result in different j for various t); however, this is less important than in the utopia trajectory case. A more precise approximation of nadir point can be obtained, similarly as in the static case, by additional p (or only p'', if stabilized objectives are included in the model) maximizations of achievement function (21) with yet other reference trajectories $\bar{q}^{(j)}$, $j = p+1, \ldots, 2p$, and by extending the minimization in (25b) to $1 \le j \le 2p$.

Once the approximations of utopia and nadir trajectories are determined, a neutral solution as well as the automatic relative scaling can be defined similarly as in the static case. Other aspects of interactive multiobjective analysis of dynamic models are similar to the static case; naturally, the graphical representation of results of analysis is in some cases more straightforward (for single optimization runs) or, in other cases, more involved (for repetitive runs, as in utopia, nadir and scanning computations) than in the static case.

4 Introductory user information IAC–DIDAS–L2

The IAC–DIDAS–L2 system is recorded on one diskette that should be installed on an IBM–PC–XT or a compatible computer with a hard disk, Hercules or a color graphic card (CGA or EGA) and, preferably, a coprocessor. The diskette contains the compiled code of IAC–DIDAS–L2. After installing it in the users directory, it can be activated (by the command `didas2 <Cr>`) and used in a program system.

4.1 Functions of the program

The system supports the following general functions:

1. The definition and edition of a substantive model of the decision situation, in a form of a linear programming model, in a user–friendly format of a spreadsheet.

2. The specification of a multiobjective decision analysis problem related to the substantive model. This is performed by specific features of spreadsheet edition.

3. The initial multiobjective analysis of the problem, resulting in estimating bounds on efficient outcomes of decisions and in learning about some extreme and some neutral decisions.

4. The interactive analysis of the problem with the stress on learning by the user of possible efficient decisions and outcomes, organized through system's response to user–specified aspiration levels or reference points for objective outcomes. In IAC–DIDAS–L2, the system responds with efficient solutions and objective outcomes obtained through the maximization of an achievement function that is parameterized by the user–specified reference points. The maximization is performed by a special linear programming algorithm called solver, written in PASCAL. The interactive analysis is supported by specific commands from the menu, including commands that might help in convergence to the most preferred solution; however, the main function of the system is helping the user to learn about novel aspects of the decision situation, not necessarily forcing him to converge to one, most preferred solution.

In the IAC–DIDAS–L2 the decision variables are defined as columns of the spreadsheet, the outcome variables are defined as rows, model coefficients are entered in the corresponding cells, there are special rows and columns for scaling units, lower and upper bounds, for defining objective outcomes and their type, for reference points, utopia and nadir points, for solutions corresponding to the reference points. Pressing the function key <F1> the user can get various help displays that suggest in an easy fashion the commands useful in a current phase of work with the system.

4.2 Data structures

All data used by IAC–DIDAS–L2 system are divided in three groups called: *model, problem* and *result*. This structure reflects interdependencies between different variables as well as the sequence of steps in interactive problem analysis.

The first and biggest item called *model* defined in *model edition phase* consists of all data defining substantive situation: names, units and bounds for all input and output variables together with coefficients of the mathematical model.

The second item called *problem* defined in *interaction phase* contains status of each outcome variable that defines its character, as well utopia and nadir points calculated for this combination of objectives. In multiobjective analysis each output variable can be used as objective function and therefore *minimized, maximized* or *stabilized* (kept as close as possible to the reference point) or as simple constraint (marked as *floating* or with empty status field). Alternative definition (*floating*) in the status field for variables acting as constraints is used to enable displaying them on the bar chart.

Last item called *result* consists of the reference point, scaling variables and solutions in objective space and in decision space.

All this items are managed in the form of 'pick up list'. This list can contain up to ten *results* together with corresponding *problems* and the *model*. (Only one *model* may be present in the operational memory each time). During interactive analysis many results might be generated. Some of them are significant and should be saved for further steps of analysis but most of them are not important and may be forgotten. To help user in dealing with such many pieces of information and avoid disturbing him by frequent questions, following rules are observed:

- A new *problem* is generated every time the user changes the status of any outcome variable.
- A new *result* is created each time the reference point or scaling coefficients are modified.
- All such new items are numbered from the beginning of session and marked as temporary.
- If the list overflows (it can contain up to 10 *results*) the result from the bottom of the list (the oldest one) is removed — entirely, if it has no name, and to the disk if it is named. This means that all named items are simply removed from the list but remain on the disk and can be loaded again into the memory but temporary items are completely discarded. To save temporary item the user has to give it a name which will uniquely identify it on the disk. Under MS–DOS or PC–DOS systems such name can be up to nine characters long and can contain letters, digits, and some punctuation characters like hyphen or underbar.

4.3 General conventions

Modular structure of the program results in small set of the rules which can be applied everywhere during interaction with the program. In particular some keys have always the same meaning:

<F1> — context sensitive help

<Esc> — abandon action

<Enter> — accept, select

Every time user is asked to enter any data (text or number) the same procedure is in action so the same set of editing keys can be used.

Text editing keys:

← , → , ↑ , ↓ (arrows)	— move cursor in desired direction
<Home>	— jump to the beginning of text
<End>	— jump to the end of text
<Ctrl> + <U>	— delete all characters preceding cursor
<Ctrl> + <Y>	— delete whole text
grey ← (backspace)	— delete one character preceding cursor
<Del>	— delete character under cursor
<Ins>	— toggle insert and overwrite mode
<Esc>	— abandon editing (discard changes)
<Enter>	— exit editing (accepting changes)

Any time user is editing spreadsheet or is expected to select one of items displayed on the screen following keys can be used:

←, →, ↑, ↓ (arrows)	— move cursor in desired direction
<Ctrl> + ←, →, ↑, ↓ (arrows with <Ctrl> key)	— jump to next window in desired direction
<PgUp>, <PgDn>, <Home>, <End>	— scroll one page up/down/left/right
<Ins>	— open cell for editing

During interaction cursor size shows what is the elementary item which can be edited at the time (one character or one cell).

Two different sound signals are used in the program:

short beep	— requested action is impossible (e.g. moving out of the spreadsheet)
long beep	— requested action is dangerous program waits for confirmation

4.4 Phases of analysis

Two main phases of work with IAC–DIDAS–L2 can be distinguished: model edition and interactive problem solving.

To start any interaction with the program user has to specify a name of the model to be used in it. If specified model exists on the disk program enters phase adequate to the state of the model: if the model is not locked the model editing phase is entered, otherwise the interaction phase is entered.

If the specified model does not exist on the disk IAC–DIDAS–L2 asks for initial size of the model and creates it after user confirmation. Initially column names are set to X1, X2, ..., Xn and row names to Y1, Y2, ..., Ym. When model is created, the program automatically enters model editing phase where user can modify predefined names, set units, lower and upper bounds for all rows and columns as well as define model coefficients.

In both phases, the user can move cursor using arrow keys, or jump to headers pressing appropriate arrow while holding down <Ctrl> button. To open a cell for editing the user should use <Ins> key or type any alphanumeric character. In the second case, this character will replace first character of current contents of the cell.

When the model is ready, it should be locked using <F4> key to ensure that all further experiments will be done with the same model. This is important for comparing data between problems and results related to the same model.

Upon locking a model, the program immediately enters the interaction phase. In this phase user can define a multiobjective optimization problem by setting status for outcome variables chosen to play the role of objective functions. After depressing <Ins> key while cursor is on the Status column, the user can choose between four possibilities: Minimize, Maximize, Stabilize or Floating. A floating variable is treated exactly as any other outcome or constraint but is also displayed on the bar chart. After a problem is defined, it can be saved by using function key <F2>.

After defining a problem, the user should estimate ranges of effective solutions (so called utopia and nadir points). This can be easily done by pressing function key <F6>. As it can

be observed on the screen during calculation, this requires one optimization run for every minimized or maximized variable and two runs for each stabilized one.

When utopia and nadir points are calculated, the user can start interaction from a neutral solution (which can be calculated by pressing function key <F8>) or select any other efficient solution by specifying a reference point, and pressing function key <F5> (an optimization run).

It can be observed that any time the user modifies the status column, the system responds with the generation of a new problem and result together with marking solutions, utopia and nadir columns as old. Similar action takes place when the reference point is modified with the difference that only solutions are marked as old and only a new result is generated.

In a typical case, the user should use automatic scaling (that is, defined by the distance between the utopia and the reference point). However it is also possible to specify any other scaling coefficients by just modifying the values in the column "Scale". Observe that after the first modification, the label of this column will change from Automatic scale to User scale. To reestablish automatic scaling, the user should move the cursor to the label and press <Ins> key and then select desired scaling method.

If the user wants to use any previously calculated result as a starting point for further interaction hi can load it from the disk using function key <F3>. The same possibility can be used on the beginning when the program asks for the name of the model to load. If the user responds with the name of the result or problem stored on the disk all necessary data will be loaded and appropriate phase will be entered.

4.5 Managing and reviewing results

All results defined during interactive session are organized in the pick list. This means that every new problem or result is pushed on the top of stack and when the list is full (it can contain up to 10 items) last item from the list is removed without any warning. If it was saved, it will remain on the disk while in another case it will be completely lost. To review and easily manage the pick list, the user can press function key <F3> while holding down <Alt> key. After this key sequence, the pick list will be displayed on the screen and user will be able to reorganize it by selecting with key <Enter> items to be moved to the top of the list, save them with function key <F2> or remove them from the list with function key <F4>.

Any time user wishes to see a result stored on the disk, it can be loaded by using function key <F3>. It can be also used to predefine the reference point values.

If there is no free place on the disk, the user can decide to delete some files by using function key <F4>.

When there are two or more results in the pick list, it is possible to compare last two items in the pick list by setting an alternative display mode while pressing function key <F6> together with <Alt> key.

4.6 Graphics

Any time in the interaction analysis phase, the results from the pick list can be displayed in the form of the bar chart by pressing function key <F7>. Utopia and nadir ranges are displayed on the screen as a thin rectangular frame. Reference point is marked with red arrow and objective status is shown by light magenta marks at the top of each frame (down triangle means minimized objective, up triangle — maximized, equal sign — stabilized, no mark — floating or constrained). All values are scaled according to the upper and the lower bound specified in the model.

During graphic presentation it is also possible to modify any reference point value and run single optimization without leaving graphics. Frame for reference point modification can be selected by moving cursor to its header and pressing <Ins> key to open it for modification. When the frame is open then the small triangle marker can be moved up and down to set new value of the reference point. When the desired value is set the frame should be closed by pressing <Enter> key. Using the <Esc> key instead of <Enter> allows to abandon modifications and leave the frame unchanged. Function key <F5> starts single optimization run (exactly as in the text mode).

5 User reference manual IAC–DIDAS-L2

There are two versions of the IAC-DIDAS-L2 system. One version requires a math coprocessor (8087/80287) 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 disk, Hercules or a color graphic card (CGA or EGA). The diskette contains the compiled code of IAC-DIDAS-L2 and two testing examples in the subdirectory MODELS.

5.1 Running the program from the floppy disk

Since it is possible to use the system from the floppy disk it is preferable to install it on the hard disk.

To run a program from a floppy disk just make a copy of a distribution disk containing interesting version of the system using DISKCOPY command. When the backup copy is made start the system with a command: DIDAS. Remember that on the distribution floppy disk there is no room for big models or numerous problems and results. If you need more free space use another disk for data. The disks can be changed before the model is loaded or created.

5.2 Installing the program on the hard disk

To install IAC-DIDAS-L2 system on the hard disk observe following instructions.

1. The didas.exe file should be installed in the user directory or any other directory named in the PATH directive.
2. In the user directory a subdirectory MODELS should be created.
3. Testing examples can be copied to this subdirectory.
4. The system can be activated by the command: DIDAS.

