Control
and Cybernetics

VOL. 17 (1988) No. 4

Simplex modifications exploiting special features of dynamic and stochastic dynamic linear programming problems

by

JACEK GONDZIO

Systems Research Institute
Polish Academy of Sciences
Newelska 6
01-447 Warszawa

This paper considers dynamic and stochastic dynamic linear programming problems. Such problems lead to large scale linear programs with special structures: staircase and angular-staircase for dynamic problems, respectively. Modifications are given that exploit these structures for both accelerating the simplex method and decreasing its storage requirements. The method has been implemented in an experimental code. Numerical experience is reported.

KEY WORDS: Large-scale linear programming, Supersparsity, Special structures, Pricing techniques.

1. Introduction

This paper presents modifications of the simplex method for dynamic and multistage stochastic linear programs. The following specializations of the simplex method for such problems are possible:

(a) exploiting supersparsity for the problem data storage;

(b) automatically constructing an advanced basis;

(c) modifying pricing techniques;

(d) exploiting the structure for a more compact basis inverse representation.

Points (a), (b) and (c) are discussed in this paper.

We consider the following form of dynamic linear programming problem

$$\text{maximize} \qquad z \equiv \sum_{i=0}^{T} q_{xi}^T x_i + \sum_{i=0}^{T-1} q_{ui}^T u_i, \tag{1.1}$$

over all $x_i \in R^n$, $i = 0, 1, ..., T$, and $u_i \in R^m$, $i = 0, 1, ..., T-1$, satisfying

$$-Gx_i - Ku_i + x_{i+1} = g_i, \quad i = 0, 1, ..., T-1, \tag{1.2a}$$

$$Cx_i + Du_i \leqslant h_i, \quad i = 0, 1, ..., T-1, \tag{1.2b}$$

$$Ex_0 \leqslant g_T, \tag{1.2c}$$

$$Fx_T \leqslant h_T, \tag{1.2d}$$

$$l_{xi} \leqslant x_i \leqslant L_{xi}, \quad i = 0, 1, ..., T, \tag{1.2e}$$

$$l_{ui} \leqslant u_i \leqslant L_{ui}, \quad i = 0, 1, ..., T-1, \tag{1.2f}$$

where x_i and u_i denote state and control variables at moment i, respectively, $g_i \in R^n$, $h_i \in R^k$, $i = 0, 1, ..., T-1$, $g_T \in R^{k_e}$, $h_T \in R^{k_f}$, where k denotes the number of constraints of type (1.2b) and k_e and k_f denote the numbers of constraints of type (1.2c) and (1.2d), respectively.

Matrices G, K, C, D, E, F have the appropriate dimensions.

The multistage stochastic linear programming problem under consideration has the following form

$$\text{maximize} \quad z \equiv q_{x0}^T x_0 + q_{u0}^T u_0 + \\ + E\left\{q_{x1}^T x_1 + q_{u1}^T u_1 + E\left\{... + E\left\{q_{xT}^T x_T\right\}...\right\}\right\}, \tag{1.3}$$

over all $x_i \in R^n$, $i = 0, 1, ..., T$, and $u_i \in R^m$, $i = 0, 1, ..., T-1$, satisfying

$$-Gx_i^j - Ku_i^j + x_{i+1}^l = b_{i+1}^{jl}, \; i = 0, 1, ..., T-1, \; r_i = \prod_{\alpha=1}^{i} k_\alpha, \\ j = 1, 2, ..., r_i, \; l = 1, 2, ..., k_{i+1}, \tag{1.4a}$$

$$l_{xi} \leqslant x_i^j \leqslant L_{xi}, \; i = 0, 1, ..., T, \; j = 1, 2, ..., r_i, \tag{1.4b}$$

$$l_{ui} \leqslant u_i^j \leqslant L_{ui}, \; i = 0, 1, ..., T-1, \; j = 1, 2, ..., r_i, \tag{1.4c}$$

here x_i^j and u_i^j denote j-th realization of state and control variables at moment i, respectively. Let us denote by k_{i+1}, $i = 0, 1, ..., T-1$, the number of the realizations of the right hand side of equation (1.4a), by p_{i+1}^l the probability of the event $\{b_{i+1} = b_{i+1}^l\}$. We assume discrete distributions (finite supports) of random elements b_{i+1}. For simplicity all randomness of the problem was concentrated in the right hand side vector.

