WORKING PAPER

A Mathematical Programming Package for Multicriteria Dynamic Linear Problems HYBRID. Methodological and User Guide to Version 3.03

Marek Makowski
IIASA
and
Janusz Sosnowski
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland

January, 1988
WP-88-002

IIASA
International Institute
for Applied Systems Analysis

A Mathematical Programming Package for Multicriteria Dynamic Linear Problems HYBRID. Methodological and User Guide to Version 3.03

Marek Makowski
IIASA
and
Janusz Sosnowski
Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland

January, 1988
WP-88-002

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

Foreword

One of the main problems in development the decision support software is the availability of efficient optimization algorithms. These algorithms, when applied in decision support systems should possess several criteria - like robustness, efficiency, high speed and low memory requirements. Moreover, the special structure of the optimization problems arising in such applications should be taken into account. All these facts motivated the System and Decision Sciences Program to investigate all these topics.

This paper presents the result of such collaborative effort. The HYBRID system, developed in the Institute of System Research of the Polish Academy of Sciences is the implementation of original, non simplex algorithm especially suited for solving dynamic multiple criteria problem. Except of high efficiency, this algorithm is especially interesting for microcomputers with small available memory.

This research is being performed upon a contracted study agreement between the International Institute for Applied Systems Analysis and the Polish Academy of Sciences.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program.

Contents

A Mathematical Programming Package for Multicriteria Dynamic Linear Problems HYBRID. Methodological and User Guide to Version 3.03

Marek Makowski and Janusz Sosnowski

1. INTRODUCTION

The purpose of this report is to provide sufficient understanding of mathematical, methodological and theoretical foundations of the HYBRID package as well as information necessary for using the package. Section 1 contains executive summary, short program description and general remarks on solution techniques and package implementation. Section 2 contains mathematical formulation of various types of problems that can be solved by HYBRID. Section 3 presents methodological problems related to solution techniques. Section 4 presents foundations of the chosen solution technique and documents the computational algorithm. Section 5 contains short discussion of testing examples. Section 6 presents the way of choosing various options provided by the package. Sections 7 contains guidelines for formulation and modification of a problem which is to be solved or at least processed by HYBRID. Section 8 describes the way in which HYBRID provides diagnostics and results. Sections 9 presents a short tutorial example. Last two sections contain conclusions and references. The appendix contains the specification of the MPS standard for input data and an example of the MPS format input file.

1.1. Executive summary

HYBRID is a mathematical programming package which includes all the functions necessary for the solution of linear programming problems. The current version of HYBRID is referred to further on as HYBRID 3.03. HYBRID 3.03. may be used for solving both static and dynamic LP problems (in fact also for problems with structure more general then the classical formulation of dynamic linear problems). HYBRID 3.03. may be used for both single- and multi-criteria problems. Since HYBRID is designed for real-life problems, it offers many options useful for diagnostic and verification of a problem being solved.

HYBRID is a member of a decision analysis and support system DIDAS family which is designed to support usage of multicriteria optimization tools. HYBRID can be used by an analyst or by a team composed of a decision maker and an analyst or - on last stage of application - by a decision maker alone. In any case we will speak further on about a user of a HYBRID package.

HYBRID can serve as a tool which helps to choose a decision in a complex situation in which many options may and should be examined. Such problems occur in many situations, such as problems of economic planning and analysis, many technological or engineering design problems, problems of environmental control. To illustrate possible range

of applications, let us list problems for which the proposed approach either has been or may be applied: planning of agriculture production policy in a decentralized economy (both for governmental agency and for production units) [2], flood control in a watershed [25], planning formation and utilization of water resources in an agricultural region, scheduling irrigation, planning and design of purification plant system for water or air pollution.

To avoid a possible misleading conclusion that the usage of HYBRID may replace a real decision maker, we should stress that HYBRID is designed to help a decision maker to concentrate on real decision making while HYBRID takes care on cumbersome computations and provides information that serves for analysis of consequences of different options or alternatives. A user is expected to define various alternatives or scenarios, changing his preferences and priorities when learning about consequences of possible decisions. This problem is shortly discussed in Section 5 and illustrated in the testing example.

HYBRID could be used for that purpose as a "stand alone" package, however - after a possible modification of a problem in an interactive way - one can also output the MPS-format file from HYBRID to be used in other packages. The later approach can be used also for a transformation of a multicriteria problem to an equivalent single-criteria LP. Diagnostic functions are not performed by many other linear programming packages, e.g., by MINOS – it is interesting to note that the authors of MINOS actually advise the user to debug and verify the problem with another package before using MINOS.

HYBRID can be used for solving any linear programming problem but it is specially useful for dynamic problems; this covers a wide area of applications of operation researches. Many optimization problems in economic planning over time, production scheduling, inventory, transportation, control dynamic systems can be formulated as linear dynamic problems [17]. Such problems are also called multistage or staircase linear programming problems [18],[19]. A dynamic problem can be formulated as an equivalent large static LP and any commercial LP code may be used for solving it, if the problem corresponds to single objective optimization. For multicriteria problems, a preprocessor may be used for transformation of that problem to an equivalent LP one. The system DIDAS, described in [28], is a package that is composed of preprocessor and postprocessor for handling transformation of multicriteria problem and processing results respectively [20]. Those pre- and postprocesors are linked with an LP package. HYBRID 3.03. has generally similar structure. The main difference is that - instead of an LP package - another algorithm is applied, which exploits the dynamics of a problem. Similarly as some other systems of DIDAS family, HYBRID has the advantage of handling a problem as a dynamic one which results in an easy way of formulation of criteria and of interpretation of results, since one may refer to one variable trajectory contrary to a "static" formulation of dynamic problems which involves separate variables for each time period.

HYBRID has been designed more for real-world problems that require scenario analysis than for academic (e.g., randomly generated) problems. Thus HYBRID is oriented towards an interactive mode of operation in which a sequence of problems is to be solved under varying conditions (e.g., different objective functions, reference points, values of constraints or bounds). Criteria for multiobjective problems may be easily defined and updated with the help of the package.

The HYBRID 3.03 is available in two versions: one for mainframes and one for PC. Each version requires a FORTRAN compiler that accepts full standard of FORTRAN-77. Implementation on a particular computer requires only changes in a routine that reads system date and time.

The package has been tested on VAX 11/780 (for f77 compiler under Berkeley UNIX 4.2) and on a PC compatible with PC IBM/AT. The minimal configuration of PC consists of 512kB RAM. Intel coprocessor 80287 is strongly recommended (in fact required by some FORTRAN compilers).

1.2. SHORT PROGRAM DESCRIPTION

1.2.1. Preparation of a problem formulation

A problem to be solved should be defined as a mathematical programming model. Formulation of a mathematical programming problem is a complex task and this paper is not devoted to discuss this question in details. Therefore this section is aimed at providing only a short summary of a recommended approach.

Firstly, a set of variables that sufficiently describe the problem - for the sake of the desired analysis - should be selected. It is desired - however not necessary - to define the problem in such a way as to possibly exploit the problem structure (further on referred to as a dynamic problem). Secondly, a set of constraints which defines a set of admissible (i.e. acceptable or recognized as feasible by a decision maker) solutions should be defined. Finally a set of criteria which could serve for a selection of a solution should be defined.

The formal definition of criteria can be performed in HYBRID in an easy way. However, it should be stressed that any definition of a complex problem usually requires cooperation of a specialist - who knowns the nature and background of the problem to be solved - with a system analyst who can advise on a suitable way of formal definition. It should be clearly pointed out that a proper definition can substantially improve the use of any computational technique. For small problems used for illustration of the method, it is fairly easy to define a problem. But for real life problems, this stage requires a close cooperation between a decision maker and a team of analysts as well as a substantial amount of time and resources.

For real life problems, the following steps are recommended:

1. Mathematical formulation of the problem being solved should be defined.
2. A data base for the problem should be created. This may be done on PC with a help of a suitable commercial product (such as Framework, dBase, Symphony, Lotus 1-2-3). Original data should be placed in this data base. A user need not worry about possible range of quantities (which usually has an impact on computational problems) because HYBRID provides automatic scaling of the problem.
3. Verification of the data base and of the model formal definition should be performed.
4. The corresponding MPS standard file (cf Appendix) should be created. This may be done by a specialized problem generator (easily written by a system analyst), or an universal generator such as GEMINI (developed at IIASA) or GAMMA (part of FMPS package on UNIVAC) or by any appropriate utility program of data base software. We strongly discourage the user from creating the MPS file with help of a standard text editor.

1.2.2. Problem verification

This stage serves for the verification of model definition which is crucial for real application of any mathematical programming approach.

First stage consists of preprocessing the MPS file by HYBRID, which offers many options helpful for that task. HYBRID points to possible sources of inconsistency in model definition. Since this information is self-explaining, details are not discussed here. It is also advisable to examine the model printout by rows and by columns, which helps to verify model specification and may help in tracing possible errors in MPS file generation.

Second stage consist of solving optimization problems for selected criteria which helps in the analysis of consistency of solutions. For larger problems a design and application of a problem oriented report writer is recommended. HYBRID generates a "user_file" for that purpose which contains all information necessary for the analysis of a solution.

After an analysis of a solution, a user may change any of the following parameters: values of coefficients, values of constraints and also any parameters discussed in next section. This may be done with help of the interactive procedure which instead of MPS file uses "communication region" that contains problem formulation processed by HYBRID. Therefore, a user needs no longer to care about original MPS file which has the backup function only.

1.2.3. Problem analysis

Problem analysis consist of consecutive stages:

- analysis of obtained solution
- modification of the problem
- solution of modified problem.

Analysis of a solution consists of following steps (some of which are optional):

1. The user should examine of values of selected criteria. Since the solution obtained in HYBRID is Pareto optimal, the user should not expect improvement in any criteria without worsening some other criteria. But values of each criterion can be mutually compared. It is also possible to compute the best solutions for each criterion separately. A point (in criteria space) composed of best solutions is called the "utopia" point (since usually it is not attainable). HYBRID provides also a point composed of worst values for each criterion. This point is called "nadir" point. Such information help to define a reference point (desired values of criteria) because it is reasonable to expect values of each criterion to lie between utopia and nadir point.
2. The user may also at this stage make modifications to the original problem without involving the MPS file.
3. For dynamic problems, HYBRID allows also for easy examination of trajectories (referred to by so called generic name of a variable).

Modification of the problem may be done in two ways:

1. At this stage, the user can modify the formulation of the original problem. But main activity in this stage is expected after the model is well defined and verified and no longer requires changes in parameters that define the set of admissible (acceptable) solutions. It should be stressed, that each change of this set usually results in change

of the set of Pareto-optimal solutions and both utopia and nadir points should be computed again.

2. If the values of all constraints and coefficients that define the admissible set of solutions are accepted, the user should start with computations of utopia point. This can be easily done in an interactive way. After utopia and corresponding nadir points are obtained (which requires n solutions of the problem, where n is the number of criteria defined) the user can also interactively change any number of the following parameters that define the selection of an efficient solution to the multicriteria problem:
   - Reference point (i.e. desired values for each criterion) might be changed. This point may be attainable or non-attainable (cf sec.2.4.).
   - Weights attached to each criterion can be modified.
   - Reference trajectories in dynamic case can be changed as reference points.
   - Regularization parameters in selection function can be adjusted.
3. Additionally, the user can temporarily remove a criterion (or a number of criteria) from analysis. This option results in the computation of a Pareto optimal point in respect to remaining "active" criteria but values of criteria that are not active are also available for review.

Solution of a problem. The problem defined by a user (after possible modification) is transformed by HYBRID to an equivalent LP problem which is solved without interaction of a user (an experienced user may however have an access to the information that characterizes the optimization run).

1.2.4. Remarks relevant to dynamic problems.

HYBRID allows for solving both static and dynamic LP problems. Static problems can be interpreted as problems for which a specific structure is not recognized nor exploited. But many real life problems have specific structure which - if exploited - can result not only in much faster execution of optimization runs but also remarkably help in problem definition and interpretation of results.

Numerous problems have dynamic nature and it is natural to take advantage of its proper definition. HYBRID offers many options for dynamic problems, such as:

1. In many situations, the user may deal with generic names of variables. A generic name consists of 6 first characters of a name while 2 last characters corresponds to the period of time. Therefore, the user may for example refer to the entire trajectory (by generic name) or to value of a variable for a specific time period (by full name). Such approach corresponds to a widely used practice of generating trajectories for dynamic problems.
2. The user may select any of 4 types of criteria that correspond to practical applications. Those can be defined for each time period (together with additional "global" conditions), but this requires rather large effort. Therefore, for dynamic problems, criteria are specified just by the type of criterion and the generic name of the corresponding variable. Types of criteria are discussed in details later.
3. A problem can be declared as a dynamic one by the definition of periods of time. For a dynamic problem, additional rules must be observed. These rules correspond to the way in which the MPS file has to be sorted and to the way in which names for rows

and columns are selected. These rules follow a widely accepted standard of generation of dynamic problems. The formulation of a dynamic problem, which is accepted by HYBRID is actually an extension of the classical formulation of dynamic problem (cf Section 2.2.). In this formulation a model may contain also a group of constraints that do not follow the standard of state equations.

1.2.5. General description of the package and data structure

The package is constructed in modules to provide a reasonably high level of flexibility and efficiency. This is crucial for a rational use of computer resources and for planned extensions of the package and possible modification of the algorithm (see Section 5).

The package consists of three subpackages:

- Preprocessor that serves to process data, enables a modification of the problem, performs diagnostics and may supply information useful for verification of a problem. The preprocessor also transforms a multicriteria problem to a parametric single criteria optimization problem, helps in the interactive change of parameters, etc.
- Optimization package called solver of a relevant optimization problem (either static or dynamic)
- Postprocessor that can provide results in the standard MPS format and can also generate the "user file" which contains all information needed for the analysis of a solution; the later option makes it easier to link HYBRID to a specialized report-writer or a graphic package.

All three subpackages are linked by communication region, that contains all data packed in an efficient way. From the user point of view, HYBRID 3.03 is still one package that may be easily used for different purposes chosen via specification file.

The chosen method of allocating storage in the memory takes maximal advantage of the available computer memory and of the features of typical real-world problems. In general, the matrix of constraints is large and sparse, while the number of all essential, non-zero coefficients that take different numerical values is much smaller than the number of all non-zero coefficients. A super-sparse-matrix technique is therefore applied to store the data that define the problem to be solved. This involves the construction of a table of these essential coefficients. In addition, all indices and logical variables are packed so that one four-byte word is being used for four indices (2 logical and 2 integer). All data is packed in blank common to minimize the storage area used.

Special commands of HYBRID support model verification and problem modification. This is necessary to facilitate scenario analysis and to reduce the problems caused by inappropriate scaling (cf sec. 4.6.).