5.3 Activating the program and loading data

There are two methods of running the program. Using the first method type just the DIDAS command to the operating system and wait while the IAC-DIDAS-L2 is loaded. After a moment the invitation screen is displayed and the red bar of main menu appears. There are four functions in the menu:

1. Help — By pressing function key <F1> the help window can be invoked.

2. Load data — After pressing the function key <F3> the system asks to enter the name of a file containing desired data. If the user does not remember the name he can specify a name using wildcard characters: '*', '?' exactly as in commands of the operating system. An empty name can be used to indicate all files (*.*). In such case the current file directory is displayed and the file can be selected by moving a cursor and pressing the <Enter> key. Selecting a subdirectory changes the directory. If there are two many files for easy choice the user can return to the name editing by pressing the <Esc> key. An attempt to load a problem or result causes all related data to be loaded before. Typing only the name of the file results in loading all files with such name existing in the directory.

3. Model creation — When the function key <F4> is pressed you are asked for a name of the new model. After this a small green window appears in the middle of the screen. In this window you can set the desired size of the new model (numbers of rows and columns). If this two numbers are specified you may move the cursor to the Create field and generate the model pressing the <Enter> key. New model will have specified numbers of columns named X1..Xn and rows named Y1..Ym.

4. Exit — To leave the IAC-DIDAS-L2 program and return to the operating system press function key <F9>.

This method is easy but not convenient for advanced users so there is another possibility. Typing the command: DIDAS NAME is equivalent to following steps described above:

1. invoking IAC-DIDAS-L2 with a DIDAS command.

2. pressing <F3> function key.

3. entering the NAME.

If the NAME uniquely identifies the file (does not contain any wildcard characters) then invitation screen appears for a while and specified file is loaded.

5.4 Text editing

Many times during work with the system the user is asked to enter any text. In all such cases the same procedure (EditLine) is invoked so the same set of editing keys can be used:

Text editing keys:

← , → (arrows)	— move cursor in desired direction
<Home>	— jump to the beginning of text
<End>	— jump to the end of text
<Ctrl> + <U>	— delete all characters preceding cursor
<Ctrl> + <Y>	— delete whole text
grey ← (backspace)	— delete one character preceding cursor
<Del>	— delete character under cursor

<Ins> — toggle insert and overwrite mode

<Esc> — abandon editing (discard changes)

<Enter> — exit editing (accepting changes)

Observe that when the text is too long to fit in current window than it can be scrolled left or right by moving the cursor in the opposite direction.

5.5 Menu or spreadsheet manipulations

In numerous moments the user is asked to choose one of several possibilities or he wants to move to a desired cell of the spreadsheet. In such cases he can use following set of the keys:

$\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$ (arrows) — move cursor in desired direction

<Ctrl> + $\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$

(arrows with <Ctrl> key) — jump to next window in desired direction

<PgUp>, <PgDn>, <Home>, <End> — scroll one page up/down/left/right

Warning! Key sequences: <Ctrl> + $\uparrow$, $\downarrow$ may not work on some computers. They can be made working by using some keyboard enhancement programs like SuperKey, Keyworks or others (sometimes it does not work for IBM-AT).

5.6 Model editing

During this phase you can move from cell to cell entering or modifying any data. You can overwrite default row and column names, and specify the units and bounds for them. Any time you may save the current state of the model on the disk pressing function key <F2>. When the model is ready you can lock it and proceed to the interaction phase. You need not to edit the whole model during one editing session. You can exit from the IAC-DIDAS-L2 any time you want (function key <F9>) remembering to save the model (as described above) before.

In this phase it is also possible to change the size of the model using <Ctrl> <Ins> or <Ctrl> <Del> keys. After the <Ctrl> <Ins> key is pressed the small window with four arrows appears on the screen. At the moment you should press the arrow key to point the direction from the current cell where the new row or column should be inserted. The <Ctrl> <Del> key can be used to delete any row or column. To do this move the cursor to the desired row or column and use the key. After this you will be asked whether the row or column should be removed.

5.7 Interaction phase

After the model is locked the interaction phase is entered. In this phase you can examine names and units of rows and columns but you can not modify them. In this phase you can overwrite bounds for any row or column. To start the interaction you should specify bounds for all rows and columns (if not specified yet) and define the multiobjective optimization problem marking several rows as objectives. To mark any row as objective please move the cursor to the status column of the desired row and press the <Ins> key. After this you can choose one of four possibilities:

1. Minimized — this indicates that you want the value of the selected row to be as low as possible (in the best case equal to the lower bound).

2. Maximized — this indicates that you want the value of the selected row to be as large as possible (in the best case equal to the upper bound).

3. Stabilized — this indicates that you want the value of the selected row to be kept as close as possible to the value of the reference point (in the best case it should be equal to it).

4. Floating — this row will not play the role of the objective but will be displayed on the bar chart as well as any objective.

In the interaction phase you can use following function keys:

1. <F5> — Start single optimization.
   This function should be used when the problem is well defined and the Utopia and Nadir points are calculated, but if they are not the system automatically calculates them first.

2. <F6> — Calculate the Utopia and Nadir points.
   This function gives you an information about ranges of variations of the objectives. The Utopia point can be understood as the best solution (but typically not attainable) on all objectives. Nadir point is just the opposite: it is the worst possible solution (typically attainable but not effective). Remember that due to numerical complications connected with exact calculation of the nadir point the values in the Nadir column of the spreadsheet specify only the rough approximation of the point.

3. <F7> — Proceed with the limited interaction in the graphic mode.
   In the graphic mode only modifications of the reference point can be done and single optimization runs can be invoked.

4. <F8> — Calculate the neutral solution.
   This function differs from the <F5> in that the program specifies the reference point and the scale instead of you. It is very useful as the first solution after calculating the utopia and nadir point. This solution gives a reasonable compromise between conflicting objectives.

5. <F2> — Save data.
   This function displays the list of all problems and result remaining in the memory. To save any temporary problem or result move the cursor to corresponding field and press the <Enter> key. After this you will be asked to enter the name for selected item. If you are saving the result and the problem was not saved then you will be asked again for the name of the problem but the name of the result will be displayed as the default. You are free to overwrite or to accept it.

6. <F3> — Load data.
   This function acts exactly as described for loading first data file after start of the system. The only difference is that if you attempt to load any file related to different model than the current one then the system will ask you whether you wish to discard all temporary results remaining in the memory or to abort loading.

7. <Alt F3> — Pick list management.
   This function displays the list of all results and problems remaining in the memory and allows to:

   - reorder them — pressing the <Enter> key moves the pointed result to the top of the list.
   - save — pressing function key <F2> causes the same action as described above at point 5.
   - erase — pressing <F4> removes the result from the list.

8. <Ctrl F4> — Delete files from the disk.
   This function gives you the possibility to free the space on the disk to save temporary results remaining in the memory (if you exit to the operating system all of them will be lost). After using this function the program asks you for the name of the file to be erased. Specifying wildcard name gives you the menu of files matching specification. In this menu you can delete files simply pressing <Enter> key while cursor points to selected file.

9. <Alt F2> — Unlock model.
   This function gives you the possibility to obey data consistency checks and modify the model. Remember that all previously generated results will point to the modified model but may have nothing common with it!

10. <Alt F6> — Toggle display mode (single or double window).
    In the single window mode only one result from the top of the pick list can be examined or modified. In double window mode two top results are displayed but only one can be modified. This two results are displayed one under another and are scrolled together.

5.8 Graphics

In many cases it is not necessary to deal with exact numerical values of reference point and solutions. Usually rough information about what has changed is satisfactory. In such cases graphic presentation of results and graphic manipulation of input values can be found very convenient. In IAC-DIDAS-L2 it is possible to modify reference point values and run single optimizations without leaving graphic mode. As for today it is not possible to modify bounds or set row status fields.

To enter graphic mode in the interaction phase simply press function key <F7>. The picture consists of several frames one for each objective or floating outcome. Every frame is labeled with the name of the objective. Values of upper and lower bounds are preceded with a letter (U for upper bound, L for lower bound). All values in the frame are scaled according to upper and lower bounds. Solutions are displayed in the form of bars. Oldest result (from the bottom of the pick list) appears on left side. The recent result is displayed on the right. Bars in all frames are displayed in the same order. Names or number of results are displayed under first frame. If an outcome plays the role of the objective then a thin white rectangle marks the range between nadir and utopia points. Small violet triangle just under frame header signals whether the objective is maximized or minimized. Stabilized objectives are marked with small violet equal sign in the same position. Outcomes without any mark are not objectives (floating outcomes). Observe that if you have several problems in the pick list then bars are grouped and utopia-nadir frame is common for all results related to the same

problem. Remember that the order of bars in the picture is defined by the pick list, so mixing results in the pick list you can cause more utopia-nadir frames to be displayed than number of problems. The last but very important element of the picture is the red arrow marker indicating the position of the reference point for particular result. In the right lower corner a small example frame can be found.

After drawing the whole picture IAC-DIDAS-L2 displays in the left upper corner of the screen a rectangular blinking cursor to inform you that it is possible to do more than just looking at the picture. In fact it is possible to move the cursor to any frame and select it for modification pressing <Ins> key (this is equivalent to opening the reference point cell in the text mode). After this a small red triangle blinking cursor appears in the position of the previously set reference point value. Moving this marker up or down you will set the reference point. To move to another frame and a modify it it is necessary to close the currently edited one. To do this press the <Enter> key. At the moment the red marker remains in the set position and rectangular cursor starts blinking again. If you use the <Esc> key instead of <Enter> then the reference point will not be changed and triangle marker will be moved back to the previous position.

When the rectangular cursor is blinking it is possible to start the optimization by pressing the function key <F5> (exactly as in the text mode). As in the text mode, if the utopia and nadir points have not been calculated yet the system starts a series of optimization runs to estimate them before running the requested optimization. After successful completion of the optimization run the screen is cleared and the picture is redrawn with one bar more. New bar is added on the right side of each frame. All old bars retain their colors.

To return back to the text mode press the <Esc> key.

5.9 Data manegement

When the user plays with IAC-DIDAS-L2 system many numeric information are entered by him into spreadsheet cells and many other are obtained as results of different computations. To keep track on what was done it is necessary to remember not only current values of cells but also several previous values. To organize and simplify this process all data describing current state of the system were divided into three groups.

The first and biggest group of data is called *model*. It defines all names of rows and columns together with their units and bounds, and with all model coefficients. All this data can be set in the model editing phase and most of them cannot be modified in further phases. The only exception are bounds which can be overwritten in the interaction phase during definition of the multiobjective programming problem. This does not imply that bounds specified in the model definition are modified. In fact model bounds are just taken as starting values for bounds of the problem. Most interesting parts of the model definition can be examined in all phases. This concerns names of rows and columns as well as units and bounds.

The second group of the data called *problem* consists of bounds for rows and columns together with the status of all rows and approximate values of the utopia and nadir points. The status defines whether this outcome plays the role of the objective (is optimized) or is just a constraint. As it was mentioned earlier there are three types of objectives. They can be minimized, maximized or stabilized (kept as close as possible to the value of the reference point). The bounds with row status and with model definition uniquely define the values for utopia and nadir points so this values, if calculated, are kept together in the problem definition.

The third and last group is called *result* and it consists of the value of the reference point, scaling factors together with the corresponding solution (if calculated).

This data can be stored on the disk as DOS files. Every such group is stored in one file. To avoid some name conflicts, files containing model definitions have file extension (type) .MOD, files containing problems have file extension .PRO, and files containing results have extension .RES. Every such file contains information about related files. This means that file containing the result stores also the name of the file containing corresponding problem. Such links enable loading all related files without asking the user for names of them and ensure correct loading order. Every file stores also the creation time of the item (model, problem or result) so it is impossible to replace the problem file and then load old result files with new problem definition. It is assumed that all related files are stored in the same directory, but it is not necessary to keep all of them together but in such case they have to be loaded manually starting from the model file and specifying path for every file.

