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

Probabilistic Constrained Programming: A Reduced Gradient Algorithm Implemented on PC

Mayer, J.

IIASA Working Paper

WP-88-039

May 1988

WORKING PAPER

PROBABILISTIC CONSTRAINED PROGRAMMING: A REDUCED GRADIENT ALGORITHM IMPLEMENTED ON PC

János Mayer

May 1988
WP-88-39

IIASA
International Institute
for Applied Systems Analysis

PROBABILISTIC CONSTRAINED PROGRAMMING: A REDUCED GRADIENT ALGORITHM IMPLEMENTED ON PC

János Mayer

May 1988
WP-88-39

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

FOREWORD

The described solution technique for stochastic linear programs with one joint probability constraint and with multinormal righthand sides was developed and implemented in the frame of the IIASA contracted study "Modelling of interconnected power systems". Very cautious use of efficient subroutines made it possible to solve such a numerically complicated optimization problem on IBM/PC-XT or AT compatibles.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

CONTENTS

PROBABILISTIC CONSTRAINED PROGRAMMING: A REDUCED GRADIENT ALGORITHM IMPLEMENTED ON PC

János Mayer

Computer and Automation Institute
Hungarian Academy of Sciences

1. INTRODUCTION

The main purpose of this paper is to present a reduced-gradient-type algorithm and its implementation on IBM/PC for the solution of probabilistic constrained stochastic programming problems. It is assumed that the random right-hand side has normal joint probability distribution.

The numerical difficulties associated with the solution of these problems originate mainly in the computation of the stochastic constraint function and its gradient vector. The evaluation of multiple integrals is required here, which is very time-consuming and introduces a numerical error of several magnitudes higher when compared with the other parts of the model.

The chief design-objectives for the algorithm and program have been the following: The structure of the algorithm should facilitate the usage of the powerful advanced LP-tools for the handling of linear constraints, and the number of calls to the procedure which computes the probabilistic constraint should be reduced as much as possible.

The program-system has been written in FORTRAN 77 with the exception of a software-toolkit written in assembler to extend the capabilities of FORTRAN with screen-handling facilities.

In the next section the model is formulated and a short historical summary is given of its origins and of the numerical efforts for solving it. Section 3 serves for the presentation of the algorithm. The subsequent section gives an overview of the program-system, followed by a section acting as a brief User's Guide. Finally some test results on problems taken from the literature are presented.

Acknowledgement: The author expresses his thanks to the following persons for making available their program-systems: Dr. I. Maros for his LP-system MILP, Dr. T. Szántai for his subroutine-package NORSUBS, and Mr. L. Sparing for his assembler-toolkit CGA3.

2. PROBLEM FORMULATION

The probabilistic constrained stochastic programming problem can be formulated as follows:

$$\begin{aligned} &\max f(x) \\ &G(x) \geq \alpha \ , \\ &g_i(x) \geq 0, \ i = 1, \cdots m \ , \end{aligned} \tag{2.1}$$

where $G(x) = P(h_1(x, \beta) \geq 0, \ldots, h_p(x, \beta) \geq 0)$, and $\beta_1, \cdots \beta_q$ are random variables with a joint probability density function of the form:

$$l(t) = e^{-Q(t)}, \ t \in \mathbf{R}^q \ , \tag{2.2}$$

$Q(t)$ being a convex function. Functions $f(x)$, $g_1(x), \ldots, g_m(x)$ and $h_1(x, t), \ldots, h_p(x, t)$ are assumed to be concave functions, $x \in \mathbf{R}^n$ and α is a prescribed probability level. Symbol "P" means probability.

The problem was formulated by A. Prékopa [13], [14] in this general form. Using his fundamental theorem concerning logarithmic concave measures [12] he proved that $G(x)$ is a logarithmic concave function which implies that (2.1) is a convex programming problem [13].

The specific numerical difficulties associated with this type of nonlinear programming problems are in connection with the computation of $G(x)$. Calculating the value of this function means multiple integration which is very time-consuming even for moderately dimensioned t vectors and leads to values with an error usually several magnitudes greater than the numerical errors connected with the computation of the other functions appearing in the model.