The data format for the input of MPS file and the output of LP results follows standards adopted by most commercial mathematical programming systems (cf e.g. [24]).

1.2.6. Outline of the solution technique

HYBRID uses a particular implementation of the Lagrange multiplier method for solving linear programming problems. General linear constraints are included within an augmented Lagrangian function. The LP problem is solved by minimizing a sequence of quadratic functions subject to simple constraints (lower and upper bounds). This minimization is achieved by the use of a method which combines the conjugate gradient method and an active constraints strategy.

In recent years many methods oriented for solving dynamic linear problems (DLP) have been developed. Most of those methods consists of adaptation of the simplex method for problems with a special structure of constraints. In HYBRID, a different approach is applied. A DLP, which should be defined together with a state equation, is solved through the use of adjoint equations and by reduction of gradients to control subspaces (more exactly, to a subspace of independent variables). The method exploits the sparseness of the matrix structure. The simple constraints (lower and upper bounds for non–slack variables) for control variables are not violated during optimization and the resulting sequence of multipliers is feasible for the dual problem. The global constraints (i.e constraints other then those defined as simple constraints) may be violated, however, and therefore the algorithm can be started from any point that satisfies the simple constraints.

The solution technique can be also used to solve single-criteria quadratic problems with virtually no changes in the algorithm. However, a routine to input and handle the relevant data and a corresponding standard for data input have yet to be designed and implemented. The solution method for multi-criteria quadratic problems requires modification of the algorithm. However the necessary modifications will be based on HYBRID 3.03.

In order to provide general information about capabilities of HYBRID, the main options are listed below. HYBRID offers the following features:

- Input of data and the formulation of an LP problem follow the MPS standard. Additional rules (that concern only sequencing of some rows and columns) should be observed in order to take advantage of the structure of a dynamic problem. An experienced user may speed up computations by setting certain options and/or parameters (cf the HYBRID User Manual).
- Solution is available in the standard MPS format and optionally in a user file which contains all data that might be useful for postoptimal analysis and reports.
- A main storage area, called the *communication region*, contains all the information corresponding to a run. The communication region is stored on disk in certain situations to allow continuation of computations from failed (or interrupted) runs or to run a modified problem while using previously obtained information without the necessity of reading and processing the MPS input file.
- The multicriteria problem is formulated and solved as a sequence of parametric optimization problems modified in interactive way upon analysis of previous results.
- For static or dynamic problem, the solution technique can be chosen.
- The problem can be modified at any stage of its solution (i.e., by changing the matrix of coefficients, introducing or altering right-hand sides, ranges or bounds).
- A special problem scaling is implemented (as described by the authors in [4] and briefly discussed in Section 3.8).
- A comprehensive diagnostics is implemented, including the checking of parallel rows, the detection of columns and rows which are empty or contain only one entry, the splitting of columns, the recognition of inconsistencies in right-hand sides, ranges and bounds, and various other features that are useful in debugging the problem formulation.
- The package supports a display of a matrix by rows (printing the nonzero elements and names of the corresponding columns, right-hand sides and ranges), as well as a display of a matrix by columns (analogous to displaying by rows).

- A check of the feasibility of a problem prior to its optimization is optionally performed.
- The optimization problem solver uses a regularization of the problem (see Section 3.7).
- More detailed information for an infeasible or unbounded problem is optionally provided by the package.

1.3. Remarks on implementation

HYBRID 3.03 is an extended version of HYBRID 2.1 documented in [27]. Therefore there are only small changes in the methodological guide in comparison to the methodology presented in [27], because the solution techniques are basically the same. However, there are some important methodological innovations:

- A modification of the problem formulation and of the solution technique as well as resulting changes in the algorithm allow for solving dynamic problems with delays in both control and state variables.
- Instead of state equations for a dynamic problem, the user may specify state inequalities.
- The optimization algorithm has been improved by an automatic evaluation of some parameters, a different technical implementation of scaling, some changes in control flow, which results in its faster execution.
- The code has been modified in a way that allows for implementation on a personal computer (compatible with IBM PC). A new approach to data handling provides for easier use of the package.
- Diagnostics have been improved and several observed bugs have been removed.

Only small changes has been introduced the solution technique applied to HYBRID version 3.01 (described in [28]), therefore a user who is familiar with the Methodological Guide [28] may skip section 2 through 5. However, old user guide [27] should no longer be used.

2. STATEMENT OF OPTIMIZATION PROBLEMS

2.1. Formulation of an LP problem

We will consider a linear programming problem (P) in the following standard form (see, e.g., [9]):

$$\min \ cx \tag{2.1}$$

$$b - r \leq Ax \leq b \tag{2.2}$$

$$l \leq x \leq u \tag{2.3}$$

where $x,c,l,u \in R^n$, $b,r \in R^m$ and A is an $m \times n$ matrix.

The constraints are divided into two groups: general constraints (2.2) and simple constraints (2.3). In the input data file (MPS file) the vectors b is called RHS and the vector r – RANGES. The vector l and u are called LOWER and UPPER BOUNDS, respectively. Obviously, some of bounds and/or ranges may have an infinite value. Therefore HYBRID may be used for solving any LP problem formulated in the way accepted by most of commercial packages.

2.2. Classical formulation of a Dynamic LP problem (CDLP)

Before discussing a formulation of a dynamic problem that can be solved by HYBRID 3.03., let us first consider a classical formulation of a dynamic linear programming problem (CDLP) (cf [17]) in the following form:

Find a control trajectory

$$u = (u_1,...,u_T)$$

and a state trajectory

$$x = (x_1,...,x_T)$$

satisfying the state equations with initial condition x_0

$$x_t = A_{t-1}x_{t-1} + B_t u_t - c_t \tag{2.4}$$

and constraints

$$d_{t-1} - r_{t-1} \leq F_{t-1}x_{t-1} + D_t u_t \leq d_{t-1} \quad t=1,...,T \tag{2.5}$$

$$e_t \leq u_t \leq f_t \quad t=1,...,T \tag{2.6}$$

$$F_T x_T \leq d_T \tag{2.7}$$

which minimize the performance index

$$\sum_{t=1}^{T} (a_t x_t + b_t u_t) \tag{2.8}$$

where:

- t=1,...T denote periods of time
- state variables x_t, control variables u_t, both for each period, are elements of Euclidian spaces of appropriate dimensions;

- matrices A_t,B_t,D_t,F_t are assumed to be given,
- RHS vectors c_t and d_t, as well as range vector r_t and bounds for control variables e_t and f_t are given,
- initial condition x_0 is given.

The above given formulation has been chosen for the purpose of simplification of presentation only. Actually, the following modifications are accepted:

1. Instead of inequality (2.5), equality constraints can be used;
2. Since no constraints of bounds type (2.6) are allowed for state variables x, such constraints may be specified in columns section of MPS file, thus formally are handled as inequality constraints of type (2.5);
3. Performance index (goal function) can either be specified as single objective or will be replaced by a dummy goal function that is defined by the transformation of a multicriteria problem to a parametric LP problem;

The structure of an CDLP problem (formulated above as in [17]) may be illustrated by the following diagram (example for T = 3, u_1,u_2,u_3,x_0,x_1,x_2,x_3 are vectors, slack variables are not shown)

u_1	u_2	u_3	x_0	x_1	x_2	x_3	rhs	var.
B_1	0	0	A_0	$-I$	0	0	c_1	state eq.
0	B_2	0	0	A_1	$-I$	0	c_2	state eq.
0	0	B_3	0	0	A_2	$-I$	c_3	state eq.
D_1	0	0	F_0	0	0	0	d_0	constr.
0	D_2	0	0	F_1	0	0	d_1	constr.
0	0	D_3	0	0	F_2	0	d_2	constr.
0	0	0	0	0	0	F_3	d_3	final state
b_1	b_2	b_3	0	a_1	a_2	a_3	—	goal

where I is identity matrix and 0 is a matrix composed of zero elements

2.3. Formulation of a Dynamic Problem (DLP)

The formulation of CDLP has been chosen for the purpose of simplification of presentation only. Actually HYBRID 3.03 is capable to solve problems of more general class, which will be referred to as Dynamic Linear Programming problems (DLP). Namely, the matrices $B = diag(B_i)$, $D = diag(D_i)$, $F = diag(F_i)$ need no longer be block diagonal matrices. Also matrices below identity matrices need no longer have any specific structure. Therefore the CDLP is a specific example of DLP. One of main generalizations - from a practical point of view - is that a problem with delays for control variables (which is not CDLP–class problem) may be solved by HYBRID. In fact, HYBRID accepts also problems with delays for both state and control variables, provided that state variables for periods "before" initial state do not enter state equations. A choice of criteria for CDLP–class problem is also limited in comparison with that for DLP (cf sec.4.3).

All variables are divided into two groups: decision variables u and state variables x_t, the latter are specified for each period of time.

A single criteria DLP problem may be formulated as follows:

Find a trajectory x_t and decision variables u such that both:

state equations:

$$-H_t x_t + \sum_{i=0}^{t-1} A_{t-1,i} x_i + B_t u = c_t, \qquad t=1,...,T \tag{2.9}$$

with given initial condition x_0

and constraints:

$$d - r \leq \sum_{t=0}^{T} F_t x_t + Du \leq d \tag{2.10}$$

$$e \leq u \leq f \tag{2.11}$$

are satisfied and the following function is minimized:

$$\sum_{t=1}^{T} a_t x_t + bu \tag{2.12}$$

Components of vector u are called decision variables for historical reasons. Actually a vector u may be composed of any variables, some of them may be specified for each time period and enter criteria defined for a dynamic case. But some components of vector u may not be specified for any time period (cf sec. 7.3.1). An example of such variable is "..dummy.", a variable generated by HYBRID for a multicriteria problem. A user may also specify variables independent of time. For the sake of keeping the formulation of the problem as simple as possible we have not introduced a separate name for such variables.

The following two symbols can be used in the specification file for definition of DLP:

NT - number of periods (stands for T in the above formulation)

NSTV - number of state variables in each period (the dimension of vectors x_t)

The user can define state inequalities instead of state equations (2.9). The slack variables for such inequalities are generated by HYBRID. Therefore, for the sake of the presentation simplicity, only the state equation will be considered further on.

The structure of an DLP problem may be illustrated by the following diagram: (corresponding to an example analogous to the above example for CDLP)

u	x_0	x_1	x_2	x_3	rhs	var.
B_1	A_{00}	$-H_1$	0	0	c_1	state eq.
B_2	A_{10}	A_{11}	$-H_2$	0	c_2	state eq.
B_3	A_{20}	A_{21}	A_{22}	$-H_3$	c_3	state eq.
D	F_0	F_1	F_2	F_3	d	constr.
b	0	a_1	a_2	a_3	-	goal

where H_t is diagonal matrix and 0 is a matrix composed of zero elements

2.4. Multicriteria optimization

2.4.1. General remarks

The specification of a single-objective function, which adequately reflects preferences of a model user is perhaps the major unresolved difficulty in solving many practical problems as a relevant optimization problem. This issue is even more difficult in the case of collective decision making. Multiobjective optimization approaches make this problem less difficult, particularly if they allow for an interactive redefinition of the problem.

The method adopted in HYBRID 3.03 is the reference point approach introduced by Wierzbicki [21]. Since the method has been described in a series of papers and reports and has been applied to DIDAS (cf [1],[20]), we give only general outline of the approach applied. This approach may be summarized in form of following stages:

1. The user of the model (referred to further as the decision maker – DM) specifies a number of criteria (objectives). For static LP problem a criterion is a linear combination of variables. For DLP problems one may also use other types of criteria (cf sec. 2.4.2). The definition of criteria in HYBRID can be performed in an easy way described in Section 7.3.2.
2. The DM specifies an aspiration level $\bar{q} = \{\bar{q}_1,....,\bar{q}_{NC}\}$, where $\bar{q}_i$ are desired values for each criterion and NC is a number of criteria. Aspiration level is called also a reference point.
3. The problem is transformed into an auxiliary parametric LP (or DLP) problem. Its solution gives a Pareto-optimal point. If specified aspiration level $\bar{q}$ is not attainable, then the Pareto-optimal point is the nearest (in the sense of a Chebyshev weighted norm) to the aspiration level. If the aspiration level is attainable, then the Pareto-optimal point is uniformly better then $\bar{q}$. Properties of the Pareto-optimal point depend on the localization of the reference point (aspiration level) and on weights associated with criteria.
4. The DM explores various Pareto-optimal points by changing either the aspiration level $\bar{q}$ or/and weights attached to criteria or/and other parameters related to the definition of the multicriteria problem.
5. The procedure described in points 3 and 4 is repeated until satisfactory solution is found.

To give more formal presentation, let us introduce following notation:

NC is the number of criteria

q_i is the i-th criterion

$\bar{q}_i$ is the aspiration level for i-th criterion

w_i is a weight associated with i-th criterion (whereas the user specifies its absolute value which is internally changed to negative depending on the type of criteria - cf sec. 2.4.3).

ϵ_m is a given non-negative parameter.

A Pareto-optimal solution can be found by the minimization of the achievement scalarizing function in the form

$$\max_{i=1,...,NC}\{w_i\,(q_i-\bar{q}_i)\} + \epsilon_m \sum_{i=1}^{NC} w_i\ q_i \ \rightarrow\ \min$$

This form of achievement function is a slight modification of a form suggested by A.Lewandowski [20] and by A.Wierzbicki [23]. Note that for ϵ_m=0 only weakly Pareto-optimal points can be guaranteed as minimal points of this function. Therefore, the use of very small ϵ_m will result in practice (except of situations in which reference point has some specific properties) in almost weakly Pareto-optimal solution. On the other hand, too big values of ϵ_m could drastically change properties associated with the first part of the scalarizing function.

2.4.2. Types of criteria

A user may define any number of criteria. To facilitate the definition 6 types of criteria are available and a user is requested to declare chosen types of criteria before their actual definition. Two types of criteria are simple linear combination of variables and those criteria may be used for both static and dynamic problems. Four other types of criteria correspond to various possible performance indices often used for dynamic problems. Since the latter criteria implicitly relate to the dynamic nature of the problem, they may be used only for variables that are defined for each time period. The only exception is the type DER of criteria, which may be defined by state variables only.

For the sake of simplicity, only the variables of the type x_i (which otherwise is used in this paper to distinguish a state variable in DLP) are used in the following formulae. Note that $x_i = \{x_{it}\}$, t=1,... T .

An k-th criterion q_k is defined in one of following ways, for static and dynamic LP:

Type MIN

$$q_k = \sum_{t=1}^{T} \sum_{i=1}^{n} a_{it} x_{it} \rightarrow \min$$

where n is number of (state and control) variables, T is number of periods; T=1 is assumed for static LP.

