Mathematical Programming Package HYBRID

Marek Makowski

*IIASA, Laxenburg, Austria.**

Janusz S. Sosnowski

Systems Research Institute, Polish Academy of Sciences, Warsaw.

Abstract

HYBRID is a mathematical programming package which includes all the functions necessary for the solution of multicriteria LP problems and single-criteria linear-quadratic problems. HYBRID is specially useful for dynamic problems since the applied algorithm exploits the structure of a dynamic problem and the user 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. 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. Besides that HYBRID offers many options useful for diagnostic and verification of a problem being solved. HYBRID is available in two versions: one for VAX 6210 (running under Ultrix-32) and one for a PC compatible with PC IBM/AT/XT.

1 Introduction

The purpose of this report is to provide a sufficient understanding of mathematical, methodological and theoretical foundations of the HYBRID package. Section 1 contains executive summary, short program description and general remarks on solution techniques and package implementation. Section 2 contains mathematical formulation of various types of problems that can be solved by HYBRID. Section 3 presents methodological problems related to solution techniques. Section 4 presents foundations of the chosen solution technique and documents the computational algorithm. Section 5 contains short discussion of testing examples. Last two sections contain conclusions and references.

*on leave from the Systems Research Institute of the Polish Academy of Sciences, Warsaw.

This paper does not include information necessary for using the package. A reader who is interested in usage of the package should consult a User Guide to HYBRID (Makowski and Sosnowski, 1988b). In the User Guide the following topics are discussed:

- the way of choosing various options provided by the package.
- guidelines for formulation and modification of a problem which is to be solved or at least processed by HYBRID.
- the way in which HYBRID provides diagnostics and results.
- a short tutorial example.
- 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, called HYBRID 3.1, may be used for solving both static and dynamic LP problems (in fact also for problems with a more general structure then the classical formulation of dynamic linear problems). HYBRID 3.1 may be used for both single- and multi-criteria LP problems as well as for single-criteria linear-quadratic 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 the DIDAS family decision analysis and support systems since it is designed to support usage of multicriteria reference point optimization. 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 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, Makowski and Sosnowski, 1985a), flood control in a watershed (Kreglewski et al., 1985), 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 the decision maker to concentrate on his actual decision tasks 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.

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. HYBRID includes also some diagnostic functions that 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 (Propoi, 1976). Such problems are also called multistage or staircase linear programming problems (Fourer, 1982, Ho and Hanne, 1974). 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. One of the first versions of the system DIDAS was a package composed of a preprocessor and a postprocessor for handling transformation of multicriteria problem and processing results respectively (Lewandowski and Grauer, 1982). Those pre- and postprocesors were linked with an LP package. HYBRID 3.1 has generally a similar structure. The main difference is that—instead of an LP package—another non-simplex algorithm is applied, which exploits the dynamics of a problem and that HYBRID is integrated package with user–friendly interface. 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 binary files with HYBRID 3.1 are available from IIASA in two versions: one for VAX 6210 (running under Ultrix-32 ver. 3.0) and one for a PC compatible with IBM/AT/XT.

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 model is a complex task and this paper

is not devoted to discuss this question in detail. 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 model 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 model usually requires cooperation of a specialist—who knows 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 model. 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, Paradox, Oracle or Symphony). 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 model.

3. Verification of the data base and of the model formal definition should be performed.

4. The corresponding MPS standard file should be created. This may be done by a specialized model 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 LPL (cf Hurlimann, 1988), 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 Model 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, the 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 Multiobjective problem analysis

For a given model, the user can define various multiobjective problems to be analyzed. 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 make at this stage 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 model. 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 sect. 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 multiobjective analysis 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 models. Static models can be interpreted as models 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 models, 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 models.

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 detail later.

3. A model can be declared as a dynamic one by the definition of periods of time. For a dynamic model, 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 models. The formulation of a dynamic model, which is accepted by HYBRID, is actually an extension of the classical formulation of a dynamic model (cf Section 2.2.). In our 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 software 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.