Problems of type (1.1), (1.2) or (1.3), (1.4) arise often in economic planning.

Each of the above programs could be formulated as a linear programming problem and solved by the simplex method. They lead to large-scale problems with special structures: staircase and angular-staircase, respectively.

Such problems (especially the first one) have received much attention (see e.g. [2], [3], [4], [5], [7], [8], [12], [13], [14], [15]), mostly due to their economic applications.

Several atractive solution methods have been proposed for staircase structured linear programs [2], [12], [13]. Their advantages are discussed in [14], but they are not more efficient than the specialized simplex method of [4] and [5], in which various algebraic properties of staircase problems are exploited.

We present in this paper simplex modifications that exploit both algebraic properties and the properties derived by interpreting the problem as one of optimal control of a discrete dynamic system.

Since the stochastic dynamic problem (1.3) and (1.4) is a natural generalization of the dynamic linear programming problem (1.1) and (1.2) our techniques found efficient for the latter one were then extended to the former one, giving even relatively better results.

The modifications described in this paper give the possibility to solve problems (1.1), (1.2) and (1.3), (1.4) in, roughly speaking, about half the time recquired by the clasical simplex method.

In section 2 we shall formulate the problems considered as linear programming problems, analyse their special structures and show their supersparsity. In section 3 we summarize the well-known revised simplex method. In section 4 we further analyse the special structures of our linear programs and, interpreting the problems discussed as those of optimal control, we derive some simplex modifications. Section 5 describes our numerical experience and section 6 gives our conclusions.

2. Specially structured linear problems

Problem (1.1), (1.2) can be formulated as the linear program

$$\text{maximize} \qquad z \equiv c_1^T y, \tag{2.1}$$

$$\text{subject to} \qquad A_1 y = b_1, \tag{2.2}$$

$$\underline{y} \leqslant y \leqslant \bar{y}, \tag{2.3}$$

where $y = (x_0, u_0, s_0, x_1, u_1, s_1, \ldots, x_{T-1}, u_{T-1}, s_{T-1}, x_T, s_T)$, and $s_0, s_1, \ldots, s_{T-1}$ denote slack (surplus, free or artificial) variables associated with constraints (1.2a), (1.2b), and s_T denotes slack variables for constraints (1.2c), (1.2d). Vectors $\underline{y}$ and $\bar{y}$ are generated from constraints (1.2e), (1.2f) and the type of the slacks.

Matrix A_1 and vector b_1 have the form shown in the next page. In phis form denotes the identity matrix of dimension $n \times n$, J_l^i denotes a diagonal matrix of dimension $I \times I$ with elements on the diagonal equal to $+1$ or -1 (the sign depends on the direction of the inequality in the appropriate constraint).

Problem (2.1)–(2.3) has $N_1 = (T+1)\,n + Tm$ structural variables and $M_1 = T(n+k) + k_e + k_f$ constraints (and slack variables).

x_0	u_0	s_0	x_1	u_1	s_1	$x_2 \ldots x_{T-1}$	u_{T-1}	s_{T-1}	x_T	s_T	
$-G$	$-K$	J^0_{n+k}	I_n								g_0
C	D										h_0
			$-G$	$-K$	J^1_{n+k}	I_n					g_1
			C	D							h_1
						$\ddots$					$\vdots$
							$-G$	$-K$	J^{T-1}_{n+k}	I_n	g_{T-1}
							C	D			h_{T-1}
E										$J^T_{k_e+k_f}$	g_T
									F		h_T

(2.4)

Analogously problem (1.3), (1.4) can be given the linear program form

$$\text{maximize} \qquad z \equiv c_2^T y, \tag{2.5}$$

$$\text{subject to} \qquad A_2 y = b_2, \tag{2.6}$$

$$\underline{y} \leqslant y \leqslant \bar{y}, \tag{2.7}$$

where $y = (x_0, u_0, x_1^1, u_1^1, x_1^2, u_1^2, \ldots, x_1^{k_1}, u_1^{k_1}, \ldots, x_{T-1}^{r_{T-1}}, u_{T-1}^{r_{T-1}}, x_T^1, x_T^2, \ldots, x_T^{r_T}, s)$

$r_i = \prod_{j=1}^{i} k_j$, and s denotes slack variables associated with constraints (1.4a). Vectors $\underline{y}$ and $\bar{y}$ are generated from constraints (1.4b), (1.4c) and the type of the slacks.