Type MAX

$$q_k = \sum_{t=1}^{T} \sum_{i=1}^{n} a_{it} x_{it} \rightarrow \max$$

and exclusively for dynamic LP:

Type SUP

$$q_k = \max_{t=1,..T} (x_{it} - \bar{x}_{it}) \rightarrow \min$$

where x_i is a selected state or control variable, $\bar{x}_i$ - its reference trajectory

Type INF

$$q_k = \min_{t=1,..T} (x_{it} - \bar{x}_{it}) \rightarrow \max$$

Type FOL

$$q_k = \max_{t=1,..T}(abs(x_{it} - \bar{x}_{it})) \rightarrow \min$$

Type DER

$$q_k = \max_{t=1,..T}(abs(x_{it} - x_{it-1})) \rightarrow \min$$

which applies only to state variables.

2.4.3. Transformation of multicriteria problem to an auxiliary LP

The transformation is done by HYBRID 3.03, therefore its description here has only informative purpose. This description may be useful in case of using the MPS file (optionally created after modifications and transformation of a problem) as input for another LP package.

Following notation is used throughout this subsection:

v - name of the auxiliary variable v

w_i - optional weight coefficient for i-th criterion (default value equal to 1.),

cn_i - name of i-th criterion,

ch_t - string (2-characters) which identifies t-th period of time,

$\bar{q}_i$ - reference point (aspiration level) for i-th criterion,

q_i - linear combination of variables that defines a criterion of the type MAX or MIN,

' ' - delimiters of a string,

T - number of time periods,

$x_j = \{x_{jt}\}, t=1,...,T$ is a variable that enters a criterion of a type SUP,INF,FOL or DER.

Transformation will be discussed for each type of criteria:

Type : **MIN**

additional row (with name which is concatenation of following three strings: '<', cn_i, '· · ·' is generated in form:

$$-v + w_i q_i \leq w_i \bar{q}_i$$

Type : **MAX**

is transformed in the way similar to type MIN, with additional (internal, for computations only) change of the signs of w_i to negative.

Type : **SUP**

additional T rows (with names which are concatenations of strings '<', cn_i, '.' ch_t, where $t=1,...,T$) are generated in forms:

$$-v + w_i x_{jt} \leq w_i \bar{x}_{jt} + w_i \bar{q}_i$$

Type : **INF**

is transformed in the way similar to type SUP, with additional (internal, for computations only) change of the signs of w_i to negative.

Type : **FOL**

- additional T columns (with names which are concatenations of strings '+', cn_i, '.', ch_t, where t=1,...,T) are generated ; in the following formulae this name is replaced by c_{it}^{+}
- additional T columns (with names which are concatenations of strings '−', cn_i, '.', ch_t, where t=1,...,T) are generated ; in the following formulae this name is replaced by c_{it}^{-}
- additional T rows (with names which are concatenation of strings '=', cn_i .', ch_t, where t=1,...,T) are generated in form :

$$c_{it}^{+} - c_{it}^{-} - x_{jt} = - \bar{x}_{jt}$$

- additional T rows (with names which are concatenations of strings '<', cn_i, '.', ch_t, where t=1,...,T) are generated in the form:

$$-v + w_i(c_{it}^{+} + c_{it}^{-}) \leq w_i \bar{q}_i$$

Type : **DER**

- additional $2 \times T$ columns are generated in the same way as described for a criterion of the type FOL;
- additional T rows (with names with are concatenations of strings '=', cn_i, '.', ch_t, where t=1,...,T) are generated in form :

$$c_{it}^{+} - c_{it}^{-} - x_{j,t} + x_{j,t-1} = 0.$$

- additional T rows (with names which are concatenations of strings '<', cn_i, '.', ch_t) are generated in form :

$$-v + w_i(c_{it}^{+} + c_{it}^{-}) \leq w_i \bar{q}_i$$

Auxiliary goal function, which is to be minimized, is generated in the following form:

$$v + \epsilon_m \left(\sum_i w_i q_i + \sum_t \left(\sum_j w_j x_{jt} + \sum_k w_k (c_{kt}^{+} + c_{kt}^{-})\right)\right)$$

where summation is done over corresponding sets of respective criteria, i.e. indices i, j, k correspond to criteria of type: MIN or MAX, SUP or INF and FOL or DER, respectively; ϵ_m is given parameter.

The name of auxiliary variable v is '..dummy.', whereas the name of auxiliary goal function is '.dummy..'.

Value of ϵ_m may be changed by the command MEPS in a routine for modification of multicriteria parameters.

3. THEORETICAL FOUNDATIONS AND METHODOLOGICAL PROBLEMS

3.1. General remarks

The most popular methods for solving linear programming problems are based on the simplex algorithm. However, a number of other iterative non–simplex approaches have recently been developed [5–7]. HYBRID belongs to this group of non–simplex methods. The solution technique is based on the minimization of an augmented Lagrangian penalty function using a modification of the conjugate gradient method. The Lagrange multipliers are updated using a modified version of the multiplier method [8] (see Sections 3.2 and 3.4).

This method is useful not only for linear programming problems but also for other purposes, as described in Section 1.2. In addition, the method may be used to solve problems with non-unique solutions (as a result of regularization – see Section 3.7).

The following notation will be used:

a_i denotes the i-th row of matrix A

x_j denotes the j-th component of vector x

$\|x\|$ denotes the Euclidian norm of vector x

$(u)_+$ denotes the vector composed of the non-negative elements of vector u (where negative elements are replaced by zeros)

A^T denotes transposition of matrix A

3.2. The multiplier method

We shall first explain how the multiplier method may be applied directly to LP problems.

Consider the problem (PO), which is equivalent to the problem (P) defined in Section 2.1:

$$\begin{aligned} &\min \ cx \\ &Bx \leq d \end{aligned} \qquad \text{(PO)}$$

where $d \in R^p$, B is a $p \times n$ matrix, and $m \leq p \leq 2(m+n)$. To apply the multiplier method to this problem we proceed as follows:

Select initial multipliers y^0 (e.g., $y^0 = 0$) and $\rho \in R$, $\rho > 0$. Then for $k = 0,1,...,$ determine successive values of x^{k+1}, y^{k+1} where

$$x^{k+1} = \underset{x}{argmin}\ L(x,y^k)$$

and

$$y^{k+1} = (y^k + \rho(Bx_{k+1}-d))_+ \qquad (3.1)$$

where

$$L(x,y^k) = cx + (\|(y^k + \rho(Bx-d))_+\|^2 - \|y^k\|^2)/(2\rho) \qquad (3.2)$$

until a stopping criterion is satisfied.

The method has the following basic properties:

1. A piecewise quadratic differentiable convex function is minimized at each iteration.
2. The algorithm terminates in a finite number of iterations for any positive ρ.
3. There exists a constant $\bar{\rho}$ such that for any $\rho \geq \bar{\rho}$ the algorithm terminates in the second iteration.

Note that it is assumed above that the function $L(\cdot,y^k)$ is minimized exactly and that the value of the penalty parameter ρ is fixed. Less accurate minimization may be performed provided that certain conditions are fulfilled (see, e.g., [7,8]). For numerical reasons, a non-decreasing sequence of penalty parameters $\{\rho^k\}$ is generally used instead of a fixed ρ.

3.3. The conjugate gradient method for the minimization of an augmented Lagrangian penalty function

The augmented Lagrangian function for a given vector of multipliers y will be called the augmented Lagrangian penalty function [22]. For minimization of that function the conjugate gradient method has been modified to take advantage of the formulation of the problem. The method may be understood as an modification of the techniques developed by Polyak [10], O'Leary [11] and Hestenes [12] for minimization of a quadratic function on an interval using the conjugate gradient method.

The problem (P) may be reformulated as follows:

$$\begin{aligned} &\min \ cx \\ &Ax + z = b \\ &l \leq x \leq u \\ &0 \leq z \leq r \end{aligned} \tag{PS}$$

where $z \in R^m$ are slack variables.

Formulation (PS) has a number of advantages over the initial formulation (PO):

1. The dimension of matrix A in (PS) is usually much smaller than that of matrix B in (PO).
2. The problem is one of minimization of a quadratic function in (PS), and of minimization of a piecewise quadratic in (PO).
3. Some computations only have to be performed for subsets of variables. Note that slack variables are introduced only for ease of interpretation and do not have to be computed.

In (PS) the augmented Lagrangian is defined by

$$L(x,z,y) = cx + (\| y + \rho(Ax+z-b) \|^2 - \| y \|^2)/(2\rho) \ . \tag{3.3}$$

We shall first discuss the problem of minimizing $L(x,z,y)$ for given $y,\rho > 0$, subject to lower and upper bounds for x and z. Let us consider the following augmented Lagrangian penalty function

$$F(x,z) = (c/\rho)x + (\| y/\rho + Ax - b + z \|^2 - \| y/\rho \|^2)/2. \tag{3.4}$$

The gradient of F is defined by

$$\frac{\partial F}{\partial x} = c/\rho + A^T(z-g)$$

$$\frac{\partial F}{\partial z} = z - g$$

where

$$g = -y/\rho - Ax + b \ .$$

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

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

$$\frac{\partial F^*}{\partial z_i} \geq 0 \ \text{ if } \ z_i^* = 0 \ , \ \frac{\partial F^*}{\partial z_i} \leq 0 \ \text{ if } \ z_i^* = r_i \ ,$$

and

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

$$\frac{\partial F^*}{\partial z_i} = 0 \ \text{ if } \ 0 < z_i^* < r_i \ .$$

For any given point such that $l \leq x \leq u$ it is possible to determine slack variables $0 \leq z \leq r$ in such a way that the optimality conditions with respect to z are obeyed. Variables z are defined by

$$z_i = \begin{cases} 0 & \text{if } \ g_i \leq 0 \quad (\partial F/\partial z_i > 0) \\ r_i & \text{if } \ g_i \geq r_i \quad (\partial F/\partial z_i < 0) \\ g_i \ \text{ if } \ r_i > g_i > 0 & (\partial F/\partial z_i = 0) \ . \end{cases} \tag{3.5}$$

We shall use the following notation and definitions. The vector of variables x with indices that belong to a set J will be denoted by x^J, and analogous notation will be used for variables g. We shall let q denote minus the gradient of the Lagrangian penalty function reduced to x-space ($q = -(\partial F/\partial x)$). The following sets of indices are defined for a given point x:

The set of indices I of violated constraints, i.e.,

$$I = \{i : g_i \geq r_i\} \bigcup \{i : g_i \leq 0\} \ .$$

$\bar{I}$ is the complement of I, i.e.,

$$\bar{I} = \{1,2,....,m\} \backslash I \ .$$

The set of indices I can be also interpreted as a set of active simple constraints for z. The set of indices J of variables that should be equal to either the upper or the lower bound, i.e.,

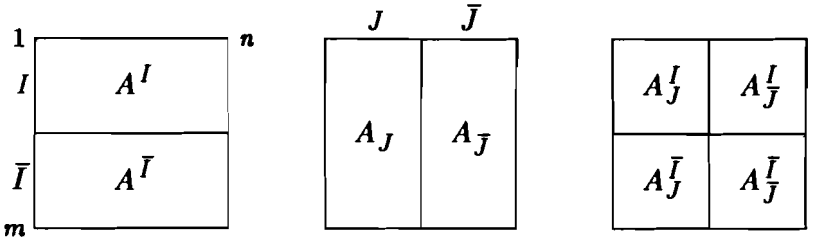
$$J = \{j : x_j = l_j \ \text{ and } \ q_j \leq 0\} \bigcup \{j : x_j = u_j \ \text{ and } \ q_j \geq 0\} \ .$$

$\bar{J}$ is the complement of J, i.e.,

$$\bar{J} = \{1,2,.....,n\} \backslash J \ .$$

For the sake of illustration the matrix A may be schematically split up in the following three ways (see the Figure below): first according to active rows, second according to basic columns and third with illustrate the part of the matrix A for which augmented Lagrangian penalty function is computed. The contents of the matrix $A_{\bar{J}}^{I}$ (for which the

augmented Lagrangian penalty function is computed) changes along with computations.



In essence, the augmented Lagrangian penalty function is minimized using the conjugate gradient method with the following modifications:

1. During the minimization process x and z satisfy simple constraints and z enters the augmented Lagrangian in the form defined by (3.5).
2. The conjugate gradient routine is run until no new constraint becomes active, i.e., neither set I nor set J increases in size. If this occurs, the computed step length is shortened to reach the next constraint, the corresponding set (I or J) is enlarged and the conjugate gradient routine is re-entered with the direction set equal to minus the gradient.
3. Sets J and I are defined before entering the procedure discussed in point 2 and may be only enlarged before the minimum is found. When the minimum with respect to the variables with indices in sets $\bar{J}$ and I has been found, sets J and I are redefined.
4. Minimization is performed subject only to those components of variables x whose indices belong to set $\bar{J}$, i.e., variables that are not currently equal to a bound value.
5. Minimization is performed subject only to those components of variables z whose indices do not belong to set I, i.e., slack variables that correspond to non-active simple constraints for z. Note that, formally, this requires only the use of different formulae for z. In actual fact it is sufficient to know only the set I, which defines the minimized quadratic function.