The subject of this paper is the solution of the special case of problem (2.1) where $f(x)$, $g_1(x), \ldots, g_m(x)$, $h_1(x, t), \ldots, h_p(x, t)$ are linear functions, and the random variables are normally distributed. Such problems may arise e.g. in the linear programming modeling process, when some components of the right-hand side vector turn out to be random variables. In this respect probabilistic constrained programming can be considered as an

extension of the LP-modeling technique.

The problem under consideration can be formulated as the following nonlinear programming problem:

$$\begin{aligned} &\max\ c^T x\ , \\ &P(Hx \geq \beta) \geq \alpha\ , \\ &Ax = b\ , \\ &x \geq 0\ , \end{aligned} \tag{2.3}$$

where the components of β are random variables with a joint normal probability distribution function, A is an $(m \times n)$ matrix, H an $(q \times n)$ matrix, x and c are n-dimensional and b is an m-dimensional vector, symbol "P" stands for probability. The following denotation will be used in the sequel: $G(x) = P(Hx \geq \beta)$. It will be assumed that for problem (2.3) the Slater-condition holds, i.e. there exists a feasible solution x with the property: $G(x) > \alpha$.

The earliest formulation of models which contain probabilistic constraints can be found in the paper of Charnes and Cooper [1], see also Charnes et al., [2]. They called their models chance constrained problems, the constraints in these models can be considered as a special case of the probabilistic constraint in problem (2.3) by assuming that the random variables $\beta_1, \ldots, \beta_q$ are independent in the stochastic sense.

Model (2.3) in its general form originates in the paper of Prékopa [11]. Being a special case of problem (2.1) the results of Prékopa referenced above imply that (2.3) is a convex programming problem. The particularities are the requirement that the joint probability distribution of the random vector-variable should be normal and the restriction of the concave functions in model (2.1) to the class of linear functions.

From the mathematical programming point of view problem (2.3) is a nonlinear programming problem with a single nonlinear constraint. The first algorithm and program system for the solution of this type of problems were developed by Prékopa and Deák in 1971 [16], they based their method on Zoutendijk's feasible-direction method P2 (for the algorithm see Prékopa [15], for the numerical results Deák [3]). Rapcsák [17] used a SUMT method for the numerical solution of small-scale problems of the type (2.3). Szántai applied Veinott's supporting hyperplane method and achieved excellent numerical results [18], his program system PCSP has been incorporated into the SDS/ADO system, see Edwards [6]. The author of this paper developed an algorithm based on the reduced gradient method and implemented it on a CDC mainframe computer [9]. This method can

be considered as an immediate predecessor of the algorithm implemented on PC.

As mentioned above the computation of $G(x)$ is in itself a very hard numerical problem, solution methods are mainly based on Monte-Carlo techniques. Deák [4] and Szántai [19] accomplished very nice and efficient numerical methods in this field, in our program system Szántai's subroutine-system is used for the computation of $G(x)$.

The specific form of the linear constraints in (2.3) has been selected for the sake of simplicity of presentation of the algorithm. Although LP-problems can obviously be transformed to that form the user of the program system is by no means forced to do this. The underlying LP-problem can be specified according to the standards of LP-packages, (see Section 5), the programmed version of the algorithm corresponding to this formulation of the problem can be derived from the method presented in the next section in a straightforward way.

3. SOLUTION TECHNIQUE

In the design phase of constructing an algorithm for the solution of problem (2.3) two main requirements were considered as desirable goals. According to the first one the structure of the algorithm should facilitate the usage of advanced LP-techniques for the handling of linear constraints and objective. The second one is the quite natural requirement of trying to reduce the number of calls of the procedure which computes $G(x)$.

The algorithm which has been implemented for the nonlinearly constrained problem (2.3) can be classified as a reduced gradient type feasible direction method.

To identify a method belonging to this class of algorithms first of all the direction-finding subproblem is to be specified.

The characteristic feature of reduced gradient algorithms is a subdivision of variables into basic and nonbasic parts. The components of the direction vector corresponding to nonbasic variables are determined in a way to ensure improvement in the objective function value whereas those corresponding to basic variables serve for maintaining feasibility. For notational simplicity in the presentation given below it will be assumed that at the beginning of a certain iteration the first m variables are the basic variables.