Matrix A_2 and vector b_2 have the form (for simplicity we assume $T = 2$ i.e. a three stage problem, $k_1 = 3$, $k_2 = 2$)

x_0	u_0	x_1^1	u_1^1	x_1^2	u_1^2	x_1^3	u_1^3	x_2^{11}	x_2^{12}	x_2^{21}	x_2^{22}	x_2^{31}	x_2^{32}	s	
$-G$	$-K$	I_n													b_1^1
$-G$	$-K$			I_n											b_1^2
$-G$	$-K$					I_n									b_1^3
		$-G$	$-K$					I_n							b_2^1
		$-G$	$-K$						I_n					J_M	b_2^2
				$-G$	$-K$					I_n					b_2^3
				$-G$	$-K$						I_n				b_2^4
						$-G$	$-K$					I_n			b_2^5
						$-G$	$-K$						I_n		b_2^6

(2.8)

where J_{M_2} denotes a diagonal matrix of dimension $M_2 \times M_2$ with elements on the diagonal equal to $+1$ or -1.

Problem (2.5)–(2.7) has $N_2 = (n+m)\left(1+\sum_{i=1}^{T-1}\prod_{j=1}^{i} k_j\right)+n\prod_{j=1}^{T} k_j$ structural variables and $M_2 = n\sum_{i=1}^{T-1}\prod_{j=1}^{i} k_j$ constraints (and slack variables).

Both problems (2.1)–(2.3) and (2.5)–(2.7) become large-scale ones as their dimensions grow with the growth of the stage number T (this growth is linear for the dynamic problem and exponential for the stochastic problem).

The simplex method operates on columns of the constraint matrix. Usually only the nonzero elements of the column are stored (see: [9], [11]). It is not necessary to store all columns of matrix A_1 (or A_2) because, as we can see, matrix (2.4) (or (2.8)) has many blocks with identical elements.

For retrieving any column of A_1 it suffices to store matrices G, K, C, D, E, F and J_I^i, $i = 0, 1, \ldots, T$ (as sparse matrices, of course). Similarly, for restoring any column of A_2 it suffices to store matrices G, K and J_{M2}.

The implementations described in [7] and [8] exploit the above remark, opening as a result the possibility of solving truly large-scale problems on a microcomputer.

3. The revised simplex method

The linear programming problem can be formulated as follows

$$\text{maximize} \qquad z \equiv c^T y, \tag{3.1}$$

$$\text{subject to} \qquad Ay = b, \tag{3.2}$$

$$y \geqslant 0, \tag{3.3}$$

As we stated before, we assume that the reader is familiar with the simplex method (see, e.g. [3], [10]).

We shall present its compact form based on [10]. Let us denote by y_B and y_N the basic and nonbasic parts of the vector y, respectively, and by B the basis (a nonsingular submatrix of A consisting of columns associated with y_B). We assume that y_B is feasible ($y_B \geqslant 0$).

The algorithm consists of the following steps.

STEP 1. Computing dual variables.

$$\text{Solve} \qquad B^T \pi = c_B. \tag{3.4}$$

STEP 2. Pricing nonbasic columns.

$$\text{Compute} \qquad d_j = \pi^T a_j - c_j, \quad j = 1, 2, \ldots, N-M. \tag{3.5}$$

STEP 3. Choosing the entering variable.

Find $$d_q = \min_{\{j: d_j < 0\}} d_j.$$

If $d_j \geqslant 0$ for all $j = 1, 2, ..., N-M$, then STOP (optimal solution found).

STEP 4. Computing the entering column.

Solve $$B\alpha_q = a_q. \tag{3.7}$$

STEP 5. Choosing the leaving variable.

Find $$\vartheta = \frac{\beta_p}{\alpha_{pq}} = \min_{\{i: \alpha_{iq} > 0\}} \frac{\beta_i}{\alpha_{iq}}, \tag{3.8}$$

where β denotes the solution of the equation $B\beta = b$. If there is no α_{iq} fulfilling the condition $\alpha_{iq} > 0$, then STOP (the objective is unbounded).

STEP 6. Updating the basis.