The package consists of five subpackages:

- Two preprocessors that serve to process data, enable a modification of the model, perform diagnostics and may supply information useful for the verification of a model. The first preprocessor is used for processing of initial formulation and diagnostics of the model. It also transforms a multicriteria problem to a parametric single criteria optimization problem. The second preprocessor allows for analysis of a solution and for the interactive change of various parameters that may correspond to choice of some option, change of parameters in definition of multicriteria problem, change of matrix coefficients, right hand sides of constraints etc.

- Optimization package called solver of a relevant optimization problem (either static or dynamic).

- Postprocessor that provides results in the standard MPS format and generates 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.

- Driver, which eases the usage of all subpackages. The PC version of driver provides a context sensitive help which helps an inexperienced user in efficient usage of the package.

All five subpackages use a binary file that contains all data defining the problem being solved. A second binary file contains a solution obtained by last run of the solver. From the user point of view, HYBRID 3.1 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 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 coefficients which take different numerical values. The memory management is handled by a flexible way. HYBRID is coded partly in C and partly in an extension of Fortran (the latter part is processed by a preprocessor to generate a code which conforms to Fortran 77 standard). Such approach results in a faster (much faster for PC version running under DOS) execution and in a decrease of memory requirements.

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 sect. 4.7).

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. Murtagh, 1981, Makowski and Sosnowski, 1988b).

1.2.6 Outline of the solution technique

HYBRID uses a non-simplex algorithm — a particular implementation of the augmented Lagrangian (or 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. So far only single criteria linear-quadratic problems in the form discussed in Section 2.5 may be solved. The solution method for multi-criteria quadratic problems requires modification of the algorithm. However the necessary modifications will be

based on HYBRID 3.1.

1.2.7 General description of options provided by the package

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).

- 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).

- The multicriteria problem is formulated and solved as a sequence of parametric optimization problems modified in interactive way upon analysis of previous results.

- The solution technique can be chosen. First choice is done by definition of a static or a dynamic problem. Some specialized techniques may be used for badly conditioned problems that usually cause numerical problems. This includes one of two regularization techniques (see Section 4.5) and/or possibility of using preconditioned conjugate gradient method (cf Section 4.6). For a badly scaled problem, an implementation of scaling algorithm is available (as described by Makowski and Sosnowski (1981) and briefly discussed in Section 4.7).

- 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. More detailed information for an infeasible or unbounded problem is optionally provided by the package.

- All data that correspond to the formulation of the problem being solved are stored in a binary file. An other binary file contains all other information corresponding to a current run. The latter file 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. Therefore the MPS input file is read and processed only by first preprocessor, which serves for initial formulation of the problem. Such approach allows also for efficient storing of many solutions that may be later used for more detailed analysis, comparisons and modifications.

- Any solution is available in the standard MPS format and in a binary file which contains all data that might be useful for postoptimal analysis and reports.

1.3 Remarks on implementation

HYBRID 3.1 is an extended version of HYBRID 3.03 documented in (Makowski and Sosnowski, 1988a, 1988b). Therefore there are only small changes in the methodological guide in comparison to the methodology presented in (Makowski and Sosnowski, 1988a), because the solution techniques are basically the same. However, there are some important methodological innovations. The main differences are the following:

- The code has been modified as to allow for solution of single criteria linear-quadratic problems.
- The preconditioned conjugate gradient technique for minimizing augmented Lagrangian has been implemented.
- The second regularization option which allows for finding the optimal solution with minimum distance from a given reference point has been made operational.
- 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 user interface (for PC version of the code) has been improved. A new approach to usage of the package and to data handling provides for easier use of the package.
- Diagnostics have been improved and several observed bugs have been removed.
- Part of the code has been rewritten in C language. This allows for more efficient memory management and usage. Change in the way of internal data handling resulted in remarkable improvement of execution speed.