Let us assume that at the beginning of the k-th iteration of the method we are given a feasible solution x^k of problem (2.3) together with a positive ϵ^k and a partition $(x^k)^T = (y^T, z^T)$ fulfilling the following conditions:

(i) $y \in \mathbf{R}^m$, and if A is partitioned in a similar way as $A = (B, N)$ then B is an $(m \times m)$ nonsingular matrix.

(ii) $y_j > \epsilon^k$, $j = 1, \ldots, m$, (nondegeneracy assumption).

To condition (i) it is worthwhile to mention that in our case A is always of full row rank, as it will be seen later. The columns of B form a basis in the column space of matrix A, and components of x and w corresponding to B will be called basic components whereas those corresponding to N nonbasic components. Let us suppose that the inverse of the basis-matrix, B^{-1} has already been computed.

Let the direction-vector w^k and the objective row c^T partitioned analogously, $(w^k)^T = (u^T, v^T)$ and $c^T = (g^T, h^T)$.

The direction-finding subproblem of the algorithm can now be formulated as the following mathematical programming problem:

$$\begin{array}{rcrcll} \max & & & & \tau, & \\ g^T u & + & h^T v & \geq & \tau, & \\ \nabla_y^T G(x^k) u & + & \nabla_z^T G(x^k) v & \geq & \theta\tau, & \text{if } G(x^k) \leq \propto + \epsilon^k \ , \\ Bu & + & Nv & = & O, & \\ & & v_j & \geq & 0, & \text{if } z_j \leq \epsilon^k,\ j = 1, \ldots, n - m \ , \\ & & \|v\| & \leq & 1, & \end{array} \tag{3.1}$$

where θ is a positive weighting factor for the directional derivatives. Number θ is held constant in the course of the iterations. Choosing the norm $\|v\| = \max_j |v_j|$ problem (3.1) becomes a linear programming problem.

It is evident that for problem (3.1) there exists an optimal solution, the optimal value of the objective function let be denoted by τ^*.

The direction-finding procedure is the following:

Problem (3.1) is solved first. Depending on the optimal value of the objective function three cases are distinguished:

Case 1 $\tau^* = 0$ holds. Under these circumstances (3.1) is solved again with ϵ^k replaced by 0.

If the optimal objective function value is still found to be equal to zero then the algorithm terminates. It is easy to see that in this case x^k is an optimal solution of problem (2.3).

If the optimal value of the objective function turns out to be positive then the tolerance-reduction procedure described below for Case 2 is initiated.

Case 2 $0 < \tau^* \leq \epsilon^k$ holds. In this situation the tolerance is reduced in a cycle consisting of the following steps:

Step 1 The tolerance is halved, $\epsilon^k := 0.5\,\epsilon^k$.

Step 2 Subproblem (3.1) corresponding to the new ϵ^k is solved. If for the optimal τ^*-value still $\tau^* \leq \epsilon^k$ holds then the cycle is repeated by returning to Step 1, otherwise the direction-finding procedure terminates in the same way as in Case 3.

Case 3 $\tau^* > \epsilon^k$ holds. In this case the w-part of the solution of (3.1) is accepted as a direction-vector for the present iteration, the direction-finding procedure terminates.

It is obvious that the procedure just described terminates in a finite number of steps. It either indicates that x^k is an optimal solution of problem (2.3) or furnishes a feasible direction along which the objective function strictly increases.

The selection of (3.1) as the direction-finding subproblem is justified by its easy solvability. In fact (3.1) is equivalent to the problem given below.

$$\begin{array}{lrll} \max & & \tau, & \\ & r^T v & \geq & \tau, \\ & s^T v & \geq & \theta\tau, \quad \text{if } G(x^k) \leq \alpha + \epsilon^k \ , \\ & v_j & \geq & 0, \quad \text{if } z_j \leq \epsilon^k,\ j = 1,\ldots,n-m \ , \\ & \|v\| & \leq & 1, \end{array} \tag{3.2}$$

where

$$\begin{aligned} r^T &= h^T - g^T B^{-1} N \ , \\ s^T &= \nabla_z^T G(x^k) - \nabla_y^T G(x^k) B^{-1} N \ . \end{aligned} \tag{3.3}$$

Having obtained the solution v^* of problem (3.2) the u-part of the solution of (3.1) can be computed as follows:

$$u^* = - B^{-1} N v^* \tag{3.4}$$

Vectors r and s are the reduced gradients of the objective function and the nonlinear constraint function, respectively.