During interaction with the IAC-DIDAS-L2 system many results can be created. Most of them are not interesting and can be immediately forgotten. To avoid frequent questions concerning names for new results and to save space on the disk, only results pointed by the user are stored in the files; the remaining ones are treated as temporary, numbered from the beginning of the session and stored only in the memory. All numbered results are considered to be temporary and only up to ten recent results with corresponding problems are kept in the memory. All three groups of data are organized in the pick list. This list can be examined any time during interaction phase by pressing <Alt F3> key. In the list every row corresponds to one result. If the new result (or problem) is generated then it appears on the top of the list. If at this moment the list is full (it contains ten results) then one result from the bottom of the list is removed. If the result was temporary then it is completely erased. If it was saved it is only removed from the pick list but remains on the disk and can be loaded again if necessary. To keep any result for the future it is necessary to give it an unique file name to store it on the disk. It can be done in two ways.

1. Pressing <F2> key in the interaction opens the pick list window. Then results and problems to be saved can be selected by moving the cursor and pressing the <Enter> key. After this the user is asked to give the name for the selected item. If the user selects a result related to a problem which has not been saved yet then system asks him again for the name which should be given to the problem suggesting the same name as for result. The user is free to accept or to overwrite it. When the item is saved the system waits for other items to be selected and saved. If all interesting results and problems are saved press the <Esc> key to return to the interaction. Observe that during all this actions the window of the pick list had the label: Save.

2. Pressing <Alt F3> in the interaction phase opens pick list window labeled: Pick. In this window it is also possible to select any item with a cursor but to save it the function key <F2> should be pressed instead of the <Enter> key. All other actions are similar to described above.

The pick list window labeled: Pick (after pressing <Alt F3> key) can be used also to reorder results in the list. To move any result to the top of the list move the cursor to it and press the <Enter> key. Observe that the same action for problem (or model) causes generation of the new empty result (new empty problem and result).

Because the pick list is used to define the order of results on the bar chart it may be sometimes useful to remove some results from the list to make the picture more clear. This

can be obtained in a similar way as was described for saving results, but with function key <F4> used instead of <F2>. Both methods can be used. Observe that in the first case (after pressing <F4> during interaction) the pick list window has the label: Delete. Removing results from the pick list does not affect the disk so if any result was saved on the disk then removing it from the pick list does not delete it from the disk.

If it is necessary to free some space on the disk (e.g. to save very important recent results) you can use the <Ctrl F4> function key. After pressing it the system asks for name of files to be erased. Specifying wildcard names gives you the directory of matching files. In this directory it is possible to point a file with the cursor and delete it pressing the <Enter> key. Before deleting any file the system asks you for acceptance. It was done to avoid accidental damages.

6 Illustrative example

Suppose we would like to compose breakfast following same dietary guidance and have a model that for six dietary items (rolls, cereals, butter, fruits, milk and coffee) determines their cost, calorie content, carbohydrates, fats content as well as main microelements (Calcium, Magnesium, Phosphor, Iron) and main vitamins (vitamin A, B, C, and PP). We add to this model two additional outcomes concerning with taste where the coefficients are entered arbitrarily by the user. First called Taste and second Stimulus because one can like more milk than coffee but need coffee to get sufficient stimulus. The model with all its coefficients is easy to enter or modify in the model editing mode.

Besides coefficients we entered also upper bounds for all decision variables (dietary items, the lower bounds are obviously zero) as well as lower and upper bounds for example it's not good to eat an breakfast containing more than a given number of vitamin A units or we obviously have on the upper bound at the calorie in take.

After the model is established and edited, we should lock it using function <F4> from the corresponding menu. This is needed in order to be sure that we work with defined model in further phases of the decision analysis process. After locking the model the system enters the interaction phase. (Fig. 2)

For given model we should define the corresponding decision analysis problem. This is done by selecting outcome variables as minimized, maximized, stabilized or floating. For this purpose we put the cursor in the status column for the corresponding outcome variable, press the insert key and select the appropriate status.

Suppose we have chosen to minimize cost and calories, maximize taste and stimulus and stabilize the content of the calcium in the breakfast. We would like graphical representation of vitamin C and then we choose vitamin C as floating variable. After defining the status of chosen outcomes we must define the bounds on efficient outcomes for this multiobjective analysis problem (so called utopia and nadir points) for this in the corresponding menu we use <F6> key. We need have four optimization runs for minimized and maximized variables as well as two additional for the stabilized variable. The obtained values in the utopia and nadir column inform us what are the reasonable ranges of the outcome variables.

For example we can't get lower cost than 8.58 units and stabilized calcium values are in range 180 to 720. All the values are extreme and do not correspond to any attainable solution for breakfast composition. To get reasonable breakfast composition corresponding in a sense to middle value between these extremes we compute so called neutral solution (function <F8>). (Fig. 5)

IAC - DIDAS - L2 V5. Model editing			Bounds:		Names	Rolls	Cereals	Butter	Fruitfr
					Units	50 g	50 g	10 g	150 g
					Value				
Names	Units	Value	lower	upper	upper lower	3.	2.	3.	2.
Cost				120.		6.	3.	5.	14.
Taste	artun		4.	40.		3.	1.	2.	4.
Stimulus	artun		4.	80.		3.	3.	4.	3.
Callorie	kcal		150.	1200.		124.	179.	75.	79.
Protein				60.		4.	3.	0.1	0.5
Carbohyd			10.	100.		26.	36.		11.
Fats				80.		1.	2.	8.	0.5
Calcium			180.	720.		8.	10.	2.	9.
Magnesiu				400.		12.	23.	0.2	5.
Phosphor				1000.		42.	103.	1.6	13.
Iron				16.		1.	1.		0.4
Vit.A			100.	1600.				270.	160.
Vit.B				3.		0.12	0.14		6.0e-2
Vit.C				200.					30.
Vit.PP				10.		0.4	1.	1.0e-2	0.23

Figure 1: Screen in the model editing mode.

IAC - DIDAS - L2 V 5. Interaction			Names	Rolls	Cereals	Butter	Fruitfre	Milk
			Upper bound	3.	2.	3.	2.	3.
			Value					
			Lower bound					
	DIET	0	0					

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	User scale
Cost							120.	1.
Taste		4.					40.	1.
Stimulus		4.					80.	1.
Callorie		150.					1200.	1.
Protein							60.	1.
Carbohyd		10.					100.	1.
Fats							80.	1.
Calcium		180.					720.	
Magnesiu							400.	1.
Phosphor							1000.	1.
Iron							16.	1.
Vit.A		100.					1600.	1.

Figure 2: Screen in the interaction phase.

IAC - DIDAS - L2 V 5. Interaction				Names	Rolls	Cereals	Butter	Fruitfre	Milk
				Upper bound	3.	2.	3.	2.	3.
				Value					
				Lower bound					
	DIET	0	0						

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	User scale
Cost	Min						120.	1.
Taste		Minimize Maximize Floating Stabilize					40.	1.
Stimulus							80.	1.
Callorie							1200.	1.
Protein							60.	1.
Carbohyd		10.					100.	1.
Fats							80.	1.
Calcium	Sta	180.					720.	1.
Magnesiu							400.	1.
Phosphor							1000.	1.
Iron							16.	1.
Vit.A	Flo	100.					1600.	1.

Figure 3: Setting the status of an outcome variable.

IAC - DIDAS - L2 V 5. Interaction				Names	Rolls	Cereals	Butter	Fruitfre	Milk
				Upper bound	3.	2.	3.	2.	3.
				Value					
				Lower bound					
	DIET	0	0						

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	User scale
Cost	Min		8.58			120.	120.	1.
Taste	Max	4.	28.91			4.	40.	1.
Stimulus	Max	4.	62.76			4.	80.	1.
Callorie	Min	150.	150.			934.89	1200.	1.
Protein							60.	1.
Carbohyd		10.					100.	1.
Fats							80.	1.
Calcium	Sta	180.	720.			180.	720.	1.
Magnesiu							400.	1.
Phosphor							1000.	1.
Iron							16.	1.
Vit.A	Flo	100.					1600.	1.

Figure 4: After calculation of the utopia and nadir.

IAC - DIDAS - L2 V 5. Interaction				Names	Rolls	Cereals	Butter	Fruitfre	Milk
				Upper bound	3.	2.	3.	2.	3.
				Value	0.58		3.	1.14	0.99
				Lower bound					
	DIET	0	1	Neutral solution					

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	Automatic scale
Cost	Min		8.58	63.72	8.58	120.	120.	111.42
Taste	Max	4.	28.91	16.58	28.91	4.	40.	24.91
Stimulus	Max	4.	62.76	33.68	62.76	4.	80.	58.76
Callorie	Min	150.	150.	538.48	150.	934.89	1200.	784.89
Protein				10.77			60.	1.
Carbohyd		10.		38.16			100.	1.
Fats				32.1			80.	1.
Calcium	Sta	180.	720.	316.36	450.	180.	720.	270.
Magnesiu				49.49			400.	1.
Phosphor				256.91			1000.	1.
Iron				1.41			16.	1.
Vit.A	Flo	100.		1268.19			1600.	1.

Figure 5: Neutral solution.

IAC - DIDAS - L2 V 5. Interaction				Names	Rolls	Cereals	Butter	Fruitfre	Milk
				Upper bound	3.	2.	3.	2.	3.
				Value	1.49		3.	0.35	0.88
				Lower bound					
	DIET	0	2	The first interactive solution					

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	Automatic scale
Cost	Min		8.58	53.41	50.	120.	120.	41.54
Taste	Max	4.	28.91	15.57	16.58	4.	40.	12.35
Stimulus	Max	4.	62.76	31.29	33.68	4.	80.	29.14
Callorie	Min	150.	150.	570.46	538.48	934.89	1200.	389.26
Protein				13.13			60.	1.
Carbohyd		10.		51.84			100.	1.
Fats				31.83			80.	1.
Calcium	Sta	180.	720.	283.16	316.36	180.	720.	404.18
Magnesiu				52.			400.	1.
Phosphor				260.8			1000.	1.
Iron				1.95			16.	1.
Vit.A	Flo	100.		1109.45			1600.	1.00e-20

Figure 6: First interactive solution.

IAC - DIDAS - L2 V5. Interaction			Names	Rolls	Cereals	Butter	Fruitfre	Milk
			Value 1.	0.5		3.	0.98	0.82
			Value 2.	1.49		3.	0.35	0.88
1. DIET	0	3	The second interactive solution					
2. DIET	0	2	The first interactive solution					

1.Names	St.	Lower b	Utopia	Value	Ref. p.	Nadir	Upper b	Scale
Cost	Min		8.58	55.5	50.	120.	120.	41.54
Taste	Max	4.	28.91	14.94	16.58	4.	40.	12.36
Stimulus	Max	4.	62.76	29.82	33.68	4.	80.	29.14
Callorie	Min	150.	150.	489.85	450.	934.89	1200.	300.78
Protein				9.07			60.	1.
Carbohyd		10.		32.61			100.	1.
Fats				30.73			80.	1.
2.								
Cost	Min		8.58	53.41	50.	120.	120.	41.54
Taste	Max	4.	28.91	15.57	16.58	4.	40.	12.36
Stimulus	Max	4.	62.76	31.29	33.68	4.	80.	29.14
Callorie	Min	150.	150.	570.45	538.48	934.89	1200.	389.26
Protein				13.13			60.	1.
Carbohyd		10.		51.84			100.	1.
Fats				31.83			80.	1.

Figure 7: Next solution compared with the first one.