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., Murtagh and Sanders, 1977):

$$\min cx \tag{1}$$

$$b - r \leq Ax \leq b \tag{2}$$

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

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) and simple constraints (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 3.1, let us first consider a classical formulation of a dynamic linear programming problem (CDLP) (cf Propoi, 1976) in the following form:

Find a control trajectory

$$u = (u_1, \ldots, u_T)$$

and a state trajectory

$$x = (x_1, \ldots, x_T)$$

satisfying the state equations with initial condition x_0

$$x_t = A_{t-1}x_{t-1} + B_t u_t - c_t \tag{4}$$

and constraints

$$d_{t-1} - r_{t-1} \leq F_{t-1}x_{t-1} + D_t u_t \leq d_{t-1} \qquad t = 1, \ldots, T \tag{5}$$

$$e_t \leq u_t \leq f_t \qquad t = 1, \ldots, T \tag{6}$$

$$F_T x_T \leq d_T \tag{7}$$

which minimize the performance index

$$\sum_{t=1}^{T} (a_t x_t + b_t u_t) \tag{8}$$

where:

- $t = 1, \ldots, T$ 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 (5), equality constraints can be used;

2. Since no constraints of bounds type (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 (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 Propoi, 1976) 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 3.1 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 sect. 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:

$$-H_t x_t + \sum_{i=0}^{t-1} A_{t-1,i} x_i + B_t u = c_t, \qquad t = 1, \ldots, T \tag{9}$$

with given initial condition x_0
and constraints:

$$d - r \leq \sum_{t=0}^{T} F_t x_t + Du \leq d \tag{10}$$

$$e \leq u \leq f \tag{11}$$

are satisfied and the following function is minimized:

$$\sum_{t=1}^{T} a_t x_t + bu \tag{12}$$

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 sect. 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 (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 3.1 is the reference point approach introduced by Wierzbicki (1980). Since the method has been described in a series of papers and reports and has been applied to DIDAS (cf Kallio et al., 1980, Lewandowski and Grauer, 1982), 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 sect. 2.4.2). The definition of criteria in HYBRID can be performed in an easy way described in the User Manual.

2. The DM specifies an aspiration level $\bar{q} = \{\bar{q}_1, \ldots, \bar{q}_{NC}\}$, 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 sect. 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

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

This form of achievement function is a slight modification of a form suggested by A. Lewandowski (1982) and by A. Wierzbicki (1978). Note that for $\varepsilon_m = 0$ only weakly Pareto-optimal points can be guaranteed as minimal points of this function. Therefore, the use of a very small ε_m results (except of situations in which reference point has some specific properties) in properly Pareto-optimal solution with trade-off coefficients bounded approximately by $\varepsilon_m NC$ and $1/\varepsilon_m NC$. If ε_m is very small, these properly efficient solutions might practically not differ from weakly efficient (Pareto optimal). 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 $x_i = \{x_{it}\}$, $t = 1, \ldots, T$.

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

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

The following four criteria types are exclusively for dynamic LP:
Type SUP

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

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

Type INF

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

Type FOL

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

Type DER (which applies only to state variables)

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

2.4.3 Transformation of multicriteria problem to an auxiliary LP

The transformation is done by HYBRID 3.1, 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,

$x_j = \{x_{jt}\}$, $t = 1,\ldots,T$ 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, ‘ $\cdots$ ‘) 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 $t = 1, \ldots, T$) 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 $t = 1, \ldots, T$) 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 $t = 1, \ldots, T$) 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 $t = 1, \ldots, T$) 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 $t = 1, \ldots, T$) 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 $t = 1, \ldots, T$) 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:

$$v + \varepsilon_m \left(\sum_i w_i q_i + \sum_t \Big(\sum_j w_j x_{jt} + \sum_k w_k (c^+_{kt} + c^-_{kt}) \Big) \right)$$

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.

2.5 Formulation of single criteria linear-quadratic problems

HYBRID 3.1 allows for solution of a single-criterion linear-quadratic problem with a simple quadratic term. For a problem which does not have recognized structure (as discussed in sect. 2.3) the formulation takes the following form:

$$\min cx + (\gamma/2)\|x - \bar{x}\|^2$$

subject (2) and (3), where $\bar{x}$ is a given point in the solution space and $\gamma > 0$ is a given parameter.

Similarly, for a dynamic problem one may formulate the problem in the following way:

$$\min \sum_{t=1}^{T} a_t x_t + bu + (\gamma/2)\|u - \bar{u}\|^2$$

subject (9) and (11), where $\bar{u}$ is a given point in the space of independent variables.