If among the components of z there exists at least one fulfilling the conditions $z_j > \epsilon^k$ and either $r_j \neq 0$ or $s_j \neq 0$ then the reduction can be carried out one step further by eliminating one of the rows in (3.2). This leads to a subproblem with the very attractive feature of explicit solvability like in the reduced gradient method of Wolfe [20].

In the implemented algorithm first an attempt is made to carry out this second reduction.

If the attempt was successful then the explicit solution of (3.2) with Euclidean norm is computed. If the reduction is unachievable then (3.2) is solved with a normalization using the maximum-norm. This means the solution of a linear continuous knapsack problem which requires essentially only a sorting procedure.

Having solved the direction-finding subproblem the algorithm proceeds by performing a standard line-search. To determine the stepsize here simply means to compute the intersection of the ray $x(\lambda) = x^k + \lambda w^k$, $\lambda \geq 0$ with the boundary of the feasible domain.

Let the stepsize be λ^k then the new feasible solution serving as a starting point for the next iteration will be $x^{k+1} = x^k + \lambda^k w^k$.

If the nondegeneracy assumption (ii) remains fulfilled also for x^{k+1} then the subdivision into basic and nonbasic components remains unchanged, $\epsilon^{k+1} = \epsilon^k$ and a new iteration begins.

If degeneracy occurs then the subdivision is changed. This is accomplished by a procedure consisting of the following steps.

Step 1 An attempt is made using a greedy strategy to exchange all basic variables with $x_j^{k+1} \leq \epsilon^k$ for nonbasic variables with $x_i^{k+1} > \epsilon^k$ in order to get a subdivision of x^{k+1} which fulfills (i) and (ii) as specified above. The inverse of the basis-matrix is updated accordingly by standard simplex pivoting techniques. In the case of a success a new iteration is started with the new subdivision, otherwise the tolerance reduced in Step2.

Step 2 The tolerance is reduced: $\epsilon^k := 0.5\,\epsilon^k$, return to Step 1.

If there exists a subdivision of the components of x^{k+1} into basic and nonbasic parts such that all basic components are positive then the procedure outlined above obviously finds a partition fulfilling (i) and (ii) in a finite number of steps. If this is not the case then it may still happen that the solution of the direction-finding problem (3.1) is a feasible direction for problem (2.3). If so then the next iteration is started, otherwise some special degeneracy-handling sub -procedure is to be initiated. This means the application of a finite pivoting scheme for a modified direction finding problem (3.1) where nonnegativity is also required for those components of vector u which correspond to zeros in the current feasible solution. Such a procedure has not been implemented in the present version of the program system because the inclusion of it would increase the size of the code substantially, and on the other hand test problems available in the literature do not necessitate it. This feature will be included in versions for the solution of large-scale prob-

lems of the type (2.3).

To give a more complete description of the algorithm it is to be explained how the procedure outlined so far gets started. A starting feasible solution to (2.3) is found in two phases which makes the whole procedure a three-phase method.

Phase 0: The linear programming problem

$$\begin{aligned} \max\ & c^T x, \\ & Hx \geq E(\beta)\ , \\ & Ax = b\ , \\ & x \geq O \end{aligned} \tag{3.5}$$

is solved, where $E(\beta)$ denotes a vector whose i-th component is the expectation $E(\beta_j)$, $j = 1, \cdots q$. To solve (3.5) that kind of the simplex method is used, where the first phase is started by augmenting the problem with logical variables thus ensuring full row rank. In the subsequent two phases of the algorithm for the solution of (2.3) these logical variables are kept, i.e. the algorithm works on an equivalent form of (2.3) which guarantees full row rank throughout.