4. SOLUTION TECHNIQUE

4.1. Algorithm for minimization of augmented Lagrangian

We may now present the algorithm for minimization of the augmented Lagrangian penalty function in a more formal way. The algorithm consists of the following steps:

1. For given y and $\rho>0$ choose a point x such that $l \leq x \leq u$
2. Compute $g = -y/\rho \ -Ax + b$
3. Determine sets I and $\bar{I}$

$$I = \{i: g_i > r_i\} \bigcup \{i: g_i < 0\},$$
$$\bar{I} = \{1,...,m\} \backslash I$$

4. Define $\bar{g}$ as follows:

$$\bar{g}_i = \begin{cases} g_i - r_i & \text{if } g_i - r_i > 0 \\ g_i & \textit{otherwise} \end{cases}$$

5. Compute the minus gradient:

$$q = -c/\rho + (A^I)^T \bar{g}^I$$

6. Determine sets J and $\bar{J}$

$$J = \{j: x_j = l_j \text{ and } q_j \leq 0\} \bigcup \{j: x_j = u_j \text{ and } q_j \geq 0\}$$
$$\bar{J} = \{1,...,n\} \backslash J$$

7. If $q_j = 0$ for all $j \in \bar{J}$ then x is a minimum point of the augmented Lagrangian penalty function
8. Set $p^{\bar{J}} = q^{\bar{J}}$
9. Compute

$$s = A_{\bar{J}} p^{\bar{J}}$$
$$h = \| q^{\bar{J}} \|^2$$
$$d = \| s^I \|^2$$
$$\alpha(1) = h/d$$

Note that $\alpha(1)$ is the conjugate gradient step length in direction $p^{\bar{J}}$

10. Find the step length that would violate the nearest non-active constraint, i.e., for $i \in \bar{I}$,

$$\alpha(2) = \min_{i \in K} \{g_i/s_i\}, \quad K = \{i : i \in \bar{I}, s_i > 0\}$$
$$\alpha(3) = \min_{i \in K} \{(g_i - r_i)/s_i\}, \quad K = \{i : i \in \bar{I}, s_i < 0\}$$

11. Find the step length that would enable a variable to reach a bound, i.e.,

$$\alpha(4) = \min_{j \in K} (l_j - x_j)/p_j, \quad K = \{j : j \in \bar{J}, p_j < 0\}$$

$$\alpha(5) = \min_{j \in K} (u_j - x_j)/p_j \ , \quad K = \{j : j \in \bar{J},\ p_j > 0\}$$

12. Determine step length $\alpha = \min_{i=1,\ldots,5} (\alpha(i))$. If $\alpha = min(\alpha(2), \alpha(3))$ add the row index for which this condition holds to set I and remove that index from set $\bar{I}$. If $\alpha = min(\alpha(4), \alpha(5))$ add the column index for which this condition holds to set J and remove that index from set $\bar{J}$.
13. Compute the new point $x^{\bar{J}} := x^{\bar{J}} + \alpha p^{\bar{J}}$ and the minus gradient at that point:

$$g_i := g_i - \alpha s_i$$
$$q^{\bar{J}} = (A^{I}_{\bar{J}})^T \bar{g}^I - c^{\bar{J}}/\rho$$

14. If $q^{\bar{J}} = 0$. go to step 2
15. If $\alpha = \alpha(1)$ continue with the conjugate gradient step, i.e.

$$\beta = \| q^{\bar{J}} \|^2 / h$$
$$p^{\bar{J}} := q^{\bar{J}} + \beta p^{\bar{J}}$$

and go to step 9

16. Go to step 8

Note that the condition $q^{\bar{J}} = 0$ is in practice replaced by $\| q^{\bar{J}} \| \leq \epsilon$, where ϵ is a gradient tolerance.

4.2 Steps of the multiplier method

Let the violation of i-th constraint in a point x^k be defined in the following way:

$$v_i^k = \max\{a_i x^k - b_i, -a_i x^k + b_i - r_i, 0\}$$

and $\| v^k \|_\infty$ denotes the l_∞ norm of violated constraints. The multiplier method will be presented in algorithmic form.

1. Compute an initial vector of multipliers on the basis of the particular option chosen (i.e., either $y^0 = 0$ or y^0 corresponding to the constraints violated at starting point x)
2. Find x^{k+1} which minimizes the augmented Lagrangian penalty function (see Section 3.3.) with accuracy ϵ^k. It is assumed that

$$\epsilon^k := \min(\epsilon^k, \| v^k \|_\infty \epsilon^k))$$

where the sequence $\epsilon^k \to 0$. In addition, $\epsilon_{mi} \geq \epsilon^k \geq \epsilon_{mx}$, where ϵ_{mi}, ϵ_{mx} is the assumed minimum and maximum accuracy, respectively.

3. Compute new multipliers

$$y_i^{k+1} := \begin{cases} y_i^k + \rho^k (a_i x^{k+1} - b_i) & \text{if } y^k + \rho^k (a_i x^{k+1} - b_i) \geq 0 \\ y_i^k + \rho^k (a_i x^{k+1} - b_i + r_i) & \text{if } y^k + \rho^k (a_i x^{k+1} - b_i + r_i) \leq 0 \\ 0 & \textit{otherwise} \end{cases}$$

4. If $\| y^{k+1} - y^k \| > \epsilon_d$ then set $\rho^{k+1} = \min(\rho^k \rho_s, \rho_{mx})$, $\rho_s > 1.$, ρ_{mx} is a given maximal value of the penalty parameter.

Set $\epsilon^{k+1}=\epsilon^k\epsilon_s$, where $\epsilon_s < 1.$ is an assumed parameter,
Set $k := k + 1$ and go to step 2

5. Set $k := k+1$ and find x^{k+1} which minimizes the augmented Lagrangian. If x^{k+1} is feasible ($\|v^k\| \leq FEAS$) then assume it as a solution and stop.
Otherwise set $\rho^{k+1} = \min(\rho^k\rho_s, \rho_{mx})$, and $\epsilon^{k+1}=\epsilon^k\epsilon_s$ and go to step 2.

The list of parameters which are referred to in the Section 6.5. and their relative symbols used above is as follows (index k is omitted):

RO - ρ, ROST - ρ_s , ROMX - ρ_{mx} , EPS - ϵ, EPSS - ϵ_s, EPSM - ϵ_{mx}, EPSD - ϵ_d.

4.3. Solution technique for DLP

We will not repeat reasoning given in the first part of sec. 2.3. Instead, let us point out basic differences between the algorithms for static LP and DLP:

1. Minimization is reduced to a subspace of decision variables. Gradient of Lagrangian penalty function is computed for variables that belong to a subspace of decision variables. This (together with arguments already presented in sec. 3.3.) shows advantages due to the use of dynamic structure of DLP problem in comparison with presentation of such a problem as a large LP.

2. The structure of matrices $B_1,.....,B_T$ and $F_0,...,F_T$ has no impact for the algorithm nor affects the technique of storage of data, because super-sparse technique is applied (cf sec.1.4.). It should be also pointed out that the method of transforming a multicriteria problem to a parametric LP one introduces constraints (cf sec.2.4.3.) that - for the proposed (cf sec.2.4.2.) types of criteria - do not fit to the staircase structure of CDLP (cf [17]). Therefore, any technique that would exploit the staircase structure of DLP would also imply a reduction of a number of criteria types. The alternative is then to treat a problem as a large LP static one or to apply a technique that does not exploit the classical DLP structure.

3. State equations are solved (for given decision variables u) by forward substitution. Therefore any single constraints for state variables have to be treated as general constraints and included into the matrix. Gradient need not to be computed for those variables, but state equation is solved twice (for state variables and variations).

4. A conjugate trajectory Ψ is computed from conjugate equation by backward substitution and has an interpretation of dual variables for state equations. No other variables associated with those rows (defined in sec. 3.3, i.e. Lagrange multipliers, shifted constraints g) are computed for state equations rows.

5. The general structure of the algorithm for DLP is similar to that presented in sec. 3.4. To sum up basic differences one may observe that:
   - we consider a problem that is equivalent to a static LP but reduced to the subspace of decision variables and is solved in the way similar to that described in sec. 3.3. and 3.4,
   - state equations are solved for control variables and for variations,
   - a conjugate trajectory Ψ is computed.

4.4. Algorithm for minimization of augmented Lagrangian for DLP

Now we may present the algorithm for minimization of the augmented Lagrangian function for DLP in a more formal way. In each iteration of multiplier method, the following optimization problem is solved: minimize the augmented Lagrangian penalty function

$$F(x,u,z) = \sum_{t=1}^{T} (a_t/\rho)x_t + (b/\rho)u +$$
$$+ (\|y/\rho + \sum_{t=0}^{T} F_t x_t + Du - d + z\|^2 - \|y/\rho\|^2)/2.$$

subject to

$$-H_t x_t + \sum_{i=0}^{t-1} A_{t-1,i} x_i + B_t u = c_t \qquad t=1,\ldots T$$

with a given initial condition x_0 and

$$e \le u \le f$$
$$0 \le z \le r$$

where z is a vector of slack variables, which - as discussed in sec. 3.3. - are not used in the algorithm. The algorithm consists of the following steps:

1. For given y and ρ choose a point u such that $e \le u \le f$
2. Solve the state equation

   $$H_t x_t = \sum_{i=0}^{t-1} A_{t-1,i} x_i + B_t u - c_t \quad t=1,\ldots,T$$

   with given initial condition x_0
3. Compute shifted constraints for constraints (2.10.)

   $$g = -y/\rho - \sum_{t=0}^{T} F_t x_t - Du + d$$

   and determine sets I, $\bar{I}$

   $$I = \{i:\ g_i > r_i\} \cup \{i:\ g_i < 0\}$$

   while $\bar{I}$ is the complement of I.
4. Define $\bar{g}$ as follows :

   $$\bar{g}_i = \begin{cases} g_i - r_i & \text{if } g_i > r_i \\ g_i & \textit{otherwise} \end{cases}$$

5. Find the conjugate trajectory by solving backwards the conjugate equations

   $$H_t^T \Psi_t = \sum_{i=t}^{T-1} A_{i,t}^T \Psi_{t+1} + (F_t^I)^T \bar{g}^I - a_t/\rho \ , \quad t=T-1,\ldots,1$$

   with boundary condition

   $$\Psi_T = (F_T^I)^T \bar{g}^I - a_T/\rho$$

6. Compute the minus gradient reduced to subspace of decision variables

$$q = -b/\rho + (D^I)^T \bar{g}^I + \sum_{t=1}^{T} B_t^T \Psi_t$$

7. Determine sets J and $\bar{J}$

$$J = \{j : u_j = e_j \text{ and } q_j \leq 0\} \bigcup \{j : u_j = f_j \text{ and } q_j \geq 0\}$$

while $\bar{J}$ is the complement of J

8. If $q_j = 0$ for all $j \in \bar{J}$ then u is a minimum point of the augmented Lagrangian penalty function

9. Set $p^{\bar{J}} = q^{\bar{J}}$

10. Solve state equation in variations

$$H_t x_t = \sum_{i=0}^{t-1} A_{t-1,i}\sigma_i + B_t^{\bar{J}} p^{\bar{J}} \qquad t=1,...,T$$

with boundary condition $\sigma_0 = 0$

11. Compute

$$s = D^{\bar{J}} p^{\bar{J}} + \sum_{t=0}^{T} F_t \sigma_t$$

$$h = \| q^{\bar{J}} \|^2$$

$$v = \| s^I \|^2$$

$$\alpha(1) = h/v$$

Note that $\alpha(1)$ is the conjugate gradient step length in direction $p^{\bar{J}}$

12. Find the step length that would violate the nearest non-violated constraint, i.e.,

$$\alpha(2) = \min_{i \in K} \{g_i/s_i\}, \quad K = \{i : i \in \bar{I} \text{ and } s_i > 0\}$$

$$\alpha(3) = \min_{i \in K} \{(g_i - r_i)/s_i\}, \quad K = \{i : i \in \bar{I} \text{ and } s_i < 0\}$$

13. Find the step length that would enable a variable to reach a bound, i.e.,

$$\alpha(4) = \min_{j \in K} \{(e_j - u_j)/p_j\}, \quad K = \{j : j \in \bar{J} \text{ and } p_j < 0\}$$

$$\alpha(5) = \min_{j \in K} \{(f_j - u_j)/p_j\}, \quad K = \{j : j \in \bar{J} \text{ and } p_j > 0\}$$

14. Determine step length $\alpha = \min_{i=1,...,5} (\alpha(i))$

If $\alpha = min(\alpha(2),\alpha(3))$ add the index for which this condition holds to set I and remove that index from set $\bar{I}$. If $\alpha = \min(\alpha(4),\alpha(5))$ add the index for which this condition holds to set J and remove that index from set $\bar{J}$.

15. Compute :

$$u^{\bar{J}} := u^{\bar{J}} + \alpha p^{\bar{J}}$$

$$x_t := x_t + \alpha \sigma_t$$

$$g_i := g_i - \alpha s_i$$

16. For the new g^I solve the conjugate equation (as in step 5)
17. Compute the minus gradient :

$$q^{\bar{J}} = -b^{\bar{J}}/\rho + (D^I_{\bar{J}})^T g^I + \sum_{t=1}^{T} (B_t)^T_{\bar{J}} \Psi_t$$

18. If $q^{\bar{J}} = 0$, then go to 2
19. If $\alpha = \alpha(1)$ continue with the conjugate gradient step, i.e.

$$\beta = \| q^{\bar{J}} \|^2 / h$$
$$p^{\bar{J}} = q^{\bar{J}} + \beta p^{\bar{J}}$$

and go to step 10

20. Go to step 9

Note that the condition $q^{\bar{J}} = 0$ is in practice replaced by $\| q^{\bar{J}} \| \leq \epsilon$ The value of ϵ may be quite large in the first few iterations; it then decreases as the number of iterations increases.

4.5. Regularization

It is possible that a linear programming problem may have nonunique optimal solutions. Although this is theoretically rare, in practice many problems actually have a large set of widely varying basic solutions for which the objective values differ very little [7]. In some cases, the simplex algorithm will stop when a basic solution is recognized as optimal for a given set of tolerances. For problems with a nonunique optimum, the first optimal solution found is accepted, so that one may not even be aware of the non-uniqueness of the solution reported as optimal.

Thus we are faced with the problem of choosing an optimal (or, in most cases, to be more accurate, a suboptimal) solution that possesses certain additional properties required by the user. This problem may be overcome by applying an approach called *regularization*. Regularization (Tikhonov's type) is a means of finding the optimal solution with either minimum Euclidian norm or minimum distance from a given reference point. The second of these options has not yet been implemented; the first may be activated by a REGZERO statement in the specification file (see the Section 6.2).

The minimum norm solution is obtained by carrying out a sequence of minimizations of regularized augmented Lagrangians rather than one minimization of an "ordinary" augmented Lagrangian [16]. Thus minimization of $L(\cdot, y^k)$ in problem (PO) is replaced by

$$x^{k+1} = \underset{x}{argmin}\ L(x, y^k) + \| x \|^2 / (2\eta^k)$$

where

$$\eta^k \to \infty\ , \quad \sum_{i=1}^{\infty} (\rho^k / \eta^k))^{1/2} < \infty\ ,$$

In the computer implementation of the algorithm the following rule is assumed for η^{k+1}:

$$\eta^{k+1} = min(\eta^k \eta_s, \eta_m)$$

η^0 , η_s and η_m are given parameters.

The list of parameters which are referred to in the Section 6.2 and their relative symbols used above is as follows :

RETA - η^0, RSETA - η_s, RMETA - η_m

4.6. Scaling

It is generally agreed that the choice of an appropriate scaling of a problem being solved can be a critical issue for numerical stability. There are obviously two approaches to deal with that problem. First, suggested by Tomlin ([15]), assume that an experienced model builder, who uses sensible units may avoid unnecessarily large or small matrix elements. This is true, but requires a lot of time consuming preparations, which are reliable source of frustrating bugs. Therefore, we have followed the second approach, suggested by Curtis and Reid ([14]) for solving the scaling problem. This approach is nowadays widely accepted (e.g. the new version of MINOS has also scaling option, which has removed many problems typical for older versions of MINOS).