3 Theoretical foundations and 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 (Mangasarian, 1981, Polyak and Tretiyakov, 1972, Sosnowski, 1981). 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 (Bertsekas, 1976) (see Sections 4.2 and 4.4).

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 4.5).

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 the 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:

$$\min cx$$

$$Bx \leq d \tag{PO}$$

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

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

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

and

$$y^{k+1} = (y^k + \rho(Bx_{k+1} - d))_+ \tag{13}$$

where

$$L(x, y^k) = cx + \left(\|(y^k + \rho(Bx - d))_+\|^2 - \|y^k\|^2 \right) / (2\rho) \tag{14}$$

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., Sosnowski, 1981, Bertsekas, 1976). 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 (Flecher, 1981). 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 (1969), O'Leary (1980) and Hestenes (1980) for minimization of a quadratic function on an interval using the conjugate gradient method.

The problem (P) may be reformulated as follows:

$$\min cx$$

$$Ax + z = b$$

$$l \leq x \leq u \qquad \text{(PS)}$$
$$0 \leq z \leq r$$

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 augmented Lagrangian 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

$$L(x,z,y) = cx + \left(\|y + \rho(Ax + z - b)\|^2 - \|y\|^2\right) / (2\rho). \qquad (15)$$

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

$$F(x,z) = (c/\rho)x + (\|y/\rho + Ax - b + z\|^2 - \|y/\rho\|^2) / 2. \qquad (16)$$

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^*):

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

and

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

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

$$z_i = \begin{cases} 0 & \text{if} \quad g_i \leq 0 \qquad (\partial F/\partial z_i > 0) \\ r_i & \text{if} \quad g_i \geq r_i \qquad (\partial F/\partial z_i < 0) \\ g_i & \text{if} \quad r_i > g_i > 0 \quad (\partial F/\partial z_i = 0). \end{cases} \tag{17}$$

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. 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.,

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

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

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

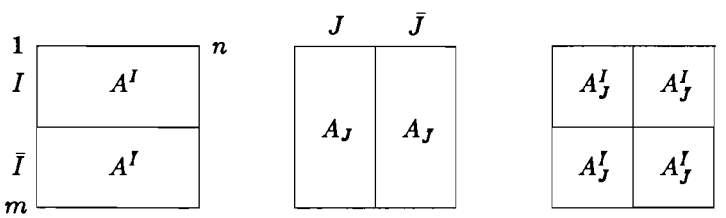
The set of indices I can be also interpreted as a set of active simple constraints for z. The set of indices J of variables that should be equal to either the upper or the lower bound, i.e.,

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

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

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

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_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 (17).

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 $\bar{I}$

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

$$\bar{I} = \{1, \ldots, m\} \setminus I$$

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

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

5. Compute the minus gradient:

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

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

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

$$\bar{J} = \{1, \ldots, n\} \setminus J$$

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 $p^J = q^J$

9. Compute

$$s = A_J p^J$$

$$h = \|q^J\|^2$$

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

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

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

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

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

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

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

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

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

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 $x^J := x^J + \alpha p^J$ and the minus gradient at that point:

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

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

14. If $q^J = 0$. go to step 2

15. If $\alpha = \alpha(1)$ continue with the conjugate gradient step, i.e.

$$\beta = \|q^J\|^2/h$$

$$p^J := q^J + \beta p^J$$

and go to step 9

16. Go to step 8

Note that the condition $q^J = 0$ is in practice replaced by $\|q^J\| \leq \varepsilon$, 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

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

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

3. Compute new multipliers

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

4. If $\|y^{k+1} - y^k\| > \varepsilon_d$ 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 $\varepsilon^{k+1} = \varepsilon^k \varepsilon_s$, where $\varepsilon_s < 1.$ 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 $\varepsilon^{k+1} = \varepsilon^k \varepsilon_s$ and go to step 2.

4.3 Solution technique for DLP

We will not repeat reasoning given in the first part of sect. 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 sect. 3.3) shows advantages due to the use of dynamic structure of DLP problem in comparison with presentation of such a problem as a large LP.