Phase I: In this phase a feasible solution of (2.3) is computed by applying the reduced gradient method of Wolfe [20] to the following linearly constrained nonlinear programming problem:

$$\begin{aligned} \max\ & G(x), \\ & Ax = b\ , \\ & x \geq 0\ . \end{aligned} \tag{3.6}$$

To ensure convergence the original reduced gradient method is enhanced by an ϵ^k-technique in the same way as the main algorithm discussed before. The starting feasible solution for the method will be the optimal solution of (3.5) and the starting basis the optimal basis, both obtained in Phase 0. The case of a degenerate basis is handled by the procedure summarized for the main algorithm. Iterations of the reduced gradient method are carried out till $G(x^k) > \propto$ is accomplished. The Slater-regularity guarantees that the algorithm in Phase I terminates in a finite number of iterations. It must be emphasized that for large-scale problems or for problems where for the Slater-points $G(x) \approx \propto$ holds the Phase I algorithm needs further improvements. More advanced forms of the reduced gradient method are to be used which are based on conjugate gradients and quasi-Newton techniques, like the algorithm implemented in the MINOS system [10]. These techniques are not implemented in the present version of the program system. The reason for this is the fact that in the scope of problems to be solved by the system no difficulties of the above mentioned nature arise.

Phase II: In this phase the original problem (2.3) is solved using the main procedure described above. The starting point and starting basis are supplied by the Phase I algorithm. The sequence of iterations is stopped when either an optimal solution is indicated by the direction-finding procedure or the tolerance ϵ^k is decreased below a prescribed stopping level.

In the remaining part of this section a brief summary of numerical techniques which were implemented in the program-system is given.

(i) Phase 0: An advanced version of the simplex method, developed by I. Maros [8] is used.

(ii) Computation of the reduced gradients and the basic components of the direction vector in Phases I and II (see (3.3) and (3.4)): LP forward and backward transformations.

(iii) Solution of the direction-finding subproblems in Phases I and II: Either achieved via explicit formulas or in Phase II by a standard sorting procedure to solve a continuous knapsack problem.

(iv) Determining the steplength in Phase I: Golden-section search (see [7]).

(v) Determining the steplength in Phase II: The intersection of the ray corresponding to the direction w with the surface $\{x|G(x) = \alpha\}$ is computed by a bisection-procedure.

(vi) Changing the basis if necessitated by degeneracy, Phases I and II: LP pivoting technique.

(vii) Reinversion of the basis: LP-technique.

(viii) Computation of $G(x)$ and $\nabla G(x)$: The Monte-Carlo technique [19] developed by T. Szántai is used. This method includes the possibility of getting lower and upper bounds for $G(x)$ on a very low computational cost (i.e. very quickly on the computer), which can be utilized to reduce the number of calls to the procedure which computes $G(x)$. In fact golden-section search and bisection ((iv) and (v) above) are both algorithms where values of the function G, computed at different points, are compared for the sake of reducing the interval of search. First the bounds on $G(x)$ are computed and an attempt is made in an obvious way to reduce the interval. The value of $G(x)$ is only computed if this trial was unsuccessful.

4. OVERVIEW OF THE PROGRAM SYSTEM

The program system is designed for IBM/PC XT and AT computers.

Programs are written in FORTRAN 77 language with the exception of a software-toolkit written in assembler which extends the capabilities of FORTRAN by screen-handling facilities.

The system consists of the following main parts:

(i) User interface: This part provides the user with an interactive facility for problem-specification and run-time interrupts.

(ii) Nonlinear programming part: It realizes Phases I and II of the procedure.

(iii) Linear programming system: This part accomplishes Phase 0 of the procedure and several sub-procedures of it are also used in a modular way (see (i), (ii), (vi), (vii) at the end of the previous section). The advanced linear programming system MILP developed by I. Maros is incorporated into the system to perform these activities.

(iv) Computation of bounds on the value of the multivariate normal distribution function, computation of the function itself and its gradient vector. The subroutine-package NORSUBS of T. Szántai was built in to accomplish this.

(v) A toolkit of assembly routines to facilitate screen-handling in FORTRAN programs. The program CGA3 developed by L. Sparing (Technical University Budapest) is utilized.

The main program, the subroutines performing the control of the iterative procedures, and MILP communicate arrays to each other through labeled COMMON fields. In the case of other subroutines and the individual modules of MILP, data communication is mainly organized with the help of parameter-lists.