Our approach is discussed in details in [4], therefore only short description follows. For the sake of simplicity we consider a problem of scaling on an example of a problem in a form:

$$\begin{aligned} Ax &= b \\ d &\leq x \leq q \end{aligned} \qquad (*)$$

where $A \in R^{m \times n}$

According to Curtis and Reid (1972) matrix A is considered as well-scaled if

$$\sum_{i=1}^{m} \sum_{j \in J_i} (log(abs(a_{ij})))^2 \leq v$$

for some acceptable v. J_i are sets of indices of columns with non-zero elements in i-th row.

Therefore, instead of solving a badly conditioned problem a of type (*), one can solve an equivalent problem in form

$$\begin{aligned} &(RAC)y = Rb \\ &C^{-1}d \leq y \leq C^{-1}q \\ &x = Cy \end{aligned}$$

Here $R = diag(r_1,....,r_m)$ and $C = diag(c_1,....,c_n)$ are two diagonal matrices with positive components. In other words, an equivalent problem is formed by multiplying i-th row by r_i and j-th column by c_j.

The problem of scaling boils down to finding coefficients r_i and c_j such that

$$\sum_{i=1}^{m} \sum_{j \in J_i} (log(r_i c_j abs(a_{ij})))^2 \rightarrow \min$$

It is easy to observe that the above stated problem has no unique solution (although the optimally scaled matrix may be unique). Therefore we minimize the following performance index:

$$\sum_{i=1}^{m}\sum_{j\in J_i}(log(r_i c_j abs(a_{ij})))^2 + \beta\sum_{i\in K}(log(abs(r_i rhs_i)))^2 +$$
$$\gamma\sum_{i\in L}(log(abs(c_j bnd_i)))^2 \rightarrow \min$$

where rhs and bnd are non-zero elements of RHS and bounds, respectively, sets of indices K and L contain indices of rows with non-zero rhs and columns with non-zero bounds, respectively.

For the numerical reasons the base of logarithms is 2 and obtained coefficients are rounded to nearest integer number.

For this formulation of the scaling problem, it was possible to design a specialized algorithm based on conjugate gradient method. Since an excessive accuracy is not required, the scaling algorithm is very efficient (usually it takes less then 10 iterations regardless of dimension of a problem). Therefore the scaling option (which is the default) should not be suppressed except if special requirements apply. The values of performance indices (3.7.) and (3.8.) are displayed both before and (if active) after scaling.

Usually there is no need to change default parameters. Should a change of parameters be desired, it may be done by entering respective values in specification file (SBETA stands for β and SETA stands for γ). Two stopping criteria are used, which may be controlled by parameters SEPS and SEP1. Let v^k be a value of the performance index (3.8.). The scaling routine is ended, if $v^k/v^{k-1} \geq$ SEPS or if the norm of gradient is less then SEP1. In addition the number of scaling iterations in constrained by ITSCAL (cf Sections 6.4 and 6.5).

Scaling coefficients are displayed as additional column in MPS-type output of results. This has only informative purpose, since all results are rescaled internally.

5. TESTING EXAMPLES

HYBRID has been tested on number of examples. For the sake of illustration of the package capabilities 3 known examples have been selected: two dynamic and one static.

5.1. Economic growth model (Manne)

This model is a linear multicriteria version of Manne's model described in [26].

The variables have the following meaning:

t time period, $t = 1,2,...,T$

c_t consumption

i_t investment,

k_t capital in time period t.

The following criteria have been selected for illustration of multicriteria optimization:

$$\max \sum_{t=1}^{T} \beta_t c_t \qquad \text{(of the type MAX)}$$

$$\max k_T \qquad \text{(of the type MAX)}$$

$$\min \max_{t=1,2,...,T} |c_t - \bar{c}_t| \qquad \text{(of the type FOL)}$$

The state equations have the following form:

$$k_t = k_{t-1} + i_t, \qquad t = 1,2,...,T$$

with k_0 given.

Linear constraints are defined for $t = 1,2,...,T$

$$c_t + i_t \leq \alpha_{t-1} k_{t-1}$$

$$k_t \geq k_0 + i_0$$

Bounds are given for both control variables (for each variable a constraint is specified for each time period $t = 1,2,...,T$:

$$c_t \geq c_0$$

$$i_t \leq (1.04)^t i_0$$

The following parameters (where $\alpha = (c_0 + i_0)/k_0$) have been assumed:

$$\beta_t = 0.95^t, \quad b = 0.25, \quad g = 0.03, \quad c_0 = 0.65,$$

$$i_0 = 0.16, \quad k_0 = 3.0, \quad \alpha_t = \alpha(1+g)^{(1-b)t}$$

In the Table 1 the test examples which refers to the modified Manne problem are denoted by ManneT, where T corresponds to a number of periods.

5.2. Flood control problem.

The problem is a model (cf [25]) of the water system which consists of three general purpose reservoirs supplying water to the main river reach. The goal of the system dispatcher is to operate the reservoirs in such a way that the flood peak on the main river do not coincide. It is assumed that inflow forecast for each reservoir is known.

The model consists of water balance equations for selected points and for each time period. The capacities of reservoirs are also constraint. Various types of criteria are examined:

FOL – corresponds to following given trajectories of water flow in selected points,

DER – corresponds to minimization water flow changes (in consecutive time periods) in selected points,

MAX – corresponds to minimization of maximal (over time) flow in selected points.

In the table 1 the test examples which refers to the multicriteria flood control problems are denoted by FloodT, where T corresponds to a number of periods.

5.3. Full dense LP problem.

This problem is a modification of the Mangasarian example [5] and has been generated for verification of the package for fully dense LP problems. Computations are performed for one criterion and elements of matrix are equal to 1.0 with exception of diagonal elements for which values of 10.0 are selected.

In the table 1 the test examples which refers to the modified Mangasarian example are denoted by MangT, where T corresponds to a dimension of LP matrix.

5.4. Discussion of test results.

Testing problems have been solved on a PC compatible with IBM/AT (running at 8 MHz) with 80287 coprocessor. The algorithm was implemented with double precision arithmetic (the machine precision about 2.22e-16). The default values of all parameters (this includes initial multipliers equal to zero) were assumed in all runs.

The results of some tests are summarized in the following table.

Problem	Number of crit.	Rows	Cols	Dens. [%]	Time (min.)	Mult. iter.	Outer iter.	Total steps
Manne05	3	29	27	12	0.4	2	13	24
Manne10	3	54	52	7	0.6	2	23	28
Manne20	2	103	102	3	3.0	2	41	72
Manne30	2	153	152	2	5.0	2	64	112
Manne40	2	203	202	2	9.5	2	84	154
Flood03	6	55	55	6	5.0	10	87	230
Flood05	3	77	79	4	4.5	2	36	172
Mang20	1	20	20	100	2.0	2	4	49
Mang30	1	30	30	100	5.0	2	4	76

Numbers of rows and columns correspond to a single criterion LP problem, which were obtained by transformation of relevant multicriteria problems. The numbers of outer iterations and of total steps correspond to execution of step 2 and step 3 of the algorithm (cf sec. 4.1.).

Due to super sparse matrix technique applied for storing data, rather long computation time is required for fully dense matrix problems. For dynamic sparse problems better performance of the algorithm was observed. HYBRID is usually slower in comparison to packages which are based on the simplex method but requires less computer memory. On the other hand HYBRID performs detailed diagnostic of a problem being solved and offers a possibility of definition and modification of a multicriteria problem and its conversion to an equivalent single criterion problem.

As an illustration of HYBRID performance the modification of the Manne problem (for the sake of creating a larger problem we have introduced 10 sectors instead of one given in formulation in sec. 5.1) for 20 time periods has been solved by both MINOS ver. 5.0 (cf [29]) and HYBRID ver. 3.03. The test has been performed on VAX 780/11 under Berkeley UNIX 4.2. A multicriteria problem with criteria presented in sec. 5.1 has been generated. The multicriteria problem has been converted by HYBRID to a corresponding single criteria problem and the MPS format input file for MINOS has been generated.

The resulting problem has 464 rows, 471 columns and 1463 elements (density 0.7%). MINOS has used 2.9 min. (the sum of user and system time) to solve the above mentioned problem. HYBRID has used 4.5 min. for processing and diagnostic of the problem (which includes interactive definition of initial reference trajectory, conversion of multicriteria problem to the equivalent single criterion problem and generation of MPS format file for the latter problem) and 5.5 min. for solving the problem. On the other hand HYBRID has used less then half of computer memory required by MINOS to solve the problem.

6. CONTROL COMMANDS OF THE PACKAGE

6.1. General description of control commands

The sequence of operations executed by HYBRID is controlled by the user through commands provided in a specification file (see Section 7.2.). Some of these control commands are listed below. It is recommended that only the commands mentioned here should be redefined by the user: as yet there is insufficient information on the effects of changing the values of other parameters or options. The authors of the package hope to formulate guidelines governing the modification of these additional parameters/options in due course.

A control statement activates or deactivates a certain option, defines or redefines the logical number of an input/output unit, or sets the value of a parameter. Each statement has a default value which is initialized prior to starting the run. These default values are also given below.

The control commands are divided into five groups:

1. Commands without parameters
2. Commands with character string parameters
3. Commands with integer parameters
4. Commands with real parameters
5. Commands with a single parameter which may be either a character string or a real number

Each group of commands is discussed separately below.

6.2. Commands without parameters

Each statement of this type activates or deactivates a certain action, and therefore they are listed in pairs. The only exception is the RECOVERY statement, which - if occurs -must be specified as a first statement in a specification file. The first of each pair of listed options is the default one.

MAXIMIZE MINIMIZE	Defines type of optimization of the objective function. This option is overwritten if multicriteria optimization is performed
NOMULTI MULTI	To activate the subpackage for multicriteria optimization (definition and modification – cf sec. 7.3.2)
SCALE NOSCALE	To activate the scaling routine
NOMODIFY MODIFY	To activate the routine for problem modification (cf sec. 7.4)
NOBROWS BYROWS	To display the matrix by rows
NOBCOLS BYCOLS	To display the matrix by columns

NOMPS
MPSOUT — To output the problem being solved in MPS–format

NOACCEPT
ACCEPT — To allow minor errors in the MPS file (such as zero elements, duplicated elements, etc.). Such errors are reported but do not cause termination of the run prior to optimization if the ACCEPT option is set

NOREGUL
REGZERO — To regularize the problem (see Section 4.5)

NOGETFEAS
GETFEAS — To check feasibility prior to optimization. This action should be avoided for problems likely to have a feasible solution because its use increases the total number of iterations.

NOPARALLEL
PARALLEL — To check for parallel rows. This option is time- consuming but helps to identify dominating rows and pairs of constraints that may be replaced by a single constraint with appropriate range.

COMMULT
ZERMULT — Sets the initial value of the Lagrange multipliers to zero. The alternative is to compute the initial multipliers before the first iteration,but this requires activation of the SPIRIT routine.

NOSPIR
SPIRIT — Activates a specialized routine for computation of a starting point.

RECOVERY — Inputs the contents of the communication region. This statement causes the communication region to be read and computations starts (after possible modification of the problem) from stored point. The absence of this statement causes so called cold start. If the statement is specified, it must be the first one in a specification file.

Declaration of a dynamic problem is implicit and is done by NT parameter (cf sec. 6.4).

6.3. Commands with character string parameters

GOAL — Name of the neutral row taken as the objective function. If absent, the neutral row encountered first is assumed to be the objective function. The name is overwritten if multicriteria optimization is performed.

RHS — Name of the set of right-hand sides and ranges. If absent, the first such name encountered is taken

BOUNDS — Name of the bounds set. If absent, the first such name encountered is taken.

NAME — Name of the problem. If absent, the name found in the MPS file is assumed.

6.4. Commands with integer parameters

MROWS	100	Maximum number of rows
MCOLS	5*MROWS	Maximum number of columns
MELEM	5*MCOLS	Maximum number of nonzero matrix elements
MDIFF	MELEM	Maximum number of different quantities defining the problem
MTIME	10	Number of CPU minutes allocated for the run
MITER	-	Maximum number of iterations (cf Section 7.5)
MERRORS	50	Maximum number of errors allowed on the MPS file before processing is terminated)
ITSCAL	30	Maximum number of iterations during scaling
ISECURE	500	Number of iterations after which the communication region is stored (in addition to secure action after each update of multipliers and termination of the run)
ITLOG	0	Level of information detail issued during optimization: 0 causes information after each update of multiplier, positive value n gives information every n steps in minimization of augmented Lagrangian function
INORM	1	L_∞ norm is assumed in the stopping criterion. For L_2 norm, 0 should be specified
IFEAS	2	Feasibility is checked if doubtful (i.e. some conditions hold - cf description of multiplier method sec. 3.4.). To check feasibility after a first update of multiplier IFEAS 1 should be stated. To force check of feasibility before entering optimization - GETF option (cf sec. 4.2.) may be used.
NT	0	Number of periods for a dynamic problem
NSTV	0	Number of state variables for a dynamic problem
MAXCRIT	10	Maximum number of criteria

Numbers or expressions given above correspond to the default values.

Note that NT=0 implies a static problem. For a dynamic problem NT should be greater then 1. For a static problem NSTV should be equal to 0.

6.5. Commands with real parameters

BIGN	1.e+30	Any number greater then BIGN is treated as infinite
TZERO	2.22e-16	Any number of absolute value less then TZERO is replaced by 0
SMALL	1.e-6	Any number in the result file with absolute value less then SMALL is replaced by 0.
FEAS	1.e-5	Feasibility tolerance
SETA	.5	Parameters for scaling (see sec. 4.6.).
SBETA	.5	
SEPS	.985	
SEP1	.01	

RO	1.	Penalty parameters for Lagrangian (see Section 4)
ROST	2.	
ROS2	4.	
ROMX	512.	
RETA	0.	Regularization parameters (see Section 4.5.)
RMET	0.	are redefined (see Section 7.5.)
RSET	0.	
EPS	0.	Stopping-criteria parameters are redefined
EPSD	FEAS	(see Section 4.2. and 7.5.)
EPSS	0.	
EPSM	1.5e-8	Maximum accuracy for minimizing of the augmented Lagrangian penalty function
MEPS	.01	Parameter of achievement scalarizing function (cf sec. 2.4.)

The parameters with values equal to zero can be changed by a user but this is not recommended. If a user does not change them, they will be computed according to the rules given in sec. 7.5. Some of the above listed parameters define tolerance which in turn should be consistent with computer machine precision (e.g. for IBM PC with math coprocessor the precision of double precision real number is equal to 2^{-52}.