2. The structure of matrices $B_1, \ldots, B_T$ and $F_0, \ldots, F_T$ has no impact for the algorithm nor affects the technique of storage of data, because super-sparse technique is applied (cf sect. 1.4). It should be also pointed out that the method of transforming a multicriteria problem to a parametric LP one introduces constraints (cf sect. 2.4.3) that—for the proposed (cf sect. 2.4.2) types of criteria—do not fit to the staircase structure of CDLP (cf Propoi, 1976). 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 sect. 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 sect. 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 sect. 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

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

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 sect. 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

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

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

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

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

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

   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 $\bar{J}$

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

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 $p^J = q^J$

10. Solve state equation in variations

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

with boundary condition $\sigma_0 = 0$

11. Compute

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

$$h = \|q^J\|^2$$

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

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

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

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

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

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

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

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

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

14. 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 $\bar{J}$.

15. Compute :

$$u^J := u^J + \alpha p^J$$

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

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

16. For the new g^I solve the conjugate equation (as in step 5)

17. Compute the minus gradient :

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

18. If $q^J = 0$, then go to 2

19. If $\alpha = \alpha(1)$ continue with the conjugate gradient step, i.e.

$$\beta = \|q^J\|^2/h$$

$$p^J = q^J + \beta p^J$$

and go to step 10

20. Go to step 9

Note that the condition $q^J = 0$ is in practice replaced by $\|q^J\| \leq \varepsilon$. 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 (Sosnowski, 1981). 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 way of finding the optimal solution with either minimum Euclidian norm or minimum distance from a given reference point. The first of these options may be activated by a REGZERO statement in the specification file. The second may be chosen by REGREF statement. For the latter case the non-zero values of $\bar{x}$ (see the following formulae) are defined in additional section (called *reference*) in the MPS input file.

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 (Sosnowski, 1978). Thus minimization of $L(\cdot, y^k)$ in problem (PO) is replaced by

$$x^{k+1} = \operatorname{argmin}_x \; L(x, y^k) + \|x - \bar{x}\|^2/(2\eta^k)$$

where $\bar{x}$ is given and

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

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.

4.6 Preconditioned conjugate gradient for minimization of augmented Lagrangian

The algorithm for minimization of the augmented Lagrangian (cf sect. 4.1) theoretically guarantees that the exact solution of a problem will be found after finite number of iterations. However, during the actual computations the rounding errors often cause numerical problems. For accelerating the convergence, the augmented Lagrangian can be minimized by using the preconditioned conjugate gradient method. This method is discussed in details for linear problems in (Golub and Van Loan, 1983) and for least squares problems in (Heath, 1984). Therefore we present only brief summary of the applied approach.

Let us consider again the algorithm for minimization of the augmented Lagrangian penalty function. Assume that the steps 8 – 15 of the algorithm are executed for fixed sets of indices $I, \bar{J}$. In other words, the conjugate gradient algorithm is used for minimization of a quadratic function which has the following Hessian matrix:

$$H = (A^I_J)^T (A^I_J)$$

A matrix M which approximates the Hessian matrix H and for which it is easy to solve the the linear system:

$$Mp^J = q^J$$

will be referred to as the *preconditioning (or scaling) matrix.*

For a given preconditioning matrix M and under assumption that the sets of indices $I, \bar{J}$ do not change, we can use the following modification of the steps 8 – 15 of the

algorithm 4.1. This modified algorithm will be called preconditioned conjugate gradient algorithm for minimization of the augmented Lagrangian penalty function. Note that steps 10., 11., 12. are void since the sets of indices $I, \bar{J}$ do not change.

8. Set $p^J = M^{-1}q^J$

9. Compute

$$s = A_J p^J$$
$$h = (q^J)^T M^{-1} q^J$$
$$d = \|s^J\|^2$$
$$\alpha = h/d$$

13. Compute the new point $x^J := x^J + \alpha p^J$ and the minus gradient at that point:

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

14. If $q^J = 0$. then STOP

15. Continue with the preconditioned conjugate gradient steps:

$$\beta = (q^J)^T M^{-1} q^J / h$$
$$p^J := M^{-1} q^J + \beta p^J$$

go to step 9.