5. A BRIEF USER'S GUIDE

The prerequisite of the usage of the system is the existence of an ASCII-file consisting of 80-character records which contains the data specifying the LP problem (3.5). The data are to be given in the input format of the LP-system MILP, which means the following file-structure (the required FORTRAN input format is also specified):

The first record contains a problem-name (A10). The next record contains the dimensions: the number of rows (m), the number of columns (n) and the index of the objective-function row: (3I5).

The next records contain codes for the type of the constraints, (10I5), where the codes have the following meaning:

0 --> "=",

1 --> "$\leq$",

2 --> "$\geq$",

3 --> free, meaning a non-restrictive constraint or objective function row.

The right-hand-side data are coming next, beginning with a new record (5F15.5).

The next records would contain data corresponding to ranges, a feature not included in the nonlinear system. The user is kindly asked to insert at this place a sufficient number of blank records corresponding to m real numbers in (5F15.5) format.

In the following records column-type information is specified, by repeating n-times the following structure:

Column-header record: index of the column, number of nonzero entries in this column, lower bound on the variable, upper bound on the variable (2I5,2F10.5).

Records of the nonzero entries: They must be given as a sequence of pairs, on the first place of which the row index and on the second place the entry is put (5(I5,F10.5)).

After starting a run the user will be asked by the system to specify the following items (in this order):

--> Identifier of the LP input-file.

--> Indices of rows to be included into the probabilistic constraint.

--> Standard deviations of the right-hand-side values corresponding to the rows selected.

--> Correlation matrix.

--> Required probability level.

--> How many random numbers are to be generated for the computation of $G(x)$ and $\nabla G(x)$, see [19].

--> Output level.

Subsequently to data specification the system starts the solution process. The following run-time informations are displayed in a scroll-area of the screen:

--> Probability-level.

--> Present value of the objective function.

--> Present value of counters for the following activities:

NOBJ : Objective function calls.

NGROBJ : Objective function gradient calls.

NPROB : Calls for the computation of $G(x)$.

NGPROB : Calls for the computation of $\nabla G(x)$.

NFTR : LP forward transformations.

NBTR : LP backward transformations.

NTRSF : LP pivot-transformations.

NINV : Reinversion of the basis-matrix.

If the more detailed output option was selected the following quantities are also displayed:

--> ACTIVITY =	-1 if the stochastic constraint is considered to be inactive by the system, 1 if it is considered as active.
--> REDNOR	: Euclidean norm of the reduced gradient.
--> EPZIG	: Present value of the tolerance ϵ^k.
--> DIRDER FOR OBJ	: The scalar-product of the direction vector and the gradient of the objective.
--> DIRDER FOR STCH	: The scalar-product of the direction vector and $\nabla G(x)$.
-->	The feasible steplength determined by the linear constraints.
-->	The feasible steplength determined by all of the constraints.
-->	The optimal steplength.

Two run-time user interrupts are available:

--> Pressing the < # > key terminates the run after completing the present iteration.

--> Pressing the < escape > key enables changing the value of the following parameters of the algorithm:

EPZIG : Present value of ϵ^k.

THETA : The θ-value in (3.1).

TOLDJ : Relative tolerance for gradient-components to be considered as zeros.

EPSTP : If the Euclidean norm of the reduced gradient decreases below this level then the present solution is checked for optimality, if it is not accepted as optimal EPZIG is reduced.

EPLN : Tolerance for linesearch in Phase I.

ESTCH : Tolerance for bisection in Phase II.

The results are stored in two output files:

--> NLOG.RES : It contains the information which was displayed on the screen during run-time.

--> NLSP.RES : This file contains a detailed listing of the results.

After a short summary of the problem specification the information concerning the optimal values of the variables follows, containing columnwise the following items:

-- Index of the variable, logicals appear first.

-- One-character codes with the following meaning:

"B" basic variable,

"L" nonbasic variable on lower bound,

"U" nonbasic variable on upper bound,

"F" nonbasic variable with a value strictly between bounds,

"S" structural variable appearing in the probabilistic constraint.

-- Lower bound for the variable.

-- Optimal value of the variable.

-- Upper bound for the variable.

-- Component of the last direction vector.

This part of the output is followed by a listing of row-type information. For each row the following information can be found in this tableau:

-- Index of the row.