IAC - DIDAS - L2 V 5. Interaction			Names	Rolls	Cereals	Butter	Fruitfre	Milk
			Upper bound	3.	2.	3.	2.	3.
			Value	1.54		1.62	1.03	0.97
			Lower bound					
DIET	4	4	Neutral solution for extended problem					

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	Automatic scale
Cost	Min		8.58	64.61	8.58	120.	120.	111.42
Taste	Max	4.	28.91	16.38	28.91	4.	40.	24.91
Stimulus	Max	4.	62.76	33.21	62.76	4.	80.	58.76
Callorie	Min	150.	150.	544.73	150.	934.89	1200.	784.89
Protein				14.39			60.	1.
Carbohyd		10.		61.82			100.	1.
Fats	Min		4.26	21.81	4.26	39.17	80.	34.91
Calcium	Sta	180.	720.	314.21	450.	180.	720.	270.
Magnesiu				60.6			400.	1.
Phosphor				289.26			1000.	1.
Iron				2.34			16.	1.
Vit.A	Flo	100.		870.45			1600.	1.00e-20

Figure 8: Neutral solution for extended problem.

IAC - DIDAS - L2 V 5. Interaction				Names	Rolls	Cereals	Butter	Fruitfre	Milk
				Upper bound	3.	2.	3.	2.	3.
				Value	1.81		1.12	0.81	0.77
				Lower bound					
	DIET	SAT	SAT	The satisfactory result					

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	Automatic scale
Cost	Min		8.58	56.08	50.	120.	120.	41.54
Taste	Max	4.	28.91	14.54	16.38	4.	40.	12.56
Stimulus	Max	4.	62.76	28.87	33.21	4.	80.	29.61
Callorie	Min	150.	150.	494.04	450.	934.89	1200.	300.78
Protein				13.83			60.	1.
Carbohyd		10.		64.36			100.	1.
Fats	Min		4.26	16.58	15.	39.17	80.	10.78
Calcium	Sta	180.	720.	254.71	314.21	180.	720.	406.33
Magnesiu				55.76			400.	1.
Phosphor				254.6			1000.	1.
Iron				2.46			16.	1.
Vit.A	Flo	100.		645.88			1600.	1.00e-20

Figure 9: The satisfactory result.

The neutral solution gives us rather high value of the cost (63.72 units) and rather high value of the calories 538.48 units. Therefore we might start interaction by defining reference point for further efficient solutions slightly lower than the obtained value of the cost outcome (50.) while rewriting in the reference point other values of the neutral solution of the selected outcome variables. With this reference point we determine the first interactive efficient solution (function <F5>).

Looking at the obtained solution (Fig. 6) we observe that indeed the cost outcome decreased to 53.41 together with slight decrease of taste and stimulus but we are worried by the calories that increased to 570.4 and fats that have rather high value 31.83. We first try decrease the reference point for the calories putting there 450.

The corresponding solution (Fig. 7) is slightly worse than expected and has less calories but still has to much fats. Therefore we decide to change problem by entering fats as minimized variable. For the new problem we must compute new utopia and nadir bounds. They don't differ in this case from the previous one at the four first objectives but they inform us that the lower bound of fat is 4.26.

We compute now the neutral solution for this problem, which is already quite reasonable. Perhaps a bit too costly and too high on calories thus we put again 50. as reference point for cost and 450. as reference point for calories while rewriting other neutral solution output as reference point for other outcome variables.

After optimization we obtain a result that is still slightly high on fat thus we decrease the reference point to 15. and obtain satisfactory result on all outcomes of interest: the cost circa 56., taste 14.54, stimulus 28.87, calories 494.04, fats only 16.58 and calcium 254.71 with 645.88 units of the floating variable vitamin A. The composed breakfast consist of 1.81 rolls with 1.12 butter, 0.81 fresh fruit, 0.77 of milk and 1.31 of coffee (it's still high on stimulus) now if we wish to analyze the history of decision process we can display a bar chart. Observe that for fats the utopia and nadir bounds are displayed only for the last three experiments

because previously fat was not included in the list of optimized outcomes. We see also that the degrees of cost, calories and fats as compared to the last neutral solution resulted in the related decrease of calcium content. If we do not like low calcium content we can return to further interaction.

7 Training example

After starting IAC-DIDAS-L2 from the distribution disk using command line: `DIDAS` our system will be loaded and will show the first screen with full name of the program, names of the authors and institutions, version number and date of release (Fig. 10).

At the bottom of the screen you can see bright red bar with brief explanation of most important keys at the moment. This line will appear in all phases of the work with the system, to remind you most important keys. To get information about all active keys in current mode with short explanation what the program expects from you please use the help (function key `<F1>` in all possible modes). On the bar under first screen you can find: "F1–help, F3–load file, F5–create new model, F9–exit". As the first exercise with the program we advice you just to play with sample models existing on the distribution disk. To see them press function key `<F3>` (load file) (Fig. 11).

Now the program asks you to enter file name. If you remember the name you may just type it in. Otherwise press the `<Enter>` key to see the file directory. As an illustrative example we choose the file `ROLPA.MOD` to experiment with the model of the medium size farm. To select the file move the cursor to the name of desired file using arrow keys and then press the `<Enter>` key.

```
Wait please  213
```

After a short while when the counter in the small box will reach zero, the model will be loaded and model editing phase should be entered (Fig. 12). (If somebody had used this model before it is possible that the interaction phase will be entered — in this case use `<Alt-F2>` — unlock model function to get into model editing phase.) In model editing phase you can change all names, units, upper and lower bounds for all rows and columns, as well as all coefficients.

What you can see at the screen can be interpreted as set of variables like: `Ara_Land`, `Cows_1`, `Cows_2`, `Rye_S`, ..., set of functions like: `Rye_SQ`, `Potat_SQ`, `Milk_SQ`, ..., with upper and lower bounds and with a matrix of coefficients of the linear functions named above. For example looking at the third row of the spreadsheet you can find that milk production depends on numbers of cows. Namely `Milk_SQ = 3 * Cows_1 + 4 * Cows_2`.

In the model editing phase you are free to enter or modify any names, units, bound or coefficients. To edit any spreadsheet cell just press `<Ins>` key to open it for editing. On the frame of the cell editing box you can find current editing mode flag: `INSERT` or `OVERWRITE`. To toggle the editing mode just press `<Ins>` key again. After pressing function key `<F1>` you can learn about active editing keys (Fig. 13).

We suggest to leave model unchanged and switch to interaction phase. To do this simply press function key `<F4>`. Remember that at this moment the model is saved automatically and locked. It means that you will not be allowed to modify it.

On the interaction screen you can see variables and functions together with bounds and

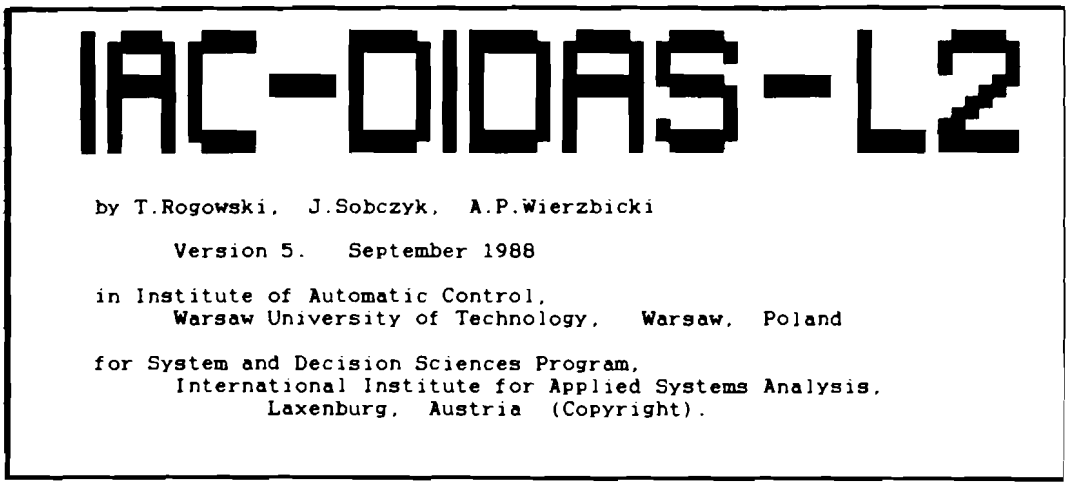

Figure 10: First screen displayed by the IAC-DIDAS-L2.

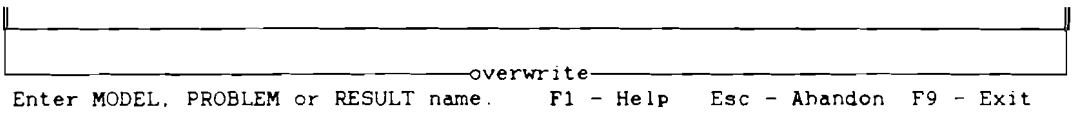

Figure 11: Bottom of the screen after pressing the <F3> key.

IAC - DIDAS - L2 V5.
Model editing

Names	Units	Value	Bounds: lower	Bounds: upper
Rye_SQ	q			2500.
Potat_SQ	q			2.500e4
Milk_SQ	hl			1500.
Ara_Land	ha			
Past_Mea	ha			50.
Fert	q			500.
Tractor	tr.h			5000.
Cow_Stan	number			200.
Workers	person			30.
OatU_Y	100 un		-2.000e4	
Prot_Y	10 kg		-2.000e4	
Dry_M_Y	100 kg		-2.000e4	
OatU_C_Y	100 un		-2.000e4	
OatU_W	100 un		-2.000e4	
Prot_W	10 kg		-2.000e4	
Dry_M_W	100 kg		-2.000e4	

Names	Ara_Land	Cows_1	Cows_2	Rye_S
Units	ha	number	number	ha
Value				
upper	100.	200.	200.	100.
lower				
				25.
		3.	4.	
	-1.			1.
				3.1
		8.3	8.3	5.1
		1.	1.	
		33.1	37.45	
		28.8	33.8	
		-55.9	-55.9	
		0.43	1.89	
		15.7	17.8	
		13.4	15.8	
		-27.1	-27.1	

F1 - Help F9 - Exit

Figure 12: Model editing phase.

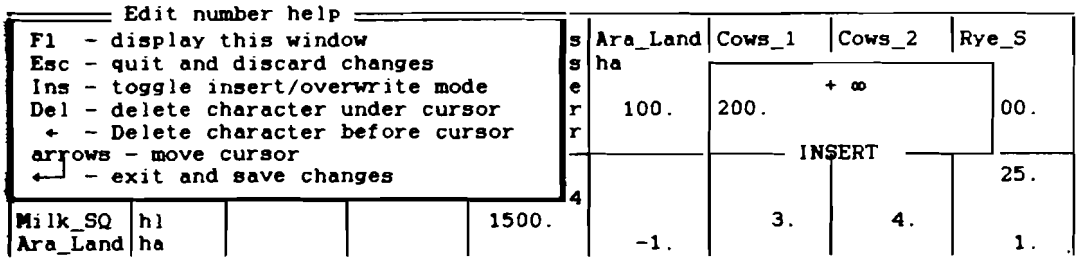

Figure 13: Help window for number editing mode.

several empty columns for problem definition and resulting values (Fig. 14). Just below the window with variables you can see one special row with names of the current model, problem and result together with the comment line for current result (Fig. 15).

In our case name of the model is ROLPA, but names of the problem and result were not given so the program set them to 0. The comment line is also empty. Please observe this fields while playing with the program. You can see that any time you change the reference point the number in the right box will be increased, and any time you change the definition of the multiobjective problem (modifying any bound or row status) the number in the right (result name) box will be increased and number in the middle (problem name) box will be set to the same value.

Let us assume that farm manager (the user of our program) needs to maximize rye and potato production as well as the production of milk and wants to minimize fertilizer consumption. To inform the program what should be maximized or minimized just move the cursor to the Status column in the proper row and press <Ins> key (Fig. 16).

Now you can see all possibilities. Please move the cursor to the corresponding item and press <Enter> key. Repeat this operation for all required objectives. You can also modify bound definitions but we assume that you don't want to do it. After specifying such problem definition we have to compute utopia point and nadir point approximation. When you press the function key <F6> it will be done in several steps. In described problem there are three maximized and one minimized objective. Such definition results in four optimization runs a one for each objective. After each run a short beep is sound and numbers in Utopia and Nadir columns are updated. Remember that numbers in nadir column specify only approximation of the mathematic nadir point while numbers in utopia column reflect actual coordinates of the utopia point.