6.6. Commands with mixed parameters

Both of these commands take either a real-valued parameter or the word NONE. The commands are used to define the lower and upper bounds for variables. This may be changed for selected variables through appropriate definitions in the BOUNDS section of the MPS file. The default values are the following:

LBOUND	0.
UBOUND	NONE

6.7. Names of input files

_specs	Specification of a problem to be solved
_mps	MPS data file (needed for cold start only)
_modif	Inputs data for modification of the problem
_crit	Definition of criteria for multicriteria optimization (needed for cold start only)
_comm	File that contains communication region (generated by a previous run of the problem being solved)
stdin	Standard input is used for interaction with the package

Names of input files may not be changed by a user. Files should be in a default directory (and on a default drive for PC version).

6.8. Names of output files.

stdout	Standard output is used for the diagnostics of the problem and information issued during preprocessing and optimization
__solpr	Results of optimization in MPS standard format
__userf	Results of optimization in a binary file of random access (cf sec. 8.4)
__back	A file used interchangeably (with file __comm) to secure the contents of the communication region
__byrows	Display a matrix by rows
__bycols	Display a matrix by columns
__mpso	output of MPS–format file (after transformation of a multicriteria problem to a corresponding single criterion one and after possible modification

Names of output files may not be changed by a user. Files should be in a default directory (and on a default drive for PC version).

7. USER-SUPPLIED INFORMATION

7.1. Overview

The user can supply information of three types:

- the problem specification
- the formulation of the problem
- modifications to the problem being solved

Problem specification is optional. If the specification file is empty all of the control statements take their default values. Problem specification is discussed in more detail in Section 7.2.

The formulation of the problem is necessary for the initial run (cold start) but not for subsequent or modification runs. Problem formulation consists of two parts. Firstly, one defines a problem as an LP (additional requirements apply for DLP) without multicriteria part. Secondly, one defines (if needed) multicriteria part. Problem formulation is discussed in more detail in Section 7.3.

The problem may be modified on either an initial run or a recovery run (after finding an optimal solution, in the case of an infeasible/unbounded problem or following an interrupted run). The way in which the problem may be modified is discussed in more detail in Section 7.4.

7.2. Problem specification

The user specifies the problem and may control some of the operations performed by HYBRID with the help of the specification file containing the control statements. The definitions and default values of these statements are given in Section 6.

Statements may be given in any order. The only exception is RECOVERY which – if it appears – must be the first statement in the specification file. Note that each new value for a given control statement will overwrite the previous one (either the default value or the value restored from a recovery file or previously defined in the same specification file) without any specific warning.

A statement in the specification file is recognized by the first four characters of the keyword and – if required – by a parameter following the keyword. The keywords are given in full in Sections 6.1 through 6.6 and may be used in this form for the sake of clarity. Each statement should be specified in free format on a separate line. Only the first 30 characters are processed. Blank(s) are used to separate the keyword and its parameter, and therefore blanks cannot be embedded in either the keyword or the parameters. The last column (i.e., the 30th) must contain a blank.

The specification file is read from the unit with logical number 2 until a star (*) is encountered in the first column or EOF (end of file) is reached. The user may control printing of input stream by placing the PRINT or NOPRINT statement in the specification file. Each statement is checked for validity and error messages are printed if a statement is incorrect. If the number of errors occurring during the processing of the specification file reaches 30 the run is terminated. Any error detected during processing causes termination of the run after the specification file has been processed. The diagnostics are printed on unit number 6. If no error occurs and the PRINT directive is not in effect, no information is issued. In any case, the current values of all control statements are listed in the diagnostics file.

A line that contains the character "c" in the first column followed by a blank in the second column is treated as a comment and ignored. There is no restriction on the contents of the remaining columns.

A control parameter is therefore defined by a default value, by a value restored from a recovery file, or by a statement in the specification file. The values of the parameters can also be overwritten in this sequence, i.e., a default value is overwritten by a value from a recovery file, which may itself be overwritten by a statement in the specification file.

7.3. Formulation of the problem

7.3.1. Preparation of input data file

At present a problem to be solved has to be presented in standard MPS format (cf Appendix); this may be done using a commercial problem generator (e.g. GEMINI or GAMMA) or general purpose generator (e.g. LAGOS [27]) or a generator tailored specifically to the problem. This does not apply however to specification of its multicriteria part which may be done in an easy way (cf next section). We shall therefore make only a few general suggestions and comments on this part of the system. Additional requirements for structure of MPS file apply for DLP (cf sec. 2.3).

The names of rows and columns should start with a letter or a number to avoid possible confusion with names generated by the package for multiobjective optimization problems. Names should not include a blank because of the syntax rules used in the modification routine.

Any line in the MPS file may contain a star (*) in the first column. Any such line is treated as a comment and there are no restrictions on the contents of the remaining columns.

We recommend that lower and upper bounds should be specified for all these variables and in these constraints whenever sensible values are known. This is useful in defining the admissible region over which optimization is to be performed and usually results in a decrease in computation time.

Since the computer code (if used for a DLP problem) for the algorithm applied in HYBRID is based on the structure of the problem, the proper formulation of a dynamic problem being solved is critical. Therefore, there are some restrictions concerning the form of MPS file for DLP. Those restriction which enable also diagnostic of a problem formulation, and are as follows:

1. All names for rows that correspond to state equations and columns that correspond to control and state variables should have exactly 8 characters. First six characters define so called generic name of a variable, whereas last two characters identify the period. Each name should start with a letter or with a number. Use of embedded blanks is not allowed because it would be in conflict with modification routine. By control variables we understand any variable that is defined for each time period and is not a state variable. Therefore a variable that is not defined for each time period may enter also a state equation and may have any name but such a variable should not be used for definition of a dynamic type criterion.
2. A name of a state equation row should be the same as a name of a state variable defined by that row.

3. State equations should be first in ROW section and they should be sorted by periods. Arrangement of rows defining state equations should be the same for all periods.
4. Columns are divided into two groups. First control (decision) variables should be specified in any arrangement. Secondly, the state variables should be specified, sorted by periods (i.e. first all state variables for first period). The arrangement of state variables in all periods should be the same.
5. Initial conditions for state equations should be specified in BOUNDS section (by fixing state variable corresponding to period number 0). Other constraints for state variables (if any) should be specified (preferably with use of ranges) in column section. The package removes any constraints for state variables (for all periods except initial) from BOUNDS section. If a constraint is needed for a state variable it should be specified as a general–type constraint (cf sec.2.2. and 2.3.).

A reasonable scaling of a problem is very important for numerical reliability. It is generally recommended that data and variables values should be as close to 1.0 as possible. However, since an automatic scaling is performed by HYBRID the user needs not to be very careful when scaling. The only requirement is that care should be taken in the formulation of the problem to ensure that only significant variables are included in the constraints formulation. This requirement is easily fulfilled for real-world problems. Since HYBRID provides the scaling option the user need not worry about differences in the magnitude of the coefficients.

7.3.2. Specification of multicriteria problem

A specification of a multicriteria problem has to be done in two parts. First part consists of a declarations of all criteria. The second on gives definitions of criteria.

A criterion is declared by specification of its name (four characters) and a type acronym given above with each criterion type. For MIN and MAX types additionally an overestimate of a number of linear combination components should also be given (if absent, the default value 10 is assumed). Each criterion should be declared on separate line (card image), * character in first column finishes declaration of criteria. Only 37 columns (for each card image) are processed. Should a criterion name be shorter then 4 characters, periods are added up to four characters. There are no restrictions on mixing various types of criteria.

Definition of criteria should follow declaration. Criteria may be defined in any order. Input stream (80 columns cards images) is processed until * character in first column or EOF is reached.

Each card image is assumed to contain at least two fields. Fields are separated by at least one blank.

First field should contain a criterion name or / (a continuation mark). Note that continuation may be required only for types MIN or MAX.

For types MIN and MAX second (and possibly further) field(s) contain a real number followed by * and by a name of variable (no embedded blanks allowed). A number corresponds to a coefficient associated with the specified variable in linear combination that defines given criterion.

For types SUP, INF, FOL and DER a second field contains a name, which is the name of variable that is associated with a criterion being defined. The name required is so called generic name (cf sec.2.4.3.) therefore should be composed of exactly 6 characters.

Reference trajectories (if required) should be specified in a way described in sec. 7.4.2.

Note that both declaration and definition of criteria should be made in so called cold start (initial), whereas parameters (described further on) may be changed in both cold start and subsequent computations.

7.4. Modification of the problem

7.4.1. Modification of a problem (matrix, RHS, RANGES, BOUNDS)

A user may interactively modify the problem being solved by activating the modification routine. This is activated by inserting the keyword MODIFY in the specification file (either during first or subsequent runs).

The modification lines should follow the MPS standard with the following exceptions:

1. Data are read in free format, and therefore there is no need to worry about placing data in the fields prescribed by the MPS standard.
2. Sections may occur in any order, and may also be subdivided.
3. Only 37 columns (in each card image) are processed.
4. Due to the problem of repacking the data (which has not yet been completely overcome), reclassifying a row or introducing new non-zero elements in the matrix is not allowed.
5. To remove a range the names of the rows affected should be specified with value 0. in the ranges section. Negative values are however illegal.

The data which are to be modified are read from a file *__modif* until a star (*) is found in the first column or EOF (end of file) is reached. The user may specify PRINT and NOPRINT commands which cause echoing of modification cards to standard output and suppress echoing, respectively.

Any user who does not want to follow the format restrictions imposed by the MPS standard should instead observe the following syntax rules:

1. Section names should follow MPS format.
2. Lines (with the exception of section names, comments, PRINT and NOPRINT commands and the star character that serves as an optional EOF mark) should have a blank in the first and 37th columns.
3. Fields are separated by at least one blank.
4. The number and contents of all fields must correspond to the information required by the modified section.
5. A line is treated as a comment if it contains the character "c" in the first column followed by a blank in the second column.

Since it is assumed that the sets of bounds and right-hand sides have been chosen, no associated name is needed. Thus, in the modification lines the corresponding fields should contain blanks (if prepared according to MPS format) or be absent (if in free format).

In addition to possible error diagnostics, other information is printed during modification.

Processing may be terminated if the number of errors detected during modification exceeds MERRORS (see Section 6.4).

7.4.2. Modification of multicriteria problem parameters

A user may change parameters of multicriteria problem in interactive way. To facilitate this task the same routine displays also information about last solution (cf sec. 8.3.).

The routine recognize following commands (only first character from terminal input is processed)

HELP	displays list of commands
VERB	set verbose mode of interaction; results in comments for supposed action of a user; this is the default mode
NVERB	set non-verbose mode, which result is replacing comments by acronyms
LIST	lists of information about obtained solution (cf sec. 8.2.)
DUAL	lists modified information i.e. values of dual variables (corresponding to criteria rows in the auxiliary LP problem) instead of values of nadir point, are listed.
STATUS	a user may temporary remove a criterion from achievement function; the status is stored for subsequent runs, but criterion may be restored by the same command; i.e. specification of a criterion name change status from active to non-active or vice versa
UTOPIA	a user may look for a utopia point for selected criterion; this results in setting status for all other criteria as non-active
RFP	change of reference point (aspiration level)
WEIGHT	change of weight coefficients; weight coefficients are normalized in such a way, that sum of weights is equal to number of active criteria; default values for weights are 1.
MEPS	change of value of ϵ_m coefficient
TRAJ	change of reference trajectories; one may also display values of the respective trajectory. To display a trajectory that does not enter any criterion, instead of a criterion name .... (four periods) should be entered and, next, a trajectory name should be specified.
COEF	change of coefficients in linear combination defining criteria MAX and MIN
END	exit modification status

All options that allow change of respective parameters, offer also a possibility of list of values of those parameters. An interactive way of changing parameters is fairly easy and - since a user may be guided in verbose mode - there is no need to include more details in this report.

7.5. Setting parameters

Various parameters occur in the algorithms presented in the preceding two sections. Most of them play an important role and have to be chosen very carefully. Moreover, the values of some of these parameters are (or should be) interrelated.

The values of any of the parameters may be reset by the user. If this is done, the PARAM procedure checks only whether the parameter meets certain general requirements, e.g., that it is positive. Thus the user should be very careful when making changes in parameters that affect tolerances.

Some parameters have a non-zero default value. This is generally the case for parameters that do not depend on the problem being solved. If the user specifies an unacceptable value for such a parameter, the default value is restored.

Other parameters default to an initial value of zero; the parameter values are then recomputed according to the rules given below as the program proceeds. If a user specifies a non-zero initial value which becomes unacceptable during the course of the calculation, the values computed from the following rules are restored:

RETA = 1./abs(AMXMAT), where
RETA is the initial regularization parameter,
AMXMAT is the largest of the matrix elements.

RSETA = ROST**2, where
RSETA is the coefficient for increasing RETA.

RMETA = 1.e+4/sqrt(FEAS), where
RMETA is the maximum value of the regularization parameter.

MITER = 2*(N+M),where
MITER is the maximum number of multiplier iterations,
N is the number of variables,
M is the number of constraints.

MSITER = (M+N)*N, where
MSITER is the maximum number of iterations during minimization of the augmented Lagrangian.

8. HYBRID-GENERATED INFORMATION

8.1. Initial information and problem diagnostics

The information generated may be divided into the following classes:

1. If the recovery option is activated, information about the recovery file (name of problem, date and time of creation, status of solution, size, etc.) is printed.
2. A summary of the current values of all control statements and parameters is printed.
3. On the occasion of a cold start, the input of the MPS file is reported. Error diagnostics and warnings are also issued, if applicable. This information should be self-explanatory, and therefore is not included in the examples presented in the Appendices.
4. For a multicriteria problem parameters of multicriteria definition and solution are reported. Interactive process of changing of parameters is also reported. Additionally, for a cold start only, definitions of criteria are printed.
5. If the modification option is activated the relevant information and possibly some diagnostics are provided (see Section 7.4).
6. A summary of input data and problem statistics is printed.
7. If a user overestimates the core required, a reallocation procedure is called and a report is printed.
8. If scaling is performed, this is reported.
9. The values of the parameters set by the PARAM procedure (see Section 7.5) are reported.

The storage allocation information issued after the problem has been set up refers to two parts of the communication region:

1. The fixed part (for a given version of HYBRID), which contains the values of all the control statements.
2. The working area, which contains the rest of the information and the data for the problem being solved.

Additional information may be placed in different files (see Section 6.7).

8.2. Information generated during optimization

Information may be provided at two levels of detail. The user may change the level by ITLOG option (cf sec. 6.4). The default setting (ITLOG 0) causes issuing information every time the multipliers are updated (see Section 4.2.). The alternative (ITLOG n) is to print information every n steps in the augmented Lagrangian minimization algorithm are executed; this produces vast amount of printout and should be used only for specific purposes.

The abbreviations used in printouts are explained below.