-- Scalar product of the corresponding row of A and the solution vector,

-- optimal value of the logical variable,

-- substitution value without right-hand-side,

-- right-hand-side component,

-- absolute error for the fulfillment of the corresponding constraint.

6. SOME TEST RESULTS

The program has been tested on two probabilistic constrained problems taken from the literature.

Problem 1. This is a model in water resources system planning [5].

$$\max\ x_1,$$

$$P\left(\begin{array}{rrr} x_3 & & \geq \beta_1, \\ & x_4 & \geq \beta_2, \\ & x_5 & \geq \beta_3. \end{array}\right) \geq 0.9,$$

$$\begin{array}{rrrrrrrrrrl}
 & & x_2 & + & x_3 & & & & & & \leq 118.348, \\
 & & x_2 & + & x_3 & + & x_4 & & & & \leq 163.776, \\
 & & x_2 & + & x_3 & + & x_4 & + & x_5 & & \leq 187.197, \\
 & x_1 & + & x_2 & & & & & & & \geq 374.786, \\
 & x_1 & + & x_2 & + & x_3 & & & & & \geq 454.772, \\
 & x_1 & + & x_2 & + & x_3 & + & x_4 & & & \geq 516.052, \\
 & x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & \geq 582.083, \\
0 \leq & x_1 & & & & & & & & & \leq 400.0, \\
0 \leq & & x_2 & & & & & & & & \leq 64.219, \\
0 \leq & & & & x_3 & & & & & & \leq 252.0, \\
0 \leq & & & & & & x_4 & & & & \leq 252.0, \\
0 \leq & & & & & & & & x_5 & & \leq 252.0,
\end{array}$$

where

$$E(\beta_1) = 32.9,\ E(\beta_2) = 40.07,\ E(\beta_3) = 23.35,$$
$$D(\beta_1) = 8.61,\ D(\beta_2) = 10.65,\ D(\beta_3) = 6.0,$$

"D" denoting here standard deviation ,and the correlation matrix is

$$R = \begin{bmatrix} 1. & .36 & .125 \\ .36 & 1. & .571 \\ .125 & .571 & 1. \end{bmatrix}$$

Denoting the variables in [5] by $\hat{x}_1, \ldots, \hat{x}_5$, the relation between the variables above and the variables in [5] are the following:

$$\hat{x}_1 = x_1 + 100.0,\ \hat{x}_2 = x_2 + 38.1,\ \hat{x}_3 = x_3,\ \hat{x}_4 = x_4,\ \hat{x}_5 = x_5 \ .$$

The results are presented in Table 1, where the first row contains the solution of the underlying LP-problem whereas the second the results for Problem 1.

TABLE 1 Results for Problem 1

x_1	x_2	x_3	x_4	x_5
394.89	0.0	59.886	103.88	23.431
394.89	0.0	61.596	82.378	43.223

Problem 2. This is a planning-model [16], [18] for the electric energy sector of the Hungarian economy, with 53 rows and 46 variables.

The computations were carried out for a probability level 0.9, the optimal value of the objective was 4370.18. The components of the solution are given in Table 2 in the same manner as the results in Table 1.

TABLE 2 Results for Problem 2

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
6118.4	12739.	4640.0	8098.8	0.	4562.1	42.892	4730.9	.675
6441.8	12808.	4640.0	8168.5	0.	4460.0	40.0	4563.1	.68741

x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}
.168	0.	.260	.156	9134.0	568.0	1327.0	498.33	0.
.12373	0.	.24237	.28678	9134.0	568.0	1327.0	498.56	0.

x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}	x_{26}	x_{27}
8814.7	0.	1112.8	1950.0	487.2	18400.	1932.5	25.555	8077.6
9095.2	0.	1075.4	1940.8	514.1	18400.	1889.3	23.832	8077.6

x_{28}	x_{29}	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}
965.73	241.43	56969.	2563.6	1424.2	993.6	110.4	93.676	4468.8
966.64	241.66	57025.	2566.1	1425.6	993.6	110.4	93.764	4464.0

x_{37}	x_{38}	x_{39}	x_{40}	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}
518.33	116.77	3224.3	1958.1	568.33	1992.4	2038.4	3061.4	3098.9
518.56	115.86	3224.2	1913.1	731.03	2026.5	2344.1	3077.7	3012.6

x_{46}
0.
0.