Variable $(y_N)_q$ becomes the p-th basic variable. Column a_q replaces the p-th column of matrix B. The new basis $\bar{B}$ can be found from the equation

$$\bar{B} = B + (a_q - Be_p)\, e_p^T. \tag{3.9}$$

Go to step 1.

The basis inverse representations of B^{-1} usualy make use of the sparsity of the problem. Basis B can be for example decomposed into two matrices L and U (lower and upper triangular) $B = LU$. A stable method of performing such a decomposition and a way of updating it at each iteration is described in [1] ([6] gives more detail).

4. Simplex modifications for specially structured problems

The most time consuming parts of the simplex method are solving equations (3.4) and (3.7) and computing prices of the nonbasic columns (3.5). Solving (3.4) and (3.7) strongly depends on the basis inverse representation. Pricing as proposed by (3.5) needs computing $N-M$ inner products at every iteration. In [10] it is shown that even for general linear programs substantial modifications of (3.5) are possible.

In partial pricing the prices

$$d_j = \pi^T a_j - c_j, \tag{4.1}$$

are not computed for all nonbasic variables but only for their subset. This subset is cyclically changed from iteration to iteration.

In multiple pricing the full pricing (3.5) is made once per a few iterations. The result of (3.5) is a list of variables that can enter the

basis at a given iteration. One of them is pivoted into the basis and in the next few iterations the other condidates from the list are checked to see if they can enter the basis. The next full pricing is repeated when the list of candidates is exhausted.

The special structures of the linear programs discussed in this paper encourage using modifications of the pricing techniques.

Different specialized pricing techniques which are efficient for staircase linear programs are presented in [5]. These techniques are derived from algebraic properties of the problem.

Let us now go back from the linear programming problem (2.1)–(2.4) to the original problem of optimal control of the dynamic linear system (1.1)–(1.2). Optimal solutions (both state x_i and control u_i) of consecutive moments should behave similarily, i.e. if at the i-th cycle the s-th coordinate of x is at its bound $x_i^s = l_{xi}^s$ (or $x_i^s = L_{xi}^s$) we may expect the same in the next cycles $x_{i+t}^s = l_{xi+t}^s$ (or $x_{i+t}^s = L_{xi+t}^s$) for $t = 1, 2, \ldots$. This property in terms of linear programming can be interpreted as having similar activity in the cycles lying close to each other. The above remark leads to the following pricing rule which combines both partial and multiple pricing.

Let us denote by q the number of the columns of constraint matrix that entered the basis at a given iteration. In the next iteration column $q' = q + w$ is first priced out (w is the width of the single cycle, $w = n + m + n + k$ for problem (2.1)–(2.4)). If price d_q enables pivoting column q' into the basis, this is done without pricing any other column. If, on the contrary, column q' cannot enter the basis, all columns of the next cycle are scanned in the search for a good candidate. The columns with prices of the appropriate signs are put into the candidate list and then are tried for pivoting into the basis.

Applying the strategy described above reduces substantialy the solution time.

Analogous results, with even greater time savings, were obtained when this strategy was applied to the multistage stochastic dynamic linear programming problem (2.5)–(2.8).

Another possibility of reducing computation time is to start from an automatically constructed advanced basis (instead of a clack basis) that is usually much closer to the optimal basis. Commercial linear programming packages [9], [11] optionally construct such a *crash* basis. After analysing the layout of the nonzero elements in the constraint matrix the slack columns in the starting basis are replaced by as many structural columns as possible. The replacement is made in a way ensuring the nonsingularity of the crash basis. Practicaly this is done by constructing the crash basis so that it can be transformed by column and row permutations to a lower triangular matrix.

Let us now return to the problems considered. Analysing the structures of both constraint matrices (2.4) and (2.8) we can easily construct crash bases for problems (2.1)–(2.3) and (2.5)–(2.7).