There is no general rule for a choice of a scaling matrix. Therefore we will discuss briefly possible preconditioned matrix selection options for general formulation of LP problems and for dynamic LP problems.

4.6.1 Scaling matrix for LP problems

For LP problems that do not have a special structure of the matrix A, a scaling matrix M should be an approximation of $H = (A^I_J)^T(A^I_J)$.

Diagonal scaling : A very simple method of scaling which requires only n elements to be stored results in a matrix M which is the diagonal of H.

Let the matrix M be equal to a matrix W^I_J which is defined (for given sets of indices $I, \bar{J}$) as follows:

$$W^I_J = diag(w_j) \qquad j \in \bar{J}$$

where $w_j = \sum_{i \in I} a^2_{ij}$. The matrix W^I_J can be easily updated if the indices sets $I, \bar{J}$ are changed. If a new index k is added to the set I then the respective element should be updated in the following way: $w_j := w_j + a^2_{kj}$. If an index is removed from the set $\bar{J}$ a column and a row should be discarded from the matrix W^I_J.

Cholesky factorization of the scaling matrix : Let n_J be dimension of a subspace in which we minimize the quadratic function, and n_d be an integer number. Assume that after $n_J + n_d$ steps of the ordinary or diagonal scaling conjugate gradient algorithm a minimum is not found. In such a case we can use as the scaling matrix the Hessian:

$$M = (A_J^I)^T (A_J^I)$$

and apply Cholesky factorization for obtaining an inverse of the scaling matrix ($M = R^T R$, where R is upper triangular matrix). Application of such scaling matrix usually results in minimimization of the function in one step of the algorithm.

4.6.2 Scaling matrix for dynamic LP problems

The main advantage of the conjugate gradient algorithm for dynamic LP problems is due to the reduction of a dimension of working space to a dimension of a subspace of independent variables. Therefore an implementation of the preconditioned conjugate algorithm should use as a scaling matrix a matrix which approximates the reduced Hessian. For a general structure of the dynamic problem, the form of the reduced Hessian is rather complicated, thus we suggest a different method for selection of a scaling matrix.

Let us consider the algorithm for minimization of the augmented Lagrangian penalty function – section 4.4. with the assumption that the steps 9 – 20 of the algorithm are executed for fixed sets of indices $I, \bar{J}$.

Matrix scaling : Let $p_k^J, k = 1, \ldots, n_J$ be conjugate directions obtained during n_J steps of the ordinary conjugate gradient algorithm, where n_J is also dimension of the working space. Then an inverse of the reduced Hessian matrix Q may be calculated as:

$$Q = \sum_{k=1}^{n_J} p_k^J (p_k^J)^T / d_k$$

where $d_k = \|s_k^I\|^2$. The preconditioned conjugate gradient algorithm can be used with $M^{-1} = Q$. The matrix M^{-1} may be obtained in the following way:

$$Q_0 = 0$$

$$Q_k = Q_{k-1} + p_k^J (p_k^J)^T / d_k \qquad k = 1, \ldots, n_J$$

$$M^{-1} = Q_{n_J}$$

Diagonal scaling : Instead of using the matrix Q defined above one can use only its diagonal. In such a case if after n_J steps of the conjugate gradient algorithm a minimum is not found, the preconditioned conjugate gradient algorithm is initiated with the scaling matrix $M^{-1} = diag(Q)$.

4.7 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 (1972), 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 (1972) for solving the scaling problem.

Our approach is discussed in details in (Makowski and Sosnowski, 1981), 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:

$$Ax = b \tag{18}$$

$$d \leq x \leq q$$

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

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

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

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

Therefore, instead of solving the original problem (18), one can solve an equivalent problem in form

$$(RAC)y = Rb$$

$$C^{-1}d \leq y \leq C^{-1}q$$

$$x = Cy$$

Here $R = diag(r_1, \ldots, r_m)$ and $C = diag(c_1, \ldots, c_n)$ 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

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

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:

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

$$\gamma \sum_{i \in L} \Big(\log(abs(c_j bnd_i))\Big)^2 \rightarrow \min$$

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

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

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

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 several known examples that cover different types of problems have been selected.