This two points specify actual limits for each objective. This means that you cannot expect a solution which will have better value for any objective than appropriate coordinate of the utopia point, and no solution will have worse value for any objective than nadir point.

If you know the problem well you can start interaction specifying a reference point, but our advice is to compute first the neutral solution which is in some sense a compromise solution. To do it please press function key <F8>. After single optimization run you can see the result (Fig. 17). Neutral solution function is very useful because it sets also reasonable starting values for reference point and scale.

In contrast to utopia point which is in typical case not attainable and nadir point which is attainable but not efficient, the neutral solution is an attainable and efficient solution.

IAC - DIDAS - L2 V 5. Interaction	Names	Ara_Land	Cows_1	Cows_2	Rye_S	Rye_F
	Upper bound	100.	200.	200.	100.	100.
	Value					
	Lower bound	0.	...	...	...	...

	ROLPA	0	0

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	User scale
Rye_SQ		...			...		2500.	...
Potat_SQ		...			...		2.500e4	...
Milk_SQ		...			...		1500.	...
Ara_Land		...			...		...	...
Past_Mea		...			...		50.	...
Fert		...			...		500.	...
Tractor		...			...		5000.	...
Cow_Stan		...			...		200.	...
Workers		...			...		30.	...
OatU_Y		-2.000e4			...		...	...
Prot_Y		-2.000e4			...		...	...
Dry_M_Y		-2.000e4			...		...	...

Figure 14: Interaction phase.

	ROLPA	0	0

Figure 15: Fragment of the window with names of the model, problem and result.

IAC - DIDAS - L2 V 5. Interaction	Names	Ara_Land	Cows_1	Cows_2	Rye_S	Rye_F
	Upper bound	100.	200.	200.	100.	100.
	Value					
	Lower bound	0.	...	...	...	...

	ROLPA	0	0

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	User scale
Rye_SQ							2500.	...
Potat_SQ		Minimize Maximize Floating Stabilize					2.500e4	...
Milk_SQ							1500.	...
Ara_Land		...			...		...	...
Past_Mea		...			...		50.	...
Fert		...			...		500.	...
Tractor		...			...		5000.	...
Cow_Stan		...			...		200.	...
Workers		...			...		30.	...
OatU_Y		-2.000e4			...		...	...
Prot_Y		-2.000e4			...		...	...
Dry_M_Y		-2.000e4			...		...	...

Figure 16: Modifying the status of the row.

IAC - DIDAS - L2 V 5. Interaction		Names	Ara_Land	Cows_1	Cows_2	Rye_S	Rye_F
		Upper bound	100.	200.	200.	100.	100.
		Value	76.42		65.58	34.64	
		Lower bound	0.	...	...	...	...
ROLPA	0	0	Neutral solution				

Names	Status	Lower bound	Utopia	Value	Reference point	Nadir	Upper bound	Automatic scale
Rye_SQ	Max	...	2449.3	866.12	866.12	0.	2500.	2449.3
Potat_SQ	Max	...	1.250e4	4420.26	4420.26	0.	2.500e4	12500.
Milk_SQ	Max	...	741.84	262.33	262.33	0.	1500.	741.84
Ara_Land		...			...		...	...
Past_Mea		...		10.23	...		50.	...
Fert	Min	...	0.	264.12	264.12	408.61	500.	408.61
Tractor		...		2612.22	...		5000.	...
Cow_Stan		...		65.58	...		200.	...
Workers		...		7.18	...		30.	...
OatU_Y		-2.000e4			...		...	...
Prot_Y		-2.000e4			...		...	...
Dry_M_Y		-2.000e4		-954.15	...		...	...

Figure 17: Neutral solution.

It is possible to play with the problem typing numbers and comparing them but in most cases you will find it more convenient to increase or decrease reference point and compare solutions using the form of a bar chart.

Switching to graphic mode can be done using function key <F7>. Do not worry about proper graphics display driver selection this will be done automatically and for most standard graphic adapters like CGA, Hercules, EGA, VGA and many others it works very well. If you have any nonstandard adapter and after pressing the key <F7> the screen image will be destroyed try to use <Esc> key to return back to the text mode. Assuming that you have no problems with graphic adapter let us continue our session in the graphic mode.

Analyzing numbers in the spreadsheet or bar chart the user may want to increase the production of the milk and may allow to use more fertilizers. To set reference point in graphic mode you should move the cursor to the objective you want to modify and press the <Ins> key. Now you can move small blinking triangle marker up or down using arrow keys. When marker will reach desired position type <Enter> to return to the objective selection. After setting all desired reference points press function key <F5> to run optimization.

After first optimization you can find that you want also bigger rye production so let us modify reference point for Rye_SQ and run optimization again (function key <F5>). Looking at the screen it can be found that rye production can be increased with very small loss on the potato production with small increase of the fertilizers consumption. So let us try to increase the rye production more. After setting new reference point and optimization run we may find that now the loss on the potato production is too big, so we move reference point for the rye to the previous position. Let us look at the fertilizers consumption, it may be considered a bit too big so lets lower the reference point for them and run optimization again.

Let us assume that this solution seems to be satisfactory, so look at the details. Press

IAC - DIDAS - L2 V5. Interaction			Names	Ara_Land	Cows_1	Cows_2	Rye_S	Rye_F
			Value 1.	76.42		65.58	34.64	
			Value 2.	89.99	0.	100.33	40.7	0.
1. ROLPA	5	5	Neutral solution					
2. ROLPA	0	4						

1.Names	St.	Lower b.	Utopia	Value	Ref. p.	Nadir	Upper b.	Scale
Rye_SQ	Max	...	2449.3	866.12	866.12	0.	2500.	2449.3
Potat_SQ	Max	...	1.250e4	4420.26	4420.26	0.	2.500e4	12500.
Milk_SQ	Max	...	741.84	262.33	262.33	0.	1500.	741.84
Ara_Land		...			...		...	...
Past_Mea		...		10.23	...		50.	...
Fert	Min	...	0.	264.12	264.12	408.61	500.	408.61
Tractor		...		2612.22	...		5000.	...
2.								
Rye_SQ	Max	...	2449.3	1017.54	1168.34	0.	2500.	1283.41
Potat_SQ	Max	...	1.250e4	3469.44	4420.26	0.	2.500e4	8092.24
Milk_SQ	Max	...	741.84	401.3	437.19	0.	1500.	305.39
Ara_Land		...		0.	...		...	...
Past_Mea		...		15.64	...		50.	...
Fert	Min	...	0.	320.14	286.43	408.61	500.	286.84
Tractor		...		3496.08	...		5000.	...

Figure 18: Comparison of two results.

IAC - DIDAS - L2 V5. Interaction			Names	Ara_Land	Cows_1	Cows_2	Rye_S	Rye_F
			Value 1.	61.16	0.	80.8	28.6	5.01
			Value 2.	81.28	0.	94.17	37.46	5.84
1. ROLPA	5	8	The best result					
2. ROLPA	5	6						

1.Names	St.	Lower b.	Utopia	Value	Ref. p.	Nadir	Upper b.	Scale
Rye_SQ	Max	...	2449.3	714.93	1155.78	0.	2500.	1295.97
Potat_SQ	Max	...	1.250e4	1667.5	4420.26	0.	2.500e4	8092.24
Milk_SQ	Max	...	741.84	323.2	429.65	0.	1500.	312.93
Ara_Land		...		0.	...		...	...
Past_Mea		...		31.75	...		50.	...
Fert	Min	...	0.	240.23	283.92	408.61	500.	284.33
Tractor		...		2348.75	...		5000.	...
2.								
Rye_SQ	Max	...	2449.3	936.45	1155.78	0.	2500.	1295.97
Potat_SQ	Max	...	1.250e4	3050.73	4420.26	0.	2.500e4	8092.24
Milk_SQ	Max	...	741.84	376.69	429.65	0.	1500.	312.93
Ara_Land		...		0.	...		...	...
Past_Mea		...		37.01	...		50.	...
Fert	Min	...	0.	310.09	283.92	408.61	500.	284.33
Tractor		...		2975.74	...		5000.	...

Figure 19: The best results.

<Esc> key to return to the text mode. Now we can find that for such production 9 persons are required. It is too many for us so we try to reduce the employment. We can introduce the Workers to the set of objectives by modifying status of the row. Observe that after this operation all numbers in utopia, nadir and value columns were marked with dark color. It means that they do not correspond to the current problem and result. For the new problem we want to calculate the neutral solution so we use function key <F8>. But it is necessary to calculate utopia and nadir points first! Do not worry about it, the program remembers such dependencies and will do it for you automatically.

After waiting a while when the new utopia, nadir and neutral solution is calculated we may want to compare this solution with the previous one. To do it let us switch double window mode on using <Alt-F6> key (Fig. 18).

Now we can find that in fact there are only 7 workers required now but production of the milk and rye decreased. Interesting fact is only that production of potatoes has increased in this case. It may be the result of substitution of rye in the fodder by potatoes.

To obtain similar result as in the previous problem specify similar values for the reference point, and run optimization again using function key <F5> (exactly as in the graphic mode).

Now we want to compare current result with the last result of the previous problem. To do this we have to reorder results in the pick list. Let us type <Alt-F3> key, move cursor down to the result numbered 4 and press the <Enter> key. The result was moved to the top of the list. Now move last result numbered 6 to the top in the order to modify it. This should be done because in the double window mode two top results from the pick list may be examined but only the top one can be modified. To exit the pick list press <Esc> key.

Now we can see that we have 7 workers as it was in solution number 4. Lets try to reduce the employment more. Let us specify 5 as the reference point for Workers, and run the optimization again.

We have obtained a solution with only 6 persons employed but the production of potatoes decreased very much. The conclusion is that it is not wise to reduce employment in our farm.

So we return to solution number 6 using the pick list. (Press <Alt-F3> then move cursor to result 6, press <Enter> and then <Esc> key.)

Remembering that all numbered problems and results are considered to be temporary and are removed when there are more than 10 results in the memory, we have to rename and save the interesting results using function key <F2> in the pick list window. After pressing it we are prompted to enter the name for result and then for corresponding problem if it was not saved before.

8 References

Dreyfus, S. (1984). Beyond rationality. In M. Grauer, M. Thompson, A. P. Wierzbicki (eds), Plural Rationality and Interactive Decision Processes, Proceedings, Sopron, 1984. Lecture Notes in Economics and Mathematical Systems, Vol.248. Springer Verlag, Berlin.

Gorecki, H., J. Kopytowski, T. Rys, M. Zebrowski (1983). A multiobjective procedure for project formulation – design of chemical installation. In Grauer,M, A.P. Wierzbicki (eds),Interactive Decision Analysis. Springer Verlag, Berlin.

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. Lecture Notes in Economic and Mathematical Systems . Springer Verlag, Berlin.

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. Lecture Notes in Economic and Mathematical Systems . Springer Verlag, Berlin.

Lewandowski, A., M. Grauer, A. P. Wierzbicki (1983). DIDAS — theory, implementation and experiences. In M. Grauer, A. P. Wierzbicki (eds), Interactive Decision Analysis, Proceedings Laxenburg 1983. Lecture Notes in Economic and Mathematical Systems, Springer Verlag, Berlin.

Lewandowski, A., T. Kreglewski, T. Rogowski, A. P. Wierzbicki (1988). Decision Support Systems of DIDAS Family (Dynamic Interactive Decision Analysis and Support). In A. Lewandowski, A.P. Wierzbicki, (eds), Theory, Software and Testing Examples for Decision Support System. WP-88-071, IIASA, 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. IIASA WP-86-10. International Institute for Applied Systems Analysis, Laxenburg, Austria.