ITER	Number of iterations
RINF	Norm of gradient (L_∞)
C	Norm of gradient (L_2)
GOAL	Value of goal function
NINF	Number of infeasibilities
SINF	Sum of absolute values of infeasibilities

MAXINF	Maximum value of infeasibilities (the name of the row concerned follows)
SITC	Small iteration (i.e., number of conjugate gradient iterations)
RO	Value of penalty parameter
EPSRO	Value of EPS/RO
COM.TIME	Computation time
GDUAL	Value of the gradient of the dual function
MULT. NORM	Value of the norm of the multipliers
FDUAL	Value of the dual function
ACT. ROWS	Number of active rows
BASIC COLUMNS	Number of columns that are not equal to a given bound
ALF	Step length

In addition, a report is issued each time the communication region is stored.

Finally, exit from the optimization routine and the status of the solution is reported.

8.3. Multicriteria optimization

A user may display (by LIST command - cf sec. 5.4.2.) following information about a solution of multicriteria problem (cf Appendix 9.2. for sample of appropriate listing). The meaning of displayed information is as follows:

WEIGTS	value of a weight coefficient (note that weight coefficients are normalized as discussed in sec. 3.4.2)
RFP	component of reference point for a criterion (aspiration level for a criterion)
VALUE	value of a criterion obtained in the last run for which an optimal solution has been found
WORST	worst value of a criterion (obtained during all modified runs). This is true nadir component, if an utopia point has been calculated (see following information).
BEST	best value for a criterion (with same reservations as for WORST)
U	logical variable indicating weather utopia point for a criterion has been already calculated (t) or has been not (f)
A	status of a criterion (t for active, f for nonactive)
DUAL	if specified in place of LIST command, gives - instead of WORST value - value of a dual variable for criteria displayed

Information about reference trajectories and components a criteria of types MIN and MAX may be displayed (also by LIST command) after command TRAJ or COEF, respectively. Those commands enables also change of respective values.

8.4. Results

Results are reported in standard MPS format with an additional column that contains (in the appropriate sections) scaling coefficients for each row and column. The definitions of additional rows and columns generated during transformation of multicriteria problem to an auxiliary single-criteria one, is given in sec. 2.4.3.

Information provided in standard MPS format file is given also in a random access binary file which may be used for problem specific report writer. The structure of that file is illustrated by the following program which produces an ASCII file that contain a solution. The latter file is in format similar to the standard MPS output format, the only difference is due to replacement of the word "none" for an upper bound by the value defined as BIGN (cf sec. 6.5) and by -BIGN for a "none" in lower bound column.

```
c
c This program reads random access binary file _userf generated by
c HYBRID and outputs results in MPS format file
c Meaning of variables is as follows:
c     pname   name of the solved problem
c     status  status of solution
c     goalv   value of objective function
c     itc     number of iterations
c     m       number of rows
c     n       number of columns
c     rown    name of a row
c     coln    name of a column
c     at      status of a variable
c     val     value of a variable
c     slacv   value of a corresponding slack variable
c     vll     value of lower bound for a row or variable
c     ull     value of upper bound for a row or variable
c     vdual   value of a corresponding dual variable
c
      character*2 at
      character*8 rown, coln,pname
      character*16 status
      real*8 goalv,val,slacv,vll,vul,vdual
      integer m,n,itc
c
      open(3,file = '_userf',access='DIRECT',recl=50)
      read(3, rec=1) m,n,pname,status,goalv, itc
      write(6,1000) pname,status,goalv,itc
      write(6,1001)
      irec=2
      do 10 i=1,m
      read(3, rec=irec)rown,at,val,slacv,vll,vul,vdual
      write(6,1003) i,rown,at,val,slacv,vll,vul,vdual
10    irec = irec+1
      write(6,1002)
      do 11 i=1,n
      read(3, rec=irec)coln,at,val,slacv,vll,vul,vdual
      write(6,1003) i,coln,at,val,slacv,vll,vul,vdual
11    irec = irec+1
c
 1000 format(  '1problem name      ',a8/
     *         ' status            ',a16/
     *         ' objective value   ',g14.5/
     *         ' iteration count   ',i6/)
 1001 format(//' section 1  -  rows'//'  number  ...row..  at  '
     *,'...activity...  slack activity  .lower limit..  '
     *,'..upper limit.  dual activity. '/)
 1002 format(//' section 2  -  columns'//'  number  .column.  at  '
     *,'...activity...  obj. gradient.  .lower limit..  '
     *,'..upper limit.  reduced cost.. '/)
 1003 format(i8,2x,a8,2x,a2,5(2x,g14.5))
      end
```

9. TUTORIAL EXAMPLE

9.1. Sample of data for a multicriteria problem

For a first run a user has to prepare the following three files:

- Specification file *__specs* specifies chosen options.
- File *__crit* that contains declaration and definition of criteria.
- MPS input file *__mps*, which contains the model to be solved (however without its multicriteria part). This file, due to its volume, is not reproduced.

The *__specs* file may contain the following statements:

```
multi
nstv 2
nt 4
accept
byrows
mpso
*
```

First four statements are necessary to declare the multicriteria problem (first one), to specify that the problem is dynamic (next two) and to allow for minor errors (like one entry in a row or duplicated elements). Next two statements are optional and are for producing printouts of the resulting (after transformation of multicriteria problem to a corresponding single criteria LP) matrix by rows and in the MPS–standard format.

Declaration and specification of criteria may be done in the following way (below a contents of a *__crit* file is shown):

```
cons max
conn fol
kapt max
*
cons .95*con...01 +.9*con...02 +.86*con...03 +.81*con...04
conn con...
kapt 1.*kap01.04 +1.*kap02.04
```

The above example is for generation of three criteria. Firstly, all criteria are declared by specification of each name followed by the criterion type. First and third criteria are of type MAX, whereas the second one is of type FOL. Secondly, after a declaration (which is ended by * in first column), definition of criteria follows. For the definition of the MAX–type criteria a linear combination of variables for chosen periods are used, while for the FOL–type criterion the 6–character generic name of a variable is used. The latter simple definition is expanded for all periods.

9.2. An example of a first run

The following output was obtained for the specification file and definition of criteria presented in sec. 9.1. As an example of interactive change of parameters the initial reference trajectory for criterion *conn* has been introduced.

```
                h y b r i d  -  version 3.03  october  1987

executed on  27.10.87  at 21:23.49


options
recovery....   no      dynamic pr..  yes     multicr....  yes
getfeasible.   no      modification  no      accept.....  yes
lower bound.   yes     upper bound.  no      scaling....  yes
mpsout......   yes     byrows......  yes     bycolumns..   no
spirit......   no      comp.multp..  yes     regulariz..   no
paral. rows.   no


input files
communicat..    none      mps input...   _mps      modificat..    none
multicriter.   _crit


output files
communicat..   _comm      back-up.....   _back     solution...  _solpr
mps output..   _mpso      byrows out..  _byrows    bycols out.    none
user file...   _userf


names
problem name              objective....dummy..     (minimize)
rhs,ranges..              bounds......


dimensions
rows.........  100      columns.....  300      elements...  1500
diff. elem..  1500

integer parameters
nt..........     4      nstv........     2      maxcr......    10
iter. log...     0      inorm.......     1      max. errors    50
miter.......     0      nbc1........     0      mtime......    30
infeas......     2


real parameters
big number.  0.1000e+31  zero  tol..  0.2200e-15  small number  0.1000e-05
lower boun.  0.    e+00  upper boun.  0.1000e+31  feas. tol...  0.1000e-05
tpara......  0.1000e-05  eps........  0.    e+00  epsm........  0.1000e-06
epss.......  0.7500e+00  epsd.......  0.    e+00  ro..........  0.1000e+01
rost.......  0.2000e+01  ro max.....  0.5120e+03  ro step2....  0.4000e+01
sbeta......  0.5000e+00  seta.......  0.5000e+00  seps........  0.9850e+00
seps1......  0.1000e-01  meps.......  0.1000e-01
reta.......  0.    e+00  rmeta......  0.    e+00  rseta.......  0.    e+00
em.........  0.1000e+00  ems........  0.7600e+00  emmin.......  0.1000e-05


 44892  bytes are preliminarly assigned for communication region


input of mps format file: _mps
 card    section
    1    name          mamn04
    2    rows
                        input of criteria definition from file _crit
                           11 rows generated for multicrit. problem
   24    columns
```

```
                          9 columns generated for multicrit. problem
                         47 elements generated for multicrit. problem
  81    rhs
  90    bounds
 105    endata


indices for time periods

period no    0   1   2   3   4
index       00  01  02  03  04
variable kap01. for period 01 - set free
variable kap01. for period 02 - set free
variable kap01. for period 03 - set free
variable kap01. for period 04 - set free
variable kap02. for period 01 - set free
variable kap02. for period 02 - set free
variable kap02. for period 03 - set free
variable kap02. for period 04 - set free

realocation of memory assignement
updated max. number of rows     is    32
updated max. number of columns  is    32
updated max. number of elements is   104
updated max. number of criteria is     4
updated max. number of criteria components is     6
core requirement decreased by  39144  bytes


definition of criteria
name  type       components
cons  max       0.95000*con...01      0.90000*con...02      0.86000*con...03
                0.81000*con...04
conn  fol         con...
kapt  max       1.00000*kap01.04      1.00000*kap02.04
update of multiobjective parameters
type help if assistance is needed
type command :  h,v,n,l,d,s,u,r,w,m,t,c,e
:
routine for handling trajectories
first enter criterion name (a4) or .... for trajectory
type a command (only first letter)
 list      change    another   end
enter index of period (a2) or * (for all periods)
enter new trajectory (  4 real numbers)
type a command (only first letter)
 list      change    another   end
trajectory for variable  con...

period     01          02          03          04
ref.    0.65000     0.70000     0.75000     0.80000
type a command (only first letter)
 list      change    another   end
type command :  h,v,n,l,d,s,u,r,w,m,t,c,e
:
crit  type  weight      rfp         value       worst       best         u  s
cons  max    1.000          0.          0.      0.1000e+31 -0.1000e+31  f  t
conn  fol    1.000          0.          0.     -0.1000e+31  0.1000e+31  f  t
kapt  max    1.000          0.          0.      0.1000e+31 -0.1000e+31  f  t
meps  =  0.10000e-01
type yes/no to accept

 input summary
 =============
 objective       .dummy..   (min)
```

```
 rhs and ranges
 bounds

number of
 rows                  32   (  12 eq,  10 le,   8 ge,   2 n)
 columns               31   (max. spec.     32)
 matrix elements      103   (max. spec.    104)
 total dif. magn.      37   (max. spec.   1500)
 rhs                    8
 ranges                 0
 bounds                14

                    matrix only     matrix, rhs, ranges and bounds
 density           10.383%               -
 v                  6.2375                 6.6462
 mean              0.81937                -
 var               0.35231                -
 min. elem.       0.81000e-02             0.81000e-02
 max. elem.        1.0000                 3.1600


scaling begins
                                  matrix only     matrix,rhs,ranges,bounds
iter     gn        v       min.coef   max.coef   min.coef   max.coef
 0  96.47     0.9017      0.3716      76.15      0.1000      76.15
 1  28.83     0.7063      0.3963      30.18      0.1359      30.18      g
 2  11.35     0.6453      0.5378      38.85      0.1744      38.85      ccd
 3  5.665     0.6330      0.5084      33.57      0.1874      33.57      ccd
 4  4.138     0.6134      0.4136      35.14      0.2175      35.14      ccd
 5  2.733     0.5933      0.4705      34.90      0.2493      34.90      ccd
 6  2.304     0.5842      0.5125      36.05      0.2436      36.05      ccd
 7  2.096     0.5728      0.4876      33.99      0.2683      33.99      ccd
 8 0.5723     0.5656      0.4877      34.85      0.2754      34.85      ccd
optimal scaling   iter=   8 stop= 0.98750
in-core swap of data

after scaling
                    matrix only     matrix, rhs, ranges and bounds
 v                  0.5297                 0.6938
 mean                1.157                 -
 var                 3.065                 -
 min. elem.         0.5000                  0.2000
 max. elem.          32.00                  32.00


scaling coef.     minimal     maximal
rows             0.50000      128.00
columns          0.25000      2.0000
  problem printed by rows on file: _byrows
  problem printed in MPS format on file: _mpso

check out and setting of undefined parameters
those being set (if any) are listed below

eps     0.3200e-04
epsd    0.1000e-05
reta    0.3125e-01
rmeta   0.1000e+08
rseta   0.4000e+01
nbc1             4
miter          126
msiter        1953


communication region is saved on file    _comm
```

```
end with seting up the problem (   0.30 min. execution time )
 4448 bytes are finnally assigned for communication region
finished on 27.10.87  at  21:25.08


communiction file        _comm
problem name            mann04
saved on               27.10.87
at                     21:24.59
status                  initial
iteration count               0
size                       1112


iter =    0 grad[inf]=  32.0000     grad[l**2]= 32.0828     goal= -8.04864
ninf =   11 sinf       =  1.70000    maxinf    = 0.30000     (=conn.04)
sitc =    0 ro         =   1.0000    epsro = 0.32000e-04 comp. time       0.
gdual    =  871.61     mult. norm =  435.80     fdual = -48.334
act. rows =   15    basic columns =    5
grad[inf] = 0.29143e-15 grad[l**2] =  0.11170e-30 alf       = 0.31129


iter =   15 grad[inf]=  32.0000     grad[l**2]= 32.0199     goal= -937.620
ninf =   14 sinf       =  73.9119    maxinf    =  14.425     (<conn.01)
sitc =   17 ro         =   2.0000    epsro = 0.12000e-04 comp. time  0.16667e-01


iter =   18 grad[inf]=  3.47970     grad[l**2]= 3.62115     goal= -171.461
ninf =    7 sinf       =  10.3960    maxinf    =  2.5274     (<conn.01)
sitc =    5 ro         =   2.0000    epsro = 0.12000e-04 comp. time  0.33333e-01


iter =   21 grad[inf]= 0.787588     grad[l**2]=0.787588     goal= -59.9263
ninf =    7 sinf       =  3.75035    maxinf    = 0.78759     (<conn.01)
sitc =    3 ro         =   4.0000    epsro = 0.60000e-05 comp. time  0.33333e-01
gdual    =  12.165     mult. norm =  129.12     fdual = -7.8305
act. rows =    9    basic columns =    1
grad[inf] = 0.11102e-15 grad[l**2] =  0.12326e-31 alf       =  1.0000


iter =   22 grad[inf]= 0.787588     grad[l**2]=0.834366     goal= -34.7235
ninf =    7 sinf       =  2.17518    maxinf    = 0.39379     (<conn.01)
sitc =    1 ro         =   8.0000    epsro = 0.22500e-05 comp. time  0.33333e-01


iter =   23 grad[inf]= 0.437142e-01 grad[l**2]=0.608400e-01 goal= -9.71292
ninf =    3 sinf       = 0.236032    maxinf    = 0.10000e+00 (=conn.02)
sitc =    1 ro         =   16.000    epsro = 0.11250e-05 comp. time  0.33333e-01
gdual    =  1.7340     mult. norm =  32.150     fdual = -7.4694
act. rows =    9    basic columns =    4
grad[inf] = 0.27756e-16 grad[l**2] =  0.10374e-32 alf       =  1.0000


iter =   24 grad[inf]= 0.653591e-01 grad[l**2]=0.823002e-01 goal= -9.76953
ninf =    3 sinf       = 0.133375    maxinf    = 0.65359e-01 (=conn.02)
sitc =    1 ro         =   32.000    epsro = 0.42188e-06 comp. time  0.50000e-01
gdual    = 0.93132e-11 mult. norm =  8.0376     fdual = -7.4694
act. rows =    9    basic columns =    4
grad[inf] = 0.13878e-16 grad[l**2] =  0.20929e-33 alf       =  1.0000
communication region is saved on the backup file
communication region is saved on the communication file


iter =   25 grad[inf]= 0.277556e-16 grad[l**2]=0.280547e-16 goal= -9.84384
ninf =    0 sinf       =       0.    maxinf    =      0.    (        )
```

```
sitc =    1 ro         =   64.000     epsro = 0.21094e-06 comp. time  0.35000
itmul =    4 itotal =     29

exit optimization routine - status: optimal solution

end of run - total execution time    0.3500 min.
finished on 27.10.87  at  21:25.31


communiction file        _comm
problem name            mann04
saved on              27.10.87
at                    21:25.22
status          optimal solution
iteration count             25
size                      1112

end of run - total execution time    0.05 min.
finished on 27.10.87  at  21:25.37
```

9.3. Consecutive runs

In a first phase (cf sec. 1.2) a user may use information provided by HYBRID to verify the problem formulation. This may lead to introduction of minor changes in a matrix, which may be done with the help of *modify* option. After verification of a problem one may start actual optimization stage.