For the first linear program we have

$$B_{01} = \begin{array}{|c|c|c|c|c|c|c|c|}
\hline
x_1 & s_{02} & x_2 & s_{12} & \dots & x_T & s_{T-1,2} & s_T \\
\hline
I_n & & & & & & & \\
 & J_k^0 & & & & & & \\
\hline
-G & & I_n & & & & & \\
C & & & J_k^1 & & & & \\
\hline
 & & & & \ddots & & & \\
\hline
 & & & & & I_n & & \\
 & & & & & & J_T^{T-1} & \\
\hline
 & & & & & \boxed{\text{F}} & & J_{k_e+k_f}^T \\
\hline
\end{array} \tag{4.2}$$

where s_{i2}, $i = 0, 1, \dots, T-1$, denote slack variables associated with constraints of type (1.2b). Solving equations (3.4) and (3.7) with basis B_{01} may yield numerical difficulties; this threat grows exponentialy with T. This problem can be overcome by using in B_{01} every, say, fifth slack variable $s_{i-1,1}$ associated with the constraint of type (1.2a) instead of the appropriate state variable x_i.

For the second linear program a crash basis can be constructed in the following form (for simplicity we assume $T = 2$, $k_1 = 3$, $k_2 = 2$)

$$B_{02} = \begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
x_1^1 & x_1^2 & x_1^3 & x_2^{11} & x_2^{12} & x_2^{21} & x_2^{22} & x_2^{31} & x_2^{32} \\
\hline
I_n & & & & & & & & \\
 & I_n & & & & & & & \\
 & & I_n & & & & & & \\
\hline
-G & & & I_n & & & & & \\
-G & & & & I_n & & & & \\
\hline
 & -G & & & & I_n & & & \\
 & -G & & & & & I_n & & \\
\hline
 & & -G & & & & & I_n & \\
 & & -G & & & & & & I_n \\
\hline
\end{array} \tag{4.3}$$

As in the previous problem equations (3.4) and (3.7) with basis B_{02} may be difficult to solve. Anyway, this is not so dangerous here, mostly due to the small values of T.

5. Computational experience

Modifications of the simplex method discussed above were implemented in experimental codes [7], [8]. Their efficiency was confirmed on several testing examples.

Six dynamic linear programming problems were employed in the experiments. Their dimensions (excluding objectives and right-hand sides) are given in Table 1. Additional information about these test problems is collected in [7].

Table 1

Problem	Periods T	Rows M1	Columns N1	Nonzero values
D1	3	20	18	78
D2	20	105	103	452
D3	100	505	503	2212
D4	180	905	903	3982
D5	70	570	635	3109
D6	70	570	635	3109

Three methods were used to solve the problems:

Method 1 a standard simplex method (with a supersparse representation of the problem data); starting basis constructed totally from slacks;

Method 2 the simplex method with modified pricing technique described in section 4; starting basis constructed totally from slacks;

Method 3 the simplex method with modified pricing technique and crash basis of type (4.2).

The results are compared in Table 2.

As we can see, aplying the modified pricing technique reduced the number of iterations (compare the number of iterations in methods 1 and 2). For problems D3, D4, D5, D6 (i.e. those of large scale) this reduction is surprisingly stable — about 15%. The reduction of the computation time is even greater (it reaches 30%) which means substantial decreases in the average time per iteration. Both observations confirm our supositions of section 4.

Starting from a crash basis (i.e. method 3) gives further savings of computation time and the number of iterations. In two sparser problems

Table 2

Problem	Solution method			% of time saved
	Method 1	Method 2	Method 3	
D1 time iterations	8 s 31	7 s 29	5 s 13	37
D2 time iterations	1 m 50 s 177	1 m 22 s 167	56 s 86	49
D3 time iterations	28 m 48 s 1077	20 m 49 s 917	15 m 25 s 645	46
D4 time iterations	60 m 37 s 1409	47 m 14 s 1223	38 m 21 s 970	37
D5 time iterations	30 m 5 s 973	21 m 39 s 812	20 m 35 s 726	32
D6 time iterations	29 m 58 s 1000	21 m 40 s 839	20 m 33 s 723	32

D3 and D4, these savings are greater (20–25%), in denser problems D5 and D6, the reductions are small (the computation time is only 5% shorter then in method 2). Explanation of this difference needs more numerical evidence.

Also six stochastic linear programming problems were employed in the experiments. Their dimensions (excluding objectives and right-hand sides) are given in Table 3. Additional information about these test problems is collected in [8].

Table 3

Problem	Periods T	Rows M1	Columns N1	Nonzero values
S1	2	27	38	135
S2	3	81	104	405
S3	4	351	410	1755
S4	7	762	1019	3028
S5	4	360	417	5310
S6	5	744	849	3844

Analogously three methods were used to solve the problems:

Method 1 a standard simplex method (with a supersparse representation of the problem data); starting basis constructed totally from slacks;

Method 2 the simplex method with modified pricing technique described in section 4; starting basis constructed totally from slacks;

Method 3 the simplex method with modified pricing technique and crash basis of type (4.3).