Murtagh, B. A. and M. A. Sanders (1977). MINOS — A large-scale nonlinear programming system. User guide. Technical Report, Systems Optimization Laboratory, Stanford University.

Wierzbicki, A. P. (1986). On the completeness and constructiveness of parametric characterizations to vector optimization problems. *OR Spektrum* 8, 73–87.

A A shortened spreadsheet format of the tutorial model of multiobjective diet selection.

IAC - DIDAS - L2 V5. Model editing					Names	Rolls	Cereals	Butter	Fruitfr
					Units	50 g	50 g	10 g	150 g
					Value				
Names	Units	Value	Bounds:		upper	3.	2.	3.	2.
			lower	upper	lower				
Cost				120.		6.	3.	5.	14.
Taste	artun		4.	40.		3.	1.	2.	4.
Stimulus	artun		4.	80.		3.	3.	4.	3.
Callorie	kcal		150.	1200.		124.	179.	75.	79.
Protein				60.		4.	3.	0.1	0.5
Carbohyd			10.	100.		26.	36.		11.
Fats				80.		1.	2.	8.	0.5
Calcium			180.	720.		8.	10.	2.	9.
Magnesiu				400.		12.	23.	0.2	5.
Phosphor				1000.		42.	103.	1.6	13.
Iron				16.		1.	1.		0.4
Vit.A			100.	1600.				270.	160.
Vit.B				3.		0.12	0.14		6.0e-2
Vit.C				200.					30.
Vit.PP				10.		0.4	1.	1.0e-2	0.23

B Data file format for IAC–DIDAS–L2

All data files read or written by IAC-DIDAS-L2 are standard ASCII files. The only exception is that in comments and in cell texts any 8-bit characters from IBM extended ASCII code may appear.

Every data file for IAC-DIDAS-L2 consists of two sections:

1. Header
   Header contains global information about file contents and related files.

2. Spreadsheet_data_blocks
   This section contains state of a number of spreadsheets. Spreadsheet data blocks can be intermixed with comment lines. Any text information can appear between end_of_spreadsheet_data_marker and spreadsheet_data_begin_marker under condition that every comment line has to begin with at least one space. Order of spreadsheets in the data file is not important and can be changed with no consequence.

REMARK: In following detailed descriptions of file sections every numbered item is stored as single line of text except: position 8. (comments) in header description and position 3. (cell_data_records) in spreadsheet data block format description. This two positions can be interpreted as sequence of any number of comment lines or cell data lines.

B.1 Data file header

Data file header has following structure:

1. load_counter
   integer number

2. identification_string:
   IAC-DIDAS-L2 V5.00

3. data_file_type:
   integer number:

   1 — external model
   2 — model
   3 — variant of the model (NOT USED IN IAC–DIDAS–L2)
   4 — problem - user scale
   5 — result

   12 — LOCKED model
   13 — CALCULATED variant of the model
   14 — CALCULATED problem - user scale
   15 — CALCULATED result

   24 — problem - automatic scale

   34 — CALCULATED problem - automatic scale

4. file_name (with no file extension)
5. creation_date
6. parent_file_name (with no file extension)
7. parent_creation_date
8. comments
   any number of text lines starting with space
9. end_of_header_marker
   $

B.2 Spreadsheet data block

Spreadsheet data block format:

1. spreadsheet_data_begin_marker
   #spreadsheet_name
2. spreadsheet_attributes
   %spreadsheet_status, X_dimension, Y_dimension

   Spreadsheet status is saved as integer number but to understand the meaning of particular number in this filed it must be represented as binary or hexadecimal number. In fact this field is used as a binary word containing 16 different binary flags.

   Spreadsheet status bit definitions (hexadecimal values):

S_Empty	= $8000	Spreadsheet marked for delete
S_Changed	= $4000	Spreadsheet was edited
S_Saved	= $2000	Spreadsheet was saved on the disk
S_Temporary	= $1000	Do not save on the disk
S_Zero	= $200	Display 0 as 0. in all cells
S_ReadOnly	= $100	Spreadsheet can not be edited

3. cell_data_records
4. end_of_spreadsheet_data_marker
   #spreadsheet_name.

B.3 Cell data record

Every cell is stored in one line. Contents of the cell [0,0] is the default contents of any cell in the spreadsheet, so only cells with contents different from the default should be stored in the file. The default cell (if present) must be the first cell in the data block. Order of all other spreadsheet cells is not important and can be changed with no consequence.

Cell record format:

```
[x,y]%status=cell_contents
```

Cell status is saved as integer number but to understand the meaning of particular number in this filed it must be represented as binary or hexadecimal number. In fact this field is used as a binary word containing 13 different binary flags and one three bit number determining the type of a cell.

Cell status bit definitions (hexadecimal values):

`C_Empty` = $8000 Cell marked for delete

`C_Locked` = $800 Cell value is being calculated

`C_Default` = $400 Default cell (make a copy before any modification)

`C_Zero` = $200 Display 0 as 0.

`C_ReadOnly` = $100 Cell can not be edited

`C_Pl_Inf` = $80 Cell can contain plus infinity

`C_Mi_Inf` = $40 Cell can contain minus infinity

`C_RefType` = $20 Cell can be displayed as '...'

`C_NotSet` = $10 Cell should be displayed as '...'

`C_NotBuilt` = $8 Cell contains not compiled formula displayed as 'X'
(NOT USED IN IAC-DIDAS-L2)

`C_VarRef` = $4 Formula refers to the parameters
(NOT USED IN IAC-DIDAS-L2)

Cell type definition filed values:

`C_Integer` = $1000 Cell type = integer

`C_Constant` = $2000 Cell type = real

`C_Formula` = $6000 Cell type = formula
(NOT USED IN IAC-DIDAS-L2)

`C_Choice` = $3000 Cell type = choice

`C_Menu` = $7000 Cell type = menu

`C_Text` = $5000 Cell type = text

`C_Tree` = $4000 Cell type = tree node
(CAN NOT BE SAVED ON THE DISK)

Cell contents depending on the cell type:

`C_Constant` : floating point number or "+" or "-" (plus or minus infinity)

`C_Formula` : floating point number (as described above), length: text
(FOR FUTURE EXTENSIONS)

`C_Choice` : key value

`C_Menu` : key value, length: text

`C_Text` : length: text

`C_Tree` : (CAN NOT BE SAVED ON THE DISK)

REMARK: The text in cells of the type: `C_Formula`, `C_Menu`, `C_Text` can be shorter than specified in the length prefix. In such cases it will be filled with spaces on the end to obtain specified length.

REMARK: Some future extensions are documented here but some other can be expected as additional lines (beginning with a character different from the space) between spreadsheet data blocks or in the header (before or after comments).

C The ROLPA model — short explanation

C.1 Introduction

The ROLPA model is being developed in the Institute of Systems Research, Polish Academy of Sciences. The model described here is only small subset of the full model, namely simplified *production* submodel. The full model consists of *production*, *investment* and *market* submodels linked with data base and generator.

The main purpose of the subset described here is to provide a realistic example for testing and experimenting with the IAC-DIDAS-L2 software. Therefore, the model has been simplified as much as possible, with all important relationships retained. The coefficients have been computed using the model generator and artificial data, and therefore we do not suggest to interpret them too carefully.

The model describes a medium size farm which produces milk, rye and potatoes for profit, under such constraints as available labor, equipment, land, machinery and fertilizers. Therefore, production of rye, potatoes and milk should normally be considered as objectives. The manager ("the user") should develop the most satisfactory profile of the production. However, it can happen that other criteria must be taken into consideration. These can be, for example, minimization of fertilizer usage, full utilization of available resources (tractors, workers) or minimization of inbalance in dry mass or oat units. The user has full freedom to choose the set of objectives which should be minimized or maximized. His opinion about required values of these objectives should be expressed in terms of *aspiration levels*. Usually he has quite good knowledge of which production levels are possible and what is the demand for the production. To provide the information about possible options as described by the model, the *utopia* and *nadir* solutions are calculated. The *utopia* point specifies the best value of all objectives computed *separately* — for example, how much milk the farm can produce if rye and potato are not important at all. The same information can be obtained for other objectives. Simultaneously, the worst case can be computed — for instance, if the production of milk is maximal the production of rye and potatoes will be rather low. The worst possible cases for all objectives if one of several of the objectives are optimized, constitute the *nadir point*.

Clearly, the utopia point is not reachable. Therefore, the system can compute the *neutral solution* — the solution, which is in some sense, the closest to the utopia point. This solution gives some idea, what the farm can produce. Usually, however, the proportions between various products are not satisfactory and the decision maker can select aspiration levels different to neutral solution — for example he can specify lower requirements for rye production in order to increase milk production. When satisfactory level for some objective is reached, this objective can be converted into constraint — for instance, if he is satisfied with milk production, he can fix the milk production level, remove milk production from the set of objectives and play only with potato and rye and for example, with fertilizers.

Summarizing, the principles of interaction with the system is rather simple. The following information is requested from the user:

- What is important (i.e. specification of objectives),
- What is the direction of improvement (maximize or minimize),
- What level of objectives is satisfactory (aspiration levels).

The answer which the system can provide is twofold:

- O.K., you can get exactly what you want, but we can suggest the solution which will be better than your wishes in all possible aspects,
- It is not possible to satisfy your requirements, but we can try to find the solution which is as close to your requirements as possible.

The experience shows that several such iterations are necessary to reach satisfactory solution. This follows from the fact that decision making process is a learning process. Therefore, aspiration can change in the course of cumulating experience regarding possible behavior of the system. There are no "convergence forcing mechanisms" built into the system, which restrict selection of new aspiration level. It is possible to implement such mechanisms, but we found them very restrictive and hard to use.

C.2 Variables

ARA–LAND arable land (in hectars)

COWS–1 cows producing 3000 liters per year (number)

COWS–2 cows producing 4000 liters per year (number)

RYE–S area of production of rye for sale (in hectars)

RYE–F area of production of rye for fodder (in hectars)

BARLEY area of production of barley for fodder (in hectars)

POTAT–1S area of production of potatos with lower harvesting technology, for sale (in hectars)

POTAT–2S area of production of potatos with higher harvesting technology, for sale (in hectars)

POTAT–1F area of production of potatos with lower harvesting technology, for fodder (in hectars)

POTAT–2F area of production of potatos with higher harvesting technology, for fodder (in hectars)

MAIZE–GF area of production of maize for green fodder (in hectars)

MAIZE–SF area of production of maize for ensilage fodder (in hectars)

LUCER–GF area of production of lucerne for green fodder (in hectars)

LUCER–SF area of production of lucerne for ensilage fodder (in hectars)

BEET–F area of production of beet for fodder (in hectars)

GRASS–GF area of grass production for green fodder (in hectars)

GRASS–HF area of grass production for hay fodder (in hectars)

PASTURE area of grazing land (in hectars)

STRAW Straw for fodder (in quintals)

C–FODDER Nutritive fodder purchase (in quintals)

WORKERS Number of workers

C.3 Constraints

RYE-SQ rye production for sale (in quintals)

POTAT-SQ potatos production for sale (in quintals)

MILK-SQ milk production for sale (in hectoliters)

ARA-LAND Balance for arable land (in hectars)

PAST-MEAD Balance of meadows and pastures (in hectars)

FERT Fertilizers consumption (in quintals)

TRACTOR Tractive force usage (in tractor hours)

COW-STAND Cow stands (number)

WORKERS Balance of number of workers

OATU-Y Balance of oat units in fodder per year (in hundreds of units, minimal)

PROT-Y Balance of proteins in fodder per year (in tens of kilograms, minimal)

DRY-M-Y Balance of dry mass in fodder per year (in hundreds of kilograms, maximal)

OATU-C-Y Balance of oat units in nutritive fodder per year (in hundreds of units, minimal)

OATU-W Balance of oat units in fodder in winter (in hundreds of units, minimal)