Computing times on an IBM/PC AT without arithmetical coprocessor were ≈ 1 minute for Problem 1 and ≈ 20 minutes for Problem 2.

It must be emphasized that the program is still in a further-development phase with the general aims to reduce computational time on the PC and increase reliability of the system.

REFERENCES

[1] Charnes, A. and W.W. Cooper: "Chance constrained programming", Management Science, (1959) pp.73–79.

[2] Charnes, A., W.W. Cooper and M.J.L. Kirby: "Chance constrained programming: An extension of statistical method", in.: Optimizing Methods in Statistics, ed.: Rustagi, J. S., Academic Press, (1971), pp.391–402.

[3] Deák, I.:"Computational experiences with a stochastic programming model", Magyar Tudományos Akadémia Számitástechnikai Központja, Közlemények, 9(1972) pp.33–49. (in Hungarian)

[4] Deák, I.: "Computing probabilities of rectangles in case of multinormal distributions", J. Statist. Comp. and Simulation, 26(1986), pp.101–114.

[5] Dupačová, J., A. Gaivoronski, Z. Kos and T. Szántai: "Stochastic programming in water resources planning: A case study and a comparison of solution techniques", WP-86-40, IIASA, Laxenburg.

[6] Edwards, J. (ed.): "Documentation for the ADO/SDS collection of stochastic programming codes", WP-85-02, IIASA, Laxenburg.

[7] Gill, P.E., W. Murray, and M.H. Wright: "Practical optimization", Academic Press, 1981.

[8] Maros, I.: "A general phase-I method in linear programming", European Journal of Operational Research, 23(1986), pp.64–77.

[9] Mayer, J.: "A nonlinear programming method for the solution of a stochastic programming model of A. Prékopa", in: Survey of Mathematical Programming, Proceedings of the 9th Mathematical Programming Symposium, ed.: A. Prékopa, Vol 2. North-Holland Publ. Co., (1979) pp.129–139.

[10] Murtagh, B.A. and M.A. Saunders: "Large-scale linearly constrained optimization", Mathematical Programming, 14(1978), pp.41–72.

[11] Prékopa, A.: "On probabilistic constrained programming", in: Proceedings of the Princeton Symposium on Mathematical Programming, Princeton University Press, New York, (1970) pp.113–138.

[12] Prékopa, A.: "Logarithmic concave measures with applications to stochastic programming", Acta Sci. Math. (Szeged) 32(1971) pp.301–316.

[13] Prékopa, A.: "A class of stochastic programming decision problems", Mathematische Operationsforschung und Statistik, 3 (1972) pp.349–354.

[14] Prékopa, A.: "Contributions to stochastic programming", Mathematical Programming 4(1973), pp.202–221.

[15] Prékopa, A.: "Eine Erweiterung der sogenannten Methode der zulässigen Richtungen der nichtlinearen Optimierung auf den Fall quasikonkaver Restriktionen", Mathematische Operationsforschung und Statistik, 5(1974), pp.281–293.

[16] Prékopa, A., S. Ganczer, I. Deák and K. Patyi: "The STABIL stochastic programming model and its experimental application to the electricity production in Hungary", in: Stochastic Programming, Proceedings of the International Conference on Stochastic Programming, ed. M.A.H. Dempster, Academic Press,(1980), pp.63–82.

[17] Rapcsák, T.: "On the numerical solution of a reservoir model", doctoral dissertation, University of Debrecen, Hungary (1974). (in Hungarian).

[18] Szántai, T.: "A Prékopa-fele STABIL sztochasztikus programozasi modell numerikus megoldasarol", Alkalmazott Matematikai Lapok, 2(1976) pp.93–101.(in Hungarian)

[19] Szántai, T.: "Calculation of the multivariate probability distribution function values and their gradients", WP-87-82, IIASA, Laxenburg.

[20] Wolfe, P.: "Methods of nonlinear programming", in Recent Results in Mathematical Programming, ed. Graves, R.L., McGraw-Hill, (1963) pp.76–77.