5.1 Economic growth model (Manne)

This model is a linear multicriteria version of Manne's model described in (Murtagh and Sanders, 1982).

The variables have the following meaning:

t – time period, $t = 1, 2, \ldots, T$

c_t – consumption

i_t – investment,

k_t – capital in time period t.

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

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

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

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

The state equations have the following form:

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

with k_0 given.

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

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

$$k_t \geq k_0 + i_0$$

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

$$c_t \geq c_0$$

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

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 following table 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 Kreglewski et al., 1985) 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 following table 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 (Mangasarian, 1981) 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 $a_{ii} = i$ are selected.

In the following table 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 Linear programming test problems

Four examples from a widely used set of test problems (cf e.g. Gay 1986), namely: Afiro, Adlittle, Share2b and Israel have been also used as testing examples. The last three problems result in badly conditioned Hessian matrices of the augmented Lagrangian.

5.5 Discussion of test results

Testing problems have been solved on a PC compatible with IBM/AT (running at 8 MHz) with 80287 coprocessor (running effectively at 5.3 MHz). 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. Numbers of rows and columns correspond to a single criterion LP problem, which were obtained by transformation of relevant multicriteria problems. The numbers of gradient iterations correspond to execution of step 8 of the algorithm (cf sect. 4.1).

Problem	Number of crit.	Rows	Cols	Dens. [%]	Time (min.)	Mult. iter.	Grad iter.
Manne05	3	29	27	12	0.08	2	21
Manne10	3	54	52	7	0.18	2	45
Manne20	2	103	102	3	0.25	2	51
Manne30	2	153	152	2	0.52	2	159
Flood03	3	37	37	6	0.37	2	13
Flood05	3	59	59	4	1.50	3	63
Mang20	1	20	20	100	0.33	2	8
Mang30	1	30	30	100	0.98	2	8
Afiro	1	28	32	10	0.17	2	68
Adlittle	1	57	97	8	17.75	10	476
Share2b	1	97	79	10	37.51	24	807
Israel	1	175	142	10	128.63	5	974

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, its conversion to an equivalent single criterion problem, as well as the possibility of effectively solving badly conditioned problems that might be difficult for other systems.

As an illustration of HYBRID performance on a mainframe computer, a 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 sect. 5.1) for 20 time periods has been solved by both MINOS ver. 5.0 (Murtagh and Saunders, 1983) and HYBRID ver. 3.1. The test has been performed on VAX 780/11 under Berkeley UNIX 4.2. A multicriteria problem with criteria presented in sect. 5.1 has been generated and 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 2.28 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 2.35 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 Conclusions

First version of HYBRID was made operational on VAX 780/11 and is documented in (Makowski and Sosnowski, 1984). Then we had improved and extended the package for dynamic linear programming models (DLP) and for multicriteria problems (both static and dynamic). The later version is documented in (Makowski and Sosnowski, 1985b). The next version, HYBRID 3.03 (described in Makowski and Sosnowski, 1988b) allowed for more general formulation of problems with recognized structure. The code of HYBRID 3.03 has been improved with taking into account robustness of its usage. Last major revision of the algorithm and code which resulted in HYBRID version 3.1 is summarized in Section 1.3.

HYBRID 3.1 is still a prototype software that requires more testing. It is true that for some problems HYBRID 3.1 performs worse than the commercial packages FMPS and MINOS. 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. This is not only due to the fact that MPS file is processed only once in a first run but mainly because in consecutive runs only updates of affected coefficients are made (the problem is generated only for the first run) and because a solution is usually obtained much faster then for the first run. On the other hand HYBRID offers the possibility of formulation, solution and analysis of a linear programming multicriteria problems and of single criteria linear-quadratic problems.

HYBRID provides very useful diagnostics for any LP model and therefore is also

useful for a model 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. Further modification of the way in which the user communicates with the package. The modification will exploit capabilities of PC and will 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). Another new option will allow for easy formulation of piece–wise linear goal function for an LP problem.

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 a faster execution.

We hope that, despite the reservations outlined above, HYBRID 3.1 will eventually be a useful tool with many practical applications. We would be grateful for any criticisms and comments that would help us to improve the package.