PROT-W Balance of proteins in fodder in winter (in tens of kilograms, minimal)

DRY-M-W Balance of dry mass in fodder in winter (in hundreds of kilograms, maximal)

OAT-C-W Balance of oat units in nutritive fodder in winter (in hundreds of units, minimal)

CFEED Balance of nutritive fodder (in quintals)

WH-2,WH-3,WH-4,WH-5 Balances of working power in agrotechnical periods 2,3,4,5 (in working hours)

STRAW-T Total balance of straw (in quintals)

STRAW-F Balance of straw for fodder (in quintals)

MANURE Balance of manure (in hundreds of quintals)

R-BARLEY,R-LUCERNE,R-MAIZE,L-BEET, L-POTAT,L-LUCERNE Limits of production areas of various crops related to crop rotation conditions (in hectars)

C.4 Equations

RYE–SQ rye production for sale (in quintals)

$$\begin{aligned} Con\_Rye\_SQ &= 25 Rye\_S \\ Con\_Rye\_SQ &\leq 5000 \end{aligned}$$

POTAT–SQ potatos production for sale (in quintals)

$$\begin{aligned} Con\_Potat\_SQ &= 250 Potat\_1S + 250 Potat\_2S \\ Con\_Potat\_SQ &\leq 25000 \end{aligned}$$

MILK–SQ milk production for sale (in hectoliters)

$$\begin{aligned} Con\_Milk\_SQ &= 3 Cows\_1 + 4 Cows\_2 \\ Con\_Milk\_SQ &\leq 1500 \end{aligned}$$

ARA–LAND Balance for arable land (in hectars)

$$\begin{aligned} Con\_Ara\_Land &= Rye\_S + Rye\_F + Barley + Potat\_1S + \\ &\quad Potat\_2S + Potat\_1F + Potat\_2F + Maize\_GF + \\ &\quad Maize\_SF + Lucer\_GF + Lucer\_SF + Beet\_F \\ Con\_Ara\_Land &= Ara\_Land \end{aligned}$$

PAST–MEAD Balance of meadows and pastures (in hectars)

$$\begin{aligned} Con\_Past\_Mead &= Grass\_GF + Grass\_HF + Pasture \\ Con\_Past\_Mead &\leq 50 \end{aligned}$$

FERT Fertilizers consumption (in quintals)

$$\begin{aligned} Con\_Fert &= 3.1 Rye\_S + 3.1 Rye\_F + 3.1 Barley + 3.5 Potat\_1S + \\ &\quad 3.5 Potat\_2S + 3.5 Potat\_1F + 3.5 Potat\_2F + 3.8 Maize\_GF + \\ &\quad 3.8 Maize\_SF + 2.55 Lucer\_GF + 2.55 Lucer\_SF + 4.8 Beet\_F + \\ &\quad 1.5 Grass\_GF + 1.5 Grass\_HF + 1.5 Pasture \\ Con\_Fert &\leq 500 \end{aligned}$$

TRACTOR Tractive force usage (in tractor hours)

$$\begin{aligned} Con\_Tractor &= 8.3Cows\_1 + 8.3Cows\_2 + 5.1Rye\_S + 5.1Rye\_F + \\ &\quad 14.8Barley + 35.4Potat\_1S + 38.7Potat\_2S + 35.4Potat\_1F + \\ &\quad 38.7Potat\_2F + 37.8Maize\_GF + 37.8Maize\_SF + 48.1Lucer\_GF + \\ &\quad 48.1Lucer\_SF + 62.9Beet\_F + 29.2Grass\_GF + 29.2Grass\_HF + \\ &\quad 9.7Pasture \\ Con\_Tractor &\leq 5000 \end{aligned}$$

COW-STAND Cow stands (number)

$$\begin{aligned} Con\_Cow\_Stand &= Cows\_1 + Cows\_2 \\ Con\_Cow\_Stand &\leq 200 \end{aligned}$$

WORKERS Balance of number of workers

$$\begin{aligned} Con\_Workers &= Workers \\ Con\_Workers &\leq 30 \end{aligned}$$

OATU-Y Balance of oat units in fodder per year (in hundreds of units, minimal)

$$\begin{aligned} Con\_OatU\_Y &= 33.1Cows\_1 + 37.45Cows\_2 + 52.5Grass\_GF \\ Con\_OatU\_Y &\leq 30.5Rye\_F + 42Barley + 47.25Potat\_1F + 47.25Potat\_2F + \\ &\quad 140Maize\_GF + 93.1Maize\_SF + 72Lucer\_GF + 37.6Lucer\_SF + \\ &\quad 112.8Beet\_F + 25Grass\_HF + 50Pasture + 0.3Straw + 0.7C\_Fodder \end{aligned}$$

PROT-Y Balance of proteins in fodder per year (in tens of kilograms, minimal)

$$\begin{aligned} Con\_Prot\_Y &= 28.8Cows\_1 + 33.8Cows\_2 \\ Con\_Prot\_Y &\leq 14.7Rye\_F + 30.1Barley + 17.5Potat\_1F + 17.5Potat\_2F + \\ &\quad 77Maize\_GF + 49Maize\_SF + 148Lucer\_GF + 70.4Lucer\_DF + 66Beet\_F + \\ &\quad 75Grass\_GF + 27.5Grass\_HF + 70Pasture + 0.06Straw + 4.5C\_Fodder \end{aligned}$$

DRY-M-Y Balance of dry mass in fodder per year (in hundreds of kilograms, maximal)

$$\begin{aligned} Con\_Dry\_M\_Y &= 21.9Rye\_F + 30.6Barley + 38.5Potat\_1S + 38.5Potat\_2F + \\ &\quad 140Maize\_GF + 98Maize\_SF + 86Lucer\_GF + 68Lucer\_SF + \\ &\quad 121.2Beet\_F + 50Grass\_GF + 42.5Grass\_HF + 47.5Pasture + \\ &\quad 0.85Straw + 0.88C\_Fodder \\ Con\_Dry\_M\_Y &\leq 55.9Cows\_1 + 55.9Cows\_2 \end{aligned}$$

OATU-C-Y Balance of oat units in nutritive fodder per year (in hundreds of units, minimal)

$$
\begin{aligned}
Con\_OatU\_C\_Y &= 0.43Cows\_1 + 1.89Cows\_2 \\
Con\_OatU\_C\_Y &\leq 30.5Rye\_F + 42Barley + 0.7C\_Fodder
\end{aligned}
$$

OATU−W Balance of oat units in fodder in winter (in hundreds of units, minimal)

$$
\begin{aligned}
Con\_OatU\_W &= 15.7Cows\_ + 17.8Cows\_2 \\
Con\_OatU\_W &\leq 30.5Rye\_F + 42Barley + 47.25Potat\_1F + 47.25Potat\_2F + \\
&\quad 93.1Maize\_SF + 37.6Lucer\_SF + 112.8Beet\_F + 25Grass\_HF + \\
&\quad 0.3Straw + 0.7C\_Fodder
\end{aligned}
$$

PROT−W Balance of proteins in fodder in winter (in tens of kilograms, minimal)

$$
\begin{aligned}
Con\_Prot\_W &= 13.4Cows\_1 + 15.8Cows\_2 \\
Con\_Prot\_W &\leq 14.7Rye\_F + 30.1Barley + 17.5Potat\_1F + 17.5Potat\_2F + \\
&\quad 49Maize\_SF + 70.4Lucer\_SF + 66Beet\_F + 27.5Grass\_HF + \\
&\quad 0.06Straw + 4.5C\_Fodder
\end{aligned}
$$

DRY−M−W Balance of dry mass in fodder in winter (in hundreds of kilograms, maximal)

$$
\begin{aligned}
Con\_Dry\_M\_W &= 21.9Rye\_F + 30.6Barley + 38.5Potat\_1F + 38.5Potat\_2F + \\
&\quad 98Maize\_SF + 68Lucer\_SF + 121.2Beet\_F + 42.5Grass\_HF + \\
&\quad 0.85Straw + 0.88C\_Fodder \\
Con\_Dry\_M\_W &\leq 27.1Cows\_1 + 27.1Cows\_2
\end{aligned}
$$

OATU−C−W Balance of oat units in nutritive fodder in winter (in hundreds of units, minimal)

$$
\begin{aligned}
Con\_OatU\_C\_W &= 0.7Cows\_1 \\
Con\_OatU\_C\_W &\leq 30.5Rye\_F + 42Barley + 0.7C\_Fodder
\end{aligned}
$$

CFEED Balance of nutritive fodder in quintals)

$$
\begin{aligned}
Con\_CFeed &= C\_Fodder \\
Con\_CFeed &= 0
\end{aligned}
$$

WH−2, WH−3, WH−4, WH−5 Balances of working power

$$
\begin{aligned}
Con\_WH\_2 &= 16.9Cows\_1 + 16.9Cows\_ + 2Rye\_S + 2Rye\_F + \\
&\quad 4.5Barley + 31Potat\_1S + 31Potat\_2S + 31Potat\_1F + \\
&\quad 31Potat\_2F + 2.5Maize\_GF + 2.5Maize\_SF + 0.6Lucer\_GF + \\
&\quad 0.6Lucer\_SF + 3.8Beet\_F + Grass\_GF + Grass\_HF \\
Con\_WH\_2 &\leq 249Workers
\end{aligned}
$$

$$
\begin{aligned}
Con\_WH\_3 &= 16Cows\_1 + 16Cows\_2 + 1.5Potat\_1S + 1.5Potat\_2S + \\
&\quad 1.5Potat\_1F + 1.5Potat\_2F + 4.4Maize\_GF + 4.4Maize\_SF + \\
&\quad 5.4Lucer\_GF + 5.4Lucer\_SF + 67.7Beet\_F + 4.7Grass\_GF + \\
&\quad 4.7Grass\_HF \\
Con\_WH\_3 &\leq 296Workers
\end{aligned}
$$

$$
\begin{aligned}
Con\_WH\_4 &= 10Cows\_1 + 10Cows\_2 + 6.5Rye\_S + 6.5Rye\_F + \\
&\quad 6.5Barley + 5.4Lucer\_GF + 5.4Lucer\_SF \\
Con\_WH\_4 &\leq 184Workers
\end{aligned}
$$

$$
\begin{aligned}
Con\_WH\_5 &= 24.7Cows\_1 + 24.7Cows\_2 + 167Potat\_1S + 27.5Potat\_2S + \\
&\quad 16Maize\_GF + 16Maize\_SF + 5.5Lucer\_GF + 5.5Lucer\_SF + \\
&\quad 15Beet\_F + 4.7Grass\_GF + 4.7Grass\_HF \\
Con\_WH\_5 &\leq 434Workers
\end{aligned}
$$

STRAW–T Total balance of straw (in quintals)

$$
\begin{aligned}
Con\_Straw\_T &= 15Cows\_1 + 15Cows\_2 + Straw \\
Con\_Straw\_T &\leq 37Rye\_S + 37Rye\_F + 21Barley
\end{aligned}
$$

STRAW–F Balance of straw for fodder (in quintals)

$$
\begin{aligned}
Con\_Straw\_F &= Straw \\
Con\_Straw\_F &\leq 21Barley
\end{aligned}
$$

MANURE Balance of manure (in hundreds of quintals)

$$
\begin{aligned}
Con\_Manure &= 0.6Ara\_Land \\
Con\_Manure &\leq Cows\_1 + Cows\_2
\end{aligned}
$$

R–BARLEY,R–LUCERNE,R–MAIZE,L–BEET, L–POTAT,L–LUCERNE Limits of production areas of various crops related to crop rotation conditions (in hectars)

$$
\begin{aligned}
Con\_R\_Barley &= Barley \\
Con\_R\_Barley &\leq Potat\_1S + Potat\_2S + Potat\_1F + Potat\_2F + Beet\_F
\end{aligned}
$$

$$
\begin{aligned}
Con\_R\_Lucern &= 0.5Lucer\_GF + 0.5Lucer\_SF \\
Con\_R\_Lucern &\leq Potat\_1S + Potat\_2S + Potat\_1F + Potat\_2F + Beet\_F
\end{aligned}
$$

$$
\begin{aligned}
Con\_R\_Maize &= Barley + Maize\_GF + Maize\_SF \\
Con\_R\_Maize &\leq Potat\_1S + Potat\_2S + Potat\_1F + Potat\_2F + Beet\_F
\end{aligned}
$$

$$
\begin{aligned}
Con\_Beet &= Beet\_F \\
Con\_Beet &\leq 0.33 Ara\_Land
\end{aligned}
$$

$$
\begin{aligned}
Con\_Potat &= Potat\_1S + Potat\_2S + Potat\_1F + Potat\_2F \\
Con\_Potat &\leq 0.5 Ara\_Land
\end{aligned}
$$

$$
\begin{aligned}
Con\_L\_Lucern &= Lucer\_GF + Lucer\_SF \\
Con\_L\_Lucern &\leq 0.2 Ara\_Land
\end{aligned}
$$

D Dynamic Interactive Decision Analysis and Support System IAC-DIDAS-L1

IAC-DIDAS-L1 is a pilot version of decision support system based on reference point methodology. The theoretical and methodological background of decision making and support in the DIDAS system was present in first part of this paper. The system was written in Fortran-77 language. The maximization is performed through a linear programming algorithm called solver, written in Fortran (e.g. a linear programming subroutine from IMSL Library). System supports the following general functions:

1. The definition and edition of a substantive model of the decision situation, in a linear programming form. IAC-DIDAS-L1 uses the MPS format of linear programming for this purpose, while IAC-DIDAS-L2 supports model definition and edition in a user-friendly format of a spreadsheet.