Having obtained first optimal solution a user may run the problem again for new values of the matrix elements and/or constraints and/or parameters for multicriteria optimization. Now a user may use (for any next run of the problem) the following *__specs* file:

```
recovery
nobrows
nompso
```

The first statement says the this will a so called hot start, whereas next two statements suppress output of matrix. Now a user may in an interactive way modify the problem by setting different values for a reference point or for weights. It is also possible to look for "utopia point" or to temporally remove a criterion and so on.

9.4. Implementation of HYBRID 3.03 on IIASA–VAX

The current version of IIASA–VAX manual entry may be directed to a terminal by the command ***man hybrid*** or may be printed by the command ***man -tq -r -l2 hybrid*** .

10. CONCLUSIONS

First version of HYBRID was made operational in 1982. This version is documented in [13]. Then we had improved and extended the package for dynamic linear programming problems (DLP) and for multicriteria problems (both static and dynamic). The later version in documented in [27].

HYBRID 3.03 is still a pilot–type of software that requires a lot of testing. It is true that for some problems HYBRID 3.03 performs worse than the commercial packages FMPS and MINOS but for some other problems HYBRID performs better, especially if a problem is defined as a dynamic one. If HYBRID is used not only for one run but for scenario analysis (solving the problem with change of multicriteria parameters, matrix elements, RHS etc.) its performance is much better. The reason being so is not only due to the fact that MPS file is processed only in a first run but mainly because in consecutive runs (which uses *communication region*) only update of affected coefficients is made (the problem is generated only for the first run) and because a solution is usually obtained much faster then for the first run (HYBRID – contrary to simplex approach – uses the same solution technique for any possible modification of a problem being solved).

HYBRID provides very useful diagnostics for any LP problem and therefore is also useful for a problem verification. It could be used for that purpose as "stand alone" package, and - also after possible modification of a problem in interactive way - one may output MPS-format file to be used by other packages. The same approach may be used for transformation of multicriteria problem to equivalent single-criteria LP.

The further development of HYBRID will proceed in following directions:

1. Modification of the way in which the user communicates with the package. The modification will exploit capabilities of PC compatible with IBM PC and will remarkably ease the use of the package.
2. Extensions of capabilities of HYBRID by introduction of new options for definition and handling of multicriteria problem (new types and more flexible definition of criteria, introduction of both aspiration and reservation levels, data base for previous runs etc).
3. Further improvement of the algorithm and its computer code (automatic evaluation of some parameters, experiments with possible modification of the algorithm) that will result in faster execution.

We hope that, despite the reservations outlined above, HYBRID 3.03. will eventually be a useful tool with many practical applications. We would be grateful for any criticisms and comments that would help us to improve the package.

11. REFERENCES

1. M. Kallio, A. Lewandowski and W. Orchard-Hays. An implementation of the reference point approach for multiobjective optimization. WP-80-35. International Institute for Applied Systems Analysis, Laxenburg, Austria, 1980.
2. M. Makowski and J. Sosnowski. A decision support system for planning and controlling agricultural production with a decentralized management structure. In: Plural Rationality and Interactive Decision Processes. Eds. M. Grauer, M. Thompson, A.P. Wierzbicki. Springer – Verlag, 1985.
3. A.P. Wierzbicki. A methodological guide to multiobjective decision making. WP-79-122. International Institute for Applied Systems Analysis, Laxenburg, Austria, 1979.
4. M. Makowski and J. Sosnowski. Implementation of an algorithm for scaling matrices and other programs useful in linear programming. CP-81-37. International Institute for Applied Systems Analysis, Laxenburg, Austria, 1981.
5. O.L. Mangasarian. Iterative solution of linear programs. *SIAM Journal for Numerical Analysis*, 18(4): 606–614, 1981.
6. B.T. Polyak and N.V. Tretiyakov. An iterative method for linear programming and its economic interpretation. *Economic and Mathematical Methods*, 8: 740–751, 1972 (in Russian).
7. J.S. Sosnowski. Linear programming via augmented Lagrangian and conjugate gradient methods. In S. Walukiewicz and A.P. Wierzbicki (Eds.), *Methods of Mathematical Programming*, Proceedings of a 1977 Conference in Zakopane. Polish Scientific Publishers, Warsaw, 1981.
8. D.P. Bertsekas. Multiplier methods: a survey. *Automatica*, 12: 133–145, 1976.
9. B.A. Murtagh and M.A. Sanders. MINOS – A large-scale nonlinear programming system (for problems with linear constraints). User guide. Technical Report, Systems Optimization Laboratory, Stanford University, 1977.
10. B.T. Polyak. The conjugate gradient method in extremal problems. *Computational Mathematics and Mathematical Physics*, 9: 94–112, 1969.
11. D.P. O'Leary. A generalized conjugate gradient algorithm for solving a class of quadratic problems. *Linear Algebra and its Applications*, 34: 371–399, 1980.
12. M.R. Hestenes. Conjugate Gradient Methods in Optimization. Springer Verlag, Berlin, 1980.
13 M.Makowski, J. Sosnowski Hybrid: A mathematical programming package, IIASA, 1984, CP-84-9.
14 A.R. Curtis, J.K. Reid On the automatic scaling of matrices for Gaussian elimination, *Journal of Mathematics and its applications*, 1972, no 10, pp.118–124.
15 J.A.Tomlin, On scaling linear programming problems, *Mathematical Programming Study 4*. North Holland Publishing Company, 1972, Amsterdam
16 J.S. Sosnowski, Dynamic optimization of multisectorial linear production model. Systems Research Institute, Warsaw, Ph.D. Thesis, (in Polish), 1978.
17 A. Propoi, Problems of Dynamic Linear Programming, IIASA, RM-76-78
18 R. Fourer, Solving staircase linear programs by the simplex method, *Mathematical Programming* 23 (1982) 274-313, 25(1983) 251-292
19 J.K.Ho, A.S. Hanne, Nested decomposition for dynamic models. *Mathematical Programming* 6 (1974) 121-140
20 A. Lewandowski and Grauer M., The reference point optimization approach - methods of efficient implementation. CP-8-S12, IIASA Collaborative Proceedings Series: Multiobjective and Stochastic Optimization Proceedings of an IIASA Task

Force Meeting, 1982

21 A.Wierzbicki, A mathematical basis for satisficing decision making, WP-80-90, IIASA, 1980).

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

23 A.Wierzbicki, On the use of penalty functions in multi- objective optimization, Institute of Automatics, Technical University of Warsaw, 1978.

24 B.A. Murtagh, Advanced Linear Programming: Computation and Practice, Mc Graw–Hill, New York, 1981.

25 Kreglewski, T., Lewandowski, A. and Rogowski, T. (1984) Dynamic Extension of the DIDAS system and its Application in Flood Control. In: Plural Rationality and Interactive Decision Processes. Eds. M. Grauer, M. Thompson, A.P. Wierzbicki. Springer – Verlag, 1985.

26 Murtagh B.A. and Sanders M.A., A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints, *Mathematical Programming Study* 16 (1982), 84-117

27 Makowski M., Sosnowski J., HYBRID 2.1: A mathematical programming package for multicriteria dynamic problems. In: A.Lewandowski, A. Wierzbicki eds., Theory Software and Testing Examples for Decision Support Systems, IIASA, Laxenburg, September 1985.

28 Makowski M., Sosnowski J., Methodological Guide to HYBRID 3.01: a mathematical programming package for multicriteria dynamic problems. In: A.Lewandowski, A. Wierzbicki eds., Theory Software and Testing Examples for Decision Support Systems, WP-87-26, IIASA, Laxenburg, April 1987.

29 Murtagh B.A., Saunders M.A., MINOS 5.0 User's Guide, Technical Report SOL 83-20, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, December 1983

APPENDIX

MPS standard for input of data

A linear programming problem is usually defined in two parts. Firstly, a specification of problem is given. The is no standard for a problem specification, therefore we have adopted for HYBRID a very permissive way for problem specification which ease the specification and modification of the problem.

Secondly, the nonzero elements of matrix have to be entered. Most codes designed for solving linear programming problems follows the data format originally designed to the MPS series of codes developed for IBM computers. This format has become de facto standard adopted by most codes designed There are slight variations between codes, therefore we present the following specification which is accepted by most codes and do not restrict possibilities offered by the original standard developed for IBM computers.

The data that correspond to a linear programming problem is grouped in the following sections:
NAME
ROWS
COLUMNS
RHS
RANGES (optional)
BOUNDS (optional)
ENDATA

A section name must be entered starting with first column. The above listed order is compulsory. Section name should be entered in all either lower or upper cases (this depends on installation). Data in each section should be entered in a card image which has the following fields and corresponding contents:

	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
Columns	2-3	5-12	15-22	25-36	40-47	50-61
Contents	Indicator	Name	Name	Value	Name	Value

The indicators are listed below (in sections ROWS and BOUNDS). Indicators should be entered in all either lower or upper cases (this depends on installation). A value should be entered in floating point format.

We will examine each of the section in turn:

NAME — This section consists of just the section name card with a title of the problem placed in columns 15-22.

ROWS — In this section all the rows names are defined together with the row type. The row type in entered in Field 1 (in column 2 or 3) and the indicators have the following meaning :

- E equality
- L less than or equal
- G greater than or equal
- N free (no restriction); the first free-type row encountered is regarded as the objective row, unless the objective is explicitly identified in the specification file or by a code itself (the latter case applies for multiobjective optimization).

COLUMNS — This section defines the variables and the coefficients of the constraints matrix (including the objective row). Only nonzero coefficients are entered. The data are entered column by column and all data for nonzero

entries in each column must be grouped together contiguously. The card image contains the column name in Field 2, the row name in Field 3 and the value of coefficient in Field 4. One may enter two coefficients for the same column on one card by placing the name of the second row in Field 5 and the value of coefficient in Field 6. Columns for slack variables are taken care of by a code.

RHS — This section contains the non-zero elements of the right-hand sides of constraints. The data format corresponds to COLUMNS section, the only difference is due to replacement of a column name by a label (with may also be blank). More than one right-hand side set may be specified in this section (the one to be used for the current run is specified in the specification file by its label). Therefore one card image contains the optional label in Field 3, the row name in Field 4 and the value of right-hand side in Field 5.

RANGES — This section contains entries for inequalities rows for which both lower and upper bound exist. The data format is the same as for RHS section. The value of range is interpreted as the difference between the upper and the lower bound for the respective row. One of those bounds (if nonzero) is entered in the RHS section, the type of the row indicates whether lower or upper bound in defined in the RHS section.

BOUNDS — This section contains changes for bounds for variables initially set to default values. The default bounds are usually defined as 0. for the lower bound and no constraint for the upper bound. The default bounds (for all variables) may be usually changed in the specification file. More than one set of bounds may be specified in this section (the one to be used for the current run is specified in the specification file by its label). More than one bound for a particular variable may be entered. The bound indicators have the following meaning :

LO lower bound

UP upper bound

FX fixed value for the variable

FR free variable (no bounds)

MI no lower bound, upper bound equal to 0.

PL no upper bound, lower bound equal to 0.

The card image has the following data format: Field 1 contains indicator of the type of bound, Field 2 contains optional label, Field 3 specifies the column name and Field 4 specifies value of bound, if applicable.

ENDATA — This section consists of just the section name card, signalling the end of data.

An example of MPS standard input file

The following example contains the input data file in MPS standard for the problem presented in sec. 5.1. The problem has been generated for two periods of time. The meaning of variables is as follows:

conn.. consumption

inv... investment

kap... capital

Note that two last characters in a variable name correspond to a period number. The line that contains only numbers and underscores serves as a ruler and is not a part of the MPS format data file.

```
name            mann02
rows
 e  kap...01
 e  kap...02
 l  mon...01
 l  mon...02
 g  cka...01
 g  cka...02
 n  goal
columns
123456789_123456789_123456789_123456789_
    con...01  goal              0.95
    con...01  mon...01          1.00
    con...02  goal              0.90
    con...02  mon...02          1.00
    inv...01  kap...01          1.00
    inv...01  mon...01          1.00
    inv...02  kap...02          1.00
    inv...02  mon...02          1.00
    kap...00  kap...01          1.00
    kap...00  mon...01         -0.27
    kap...01  kap...01         -1.00
    kap...01  kap...02          1.00
    kap...01  cka...01          1.00
    kap...01  mon...02         -0.28
    kap...02  kap...02         -1.00
    kap...02  cka...02          1.00
rhs
    test1     cka...01          3.16
    test1     cka...02          3.16
    test2     cka...01          4.11
    test2     cka...02          4.11
bounds
 up bnd1      inv...01          0.17
 up bnd1      inv...02          0.17
 lo bnd1      con...01          0.65
 lo bnd1      con...02          0.65
 fx bnd1      kap...00          3.00
 fr bnd1      kap...01
 fr bnd1      kap...02
endata
```