7 References

Bertsekas, D.P. (1976). Multiplier methods: a survey. *Automatica*, 12: 133–145.

Curtis, A.R. and J.K. Reid (1972). On the automatic scaling of matrices for Gaussian elimination. *Journal of Mathematics and its applications*, No. 10, pp. 118–124.

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

Fourer, R. (1982). Solving staircase linear programs by the simplex method. *Mathematical Programming*, 23(1982) 274–313, 25(1983) 251–292.

Gay, D.M. (1986). Electronic Mail Distribution of Linear Programming Test Problems. *Numerical Analysis Manuscript*, 86-0(1986), AT&T Laboratories, Murray Hill, New Jersey.

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

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

Hestenes, M.R. (1980). Conjugate Gradient Methods in Optimization. Springer Verlag, Berlin.

Hurlimann, T. (1988). Reference manual for the LPL Modeling Language. Research Report, University of Fribourg, Fribourg, Switzerland.

Ho, J.K. and A.S. Hanne (1974). Nested decomposition for dynamic models. *Mathematical Programming* , 6(1974) 121–140

Kallio, M., A. Lewandowski and W. Orchard-Hays (1980). An implementation of the reference point approach for multiobjective optimization. WP-80-35, International Institute for Applied Systems Analysis, Laxenburg, Austria.

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

Lewandowski, A. and M. Grauer (1982). 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.

Makowski, M. and J. Sosnowski (1981). 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.

Makowski, M. and J. Sosnowski (1984). Hybrid: A mathematical programming package, IIASA, CP-84-9.

Makowski, M. and J. Sosnowski (1985a). A decision support system for planning and controlling agricultural production with a decentralized management structure. In M. Grauer, M. Thompson, A.P. Wierzbicki, editors: Plural Rationality and Interactive Decision Processes, Springer Verlag.

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

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

Makowski, M. and J. Sosnowski (1988a). A Mathematical Programming Package for Multicriteria Dynamic Linear Problems HYBRID. Methodological and User Guide to Version 3.03, WP-88-002, IIASA, Laxenburg, January 1988.

Makowski, M. and J. Sosnowski (1988b). User Guide to a Mathematical Programming Package for Multicriteria Dynamic Linear Problems HYBRID Version 3.1, WP-88-111, IIASA, Laxenburg, December 1988.

Mangasarian, O.L. (1981). Iterative solution of linear programs. *SIAM Journal for Numerical Analysis*, 18(4): 606–614.

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

Murtagh, B.A. and M.A. Sanders (1977). MINOS – A large-scale nonlinear programming system (for problems with linear constraints). User guide. Technical Report, Systems Optimization Laboratory, Stanford University.

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

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

O'Leary, D.P. (1980). A generalized conjugate gradient algorithm for solving a class of quadratic problems. *Linear Algebra and its Applications*, 34: 371–399.

Polyak, B.T. (1969). The conjugate gradient method in extremal problems. *Computational Mathematics and Mathematical Physics*, 9: 94–112.

Polyak, B.T. and N.V. Tretiyakov (1972). An iterative method for linear programming and its economic interpretation. *Economic and Mathematical Methods*, 8: 740–751, (in Russian).

Propoi, A. (1976). Problems of Dynamic Linear Programming, IIASA, RM-76-78.

Sosnowski, J.S. (1978). Dynamic optimization of multisectorial linear production model. Systems Research Institute, Warsaw, Ph.D. Thesis, (in Polish).

Sosnowski, J.S. (1981). Linear programming via augmented Lagrangian and conjugate gradient methods. In S. Walukiewicz and A.P. Wierzbicki, editors: *Methods of Mathematical Programming*, Proceedings of a 1977 Conference in Zakopane. Polish Scientific Publishers, Warsaw.

Tomlin, J.A. (1972). On scaling linear programming problems. *Mathematical Programming Study 4*, North Holland Publishing Company, Amsterdam.

Wierzbicki, A. (1978). On the use of penalty functions in multiobjective optimization, Institute of Automatics, Technical University of Warsaw.

Wierzbicki, A.P. (1979). A methodological guide to multiobjective decision making, WP-79-122, IIASA.

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