2. The specification of a multiobjective decision analysis problem related to the substantive model. This is performed by several commands from the main menu of IAC-DIDAS-L1.

3. The initial multiobjective analysis of the problem, resulting in estimating bounds on efficient outcomes of decisions and in learning about some extreme and some neutral decisions. These functions are also supported by some specific commands from the main menu.

4. The interactive analysis of the problem with the stress on learning by the user of possible efficient decisions and outcomes, organized through systems response to user-specified aspiration levels or reference points for objective outcomes. The system responds with efficient solutions and objective outcomes obtained through the maximization of an achievement function that is parameterized by the user-specified reference points.

D.1 Preparation of the problem

IAC-DIDAS-L1 needs a two input files:

- first — `MODEL` — defined the linear structure of the problem in MPS standard described by Murtagh (1977). The following rules must be taken into account when creating this file:

  1. All objective rows must be defined as equality (E) type.
  2. It is necessary to remember that system modifies this problem by adding some additional rows, columns etc. Names of these additional rows begins with 'mmmm'; therefore the file `MODEL` generated by the user should not contain names beginning with this sequence.
  3. There are no other restriction on the form of MPS file.

- second — `OBJECTIVE.OLD` — contain directives which also can be used during the interactive session. When DIDAS begins the execution, the input stream is associated with the `OBJECTIVE.OLD` file and the commands contained there are executed. If the last command in the file is `STOP`, program terminates. If `STOP` command is not present, the input stream is switched to the terminal. In this way, some initial runs and calculations can be specified by the user as a well purely batch processing can be performed. After

termination run of the program the current status of the program is saved in the file OBJECTIVE.NEW. Evidently, only the structural information is saved — this related to the names and types of objectives, values of reference points, bounds, parameters etc. This file, after renaming to OBJECTIVE.OLD, can be used to restart the system. List of the commands and directives which can be present in this file are discussed below.

D.2 Main menu

The main menu of commands in IAC-DIDAS-L1 is the following:

1. Problem setting phase:
   - ? <Cr> — displays help.
   - MAX | MIN | GUI | FLO | REM *objectivename* <Cr> — includes new objectives (from the list of names of outcomes and decision variables of the model), changes status (to maximized, minimized, guided — that is, corresponding to an equality constraint, or floating — that is, displayed only for information purposes) or removes an objective from the definition of the multiobjective analysis problem.
   - UPP | LOW | FIX *objectivename value* <Cr> — sets bounds for objective values (UPP for upper bounds, LOW for lower bounds, FIX for equality constraints of GUI type; all objectives except of GUI and FLO types must have specified bounds in this phase; defaults are zero and rhs or bounds — as specified in the model).
   - SCA *objectivename value* <Cr> — sets user-specified scaling units for an objective (all objectives except of GUI and FLO types must have specified scaling units in this phase; default is 1).
   - RAS *binary* (0 or 1) <Cr> — sets off or on automatic utopia-reference scaling (after computing utopia point, see further commands, the user-supplied scaling can be replaced by a more convenient type of scaling).
   - EPS *value* <Cr> — sets the value of parameter $0 < \text{eps} < 1$ in the achievement function.
   - XRH *value* <Cr> — sets the value of parameter $\text{rho} > 1$ in the achievement function.
   - EPS | XRH <Cr> — displays the value of parameter eps or rho.
2. Initial analysis phase:
   - FOR *objectivename* <Cr> — results in the calculation of an extreme solution, that is, the optimal solution for a given, single objective.
   - UTO <Cr> — calculates utopia and approximate nadir points (that is, upper and lower bounds for efficient decision outcomes).
   - NAD <Cr> — improves the approximation of nadir point.
   - NEU <Cr> — calculates a neutral solution using scaling coefficients based on utopia-nadir differences.
3. Interactive analysis.
   - RFP | REF *objectivename value* <Cr> — sets reference point for an objective.
   - GO <Cr> — calculates an efficient solution related to the last specified reference point.

- DIS BOU | UTO | SOL | <Cr> — displays numerically bounds, or utopia and nadir points, or the last solution.
- SCN *value* <Cr> — starts the SCAN procedure with the step d = 'value'.
- ACC *objname* — accepts the solution obtained during the SCAN process, when the reference point component corresponding to 'objname' was perturbed, as a new reference point.
- PRI <Cr> — writes the last results on the file RESULTS.
- PSC <Cr> — writes the results of the last scan on the file

4. Results.

- BAS <Cr> — makes possible manipulating with the data base for solution (up to 10 items). After invoking this command the following menu appears at the screen:

  (1) save

  (2) load

  (3) remove

  (4) list

  (5) quit.

  The user ought to select the option number:
  - option (1) save — at this point the program asks:
    save as ?:
    and the user gives a name to the last solution to be saved in the data base,
  - option (2) load — at this point the user gives the names of the data and the solution to be retrieved from the data base,
  - option (3) remove — removes a name from the data base,
  - option (4) list — lists the names saved in the data base,
  - option (5) quit — returns to the main menu.
- STOP <Cr> — ends work with the system.

D.3 Example of diet problem

1. Example of OBJECTIVE.OLD file.

```
name DietDemo
rhs  Rhs
ran
bou  BND
xrho     1.0000000
eps   0.00000000E-01
min  Cost
rfp  Cost          40.000000
sca  Cost          2.0000000
```

```
low  Cost        0.00000000E-01
upp  Cost         100.00000
min  Callorie
rfp  Callorie     400.00000
sca  Callorie     2.0000000
low  Callorie     100.00000
upp  Callorie     1800.0000
max  Taste
rfp  Taste        25.000000
sca  Taste       0.10000000
low  Taste        6.0000000
upp  Taste        100.00000
max  Stimulus
rfp  Stimulus     50.000000
sca  Stimulus    0.20000000
low  Stimulus     4.0000000
upp  Stimulus     60.000000
max  Calcium
rfp  Calcium      300.00000
sca  Calcium      1.0000000
low  Calcium      100.00000
upp  Calcium      600.00000
flo  Vit.A
flo  Rolls
flo  Cereals
flo  Butter
flo  Fruitfre
flo  Milk
flo  Coffee
flo  Carbohyd
```

2. Example of MODEL file.

```
name          dieta
rows
 e  Cost
 e  Taste
 e  Stimulus
 l  Callorie
 l  Protein
 l  Carbohyd
 l  Fats
 l  Calcium
 l  Magnesiu
 l  Phosphor
 l  Iron
 l  Vit.A
```

```
 l  Vit.B
 l  Vit.C
 l  Vit.PP
columns
    Rolls     Cost      5.
    Rolls     Taste     3.
    Rolls     Stimulus  3.
    Rolls     Callorie  124.
    Rolls     Protein   4.
    Rolls     Carbohyd  26.
    Rolls     Fats      1.
    Rolls     Calcium   8.
    Rolls     Magnesiu  12.
    Rolls     Phosphor  42.
    Rolls     Iron      1.
    Rolls     Vit.B     0.12
    Rolls     Vit.PP    0.4
    Cereals   Cost      4.
    Cereals   Taste     2.
    Cereals   Stimulus  1.
    Cereals   Callorie  179.
    Cereals   Protein   3.
    Cereals   Carbohyd  36.
    Cereals   Fats      2.
    Cereals   Calcium   10.
    Cereals   Magnesiu  23.
    Cereals   Phosphor  103.
    Cereals   Iron      1.
    Cereals   Vit.B     0.14
    Cereals   Vit.PP    1.
    Butter    Cost      5.
    Butter    Taste     2.
    Butter    Stimulus  4.
    Butter    Callorie  75.
    Butter    Protein   0.1
    Butter    Carbohyd  0.0
    Butter    Fats      8.
    Butter    Calcium   2.
    Butter    Magnesiu  0.2
    Butter    Phosphor  1.6
    Butter    Iron      0.0
    Butter    Vit.A     270.
    Butter    Vit.B     0.0
    Butter    Vit.C     0.0
    Butter    Vit.PP    0.01
    Fruitfre  Cost      14.
    Fruitfre  Taste     2.
    Fruitfre  Stimulus  0.5
```

```
    Fruitfre  Callorie  79.
    Fruitfre  Protein   0.5
    Fruitfre  Carbohyd  11.
    Fruitfre  Fats      0.5
    Fruitfre  Calcium   9.
    Fruitfre  Magnesiu  5.
    Fruitfre  Phosphor  13.
    Fruitfre  Iron      0.4
    Fruitfre  Vit.A     160.
    Fruitfre  Vit.B     0.06
    Fruitfre  Vit.C     30.
    Fruitfre  Vit.PP    0.23
    Milk      Cost      6.
    Milk      Taste     3.
    Milk      Stimulus  1.
    Milk      Callorie  137.
    Milk      Protein   7.
    Milk      Carbohyd  10.
    Milk      Fats      7.
    Milk      Calcium   295.
    Milk      Magnesiu  30.
    Milk      Phosphor  213.
    Milk      Iron      0.25
    Milk      Vit.A     277.
    Milk      Vit.B     0.73
    Milk      Vit.C     2.5
    Milk      Vit.PP    0.25
    Coffee    Cost      18.
    Coffee    Taste     1.
    Coffee    Stimulus  12.
    Coffee    Callorie  2.
rhs
    Rhs       Cost      0.
    Rhs       Taste     0.
    Rhs       Stimulus  0.
    Rhs       Callorie  0.
    Rhs       Protein   60.
    Rhs       Carbohyd  0.
    Rhs       Fats      80.
    Rhs       Calcium   0.
    Rhs       Magnesiu  400.
    Rhs       Phosphor  1000.
    Rhs       Iron      16.
    Rhs       Vit.A     0.
    Rhs       Vit.B     3.
    Rhs       Vit.C     200.
    Rhs       Vit.PP    10.
bounds
```

```
 up BND       Rolls     5.
 up BND       Cereals   2.
 up BND       Butter    5.
 up BND       Fruitfre  2.
 up BND       Milk      3.
 up BND       Coffee    3.
 lo BND       Rolls     0.
 lo BND       Cereals   0.
 lo BND       Butter    0.
 lo BND       Fruitfre  0.
 lo BND       Milk      0.
 lo BND       Coffee    0.
endata
```