The results are compared in Table 4.

Let us observe that application the modified pricing technique does not change the number of iterations (compare the appropriate values for methods 1 and 2). The only exception is the problem S4, in which this number is reduced about 10% (problem S4 has the lowest density of the nonzero values in the constraint matrix).

Although the number of iterations is unchanged, there are savings of the computation time. This is due to the reduction of the average time per iteration.

Method 2 is not so efficient for stochastic problems as it was for those of dynamic type, because of the more complicated analogies between the cycles of the constraint matrix. For stochastic problems a more sophisticated pricing rule has to be applied.

Starting from a crash basis further reduces both the iterations number and the computation time. For sparser problems such as S4 or S6 those reductions are greater.

The results of running test problems with the modified simplex method may be summarized as follows:

(i) The modified pricing technique described in section 4 is efficient for both dynamic and stochastic problems. It reduces the average time per iteration and also decreases the number of iterations.

(ii) This pricing rule gives better results in phase 2 of the simplex method, i.e. when analogies between the cycles are more regular (the objective changes in phase 1 but does not change in phase 2).

(iii) Starting from a crash basis substantially reduces the number of iterations (especially in phase 1 of the method, i.e. when a feasible solution is looked for). Its further consequence is a noticable decrease in computation time.

Table 4

Problem	Solution method			% of time saved
	Method 1	Method 2	Method 3	
D1 time iterations	10 s 48	10 s 50	6 s 21	40
D2 time iterations	54 s 151	50 s 151	31 s 76	43
D3 time iterations	10 m 15 s 607	9 m 22 s 595	6 m 13 s 294	40
D4 time iterations	43 m 50 s 1242	36 m 12 s 1125	19 m 04 s 443	56
D5 time iterations	14 m 14 s 495	13 m 13 s 477	8 m 42 s 178	38
D6 time iterations	38 m 49 s 1110	37 m 40 s 1102	16 m 7 s 399	58

The modifications described in this paper shorten by about 30–50% the solution time of the dynamic problem and by about 40–60% the solution time of the stochastic problem.

6. Conclusions

We have presented simple modifications of the simplex method specializing it for solving dynamic and mutlistage stochastic problems. The modifications exploit special structures of the problems considered.

It has been shown that such modifications accelerate the method about two times.

Further modifications should include specializing the basis inverse representation for the structures of the problems considered. Three different methods, presented in [2], [4] and [15], seem attractive for this purpose. We may expect that a more compact basis inverse representation will result in the possibility of solving larger problems and will also give time savings.

Acknowledgment

The author is grateful to Dr Andrzej Ruszczyński for introducing into this research and to Dr Krzysztof Kiwiel for his close reading of this paper.

References

[1] Bartels R. H., Golub G. H. The simplex method of linear programming using LU decomposition. *Communication on ACM*, **12** (1969), 266–268.

[2] Bisschop J. Meeraus A. Matrix augmentation and structure preservation in linearly constrained control problems. *Mathematical Programming*, **18** (1980), 7–15.

[3] Dantzig G. Linear Programming and Extensions. Princeton 1963.

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

[5] Fourer R. Solving staircase linear programs by the simplex method 2: pricing. *Mathematical Programming*, **25** (1983), 251–293.

[6] Golub G. H. Van Loan C. F. Matrix Computations. North Oxford Academic, 1983.

[7] Gondzio J. User's guide for a package for solving large scale dynamic linear programming problems. (In Polish). Technical report. Systems Research Institute, Polish Academy of Sciences, Warsaw 1988.

[8] Gondzio J. User's guide for a package for solving multistage stochastic linear programming problems. (In Polish). Technical report. Systems Research Institute, Polish Academy of Sciences, Warsaw 1988.

[9] Marsten R. The design of the XMP linear programming library. *ACM Transactions on Mathematical Software*, **7** (1981), 481–497.

[10] Murtagh B. Advanced Linear Programming. McGraw-Hill, 1981.

[11] MURTAGH B., SAUNDERS M. MINOS 5.0 user's guide. Stanford University, 1983.
[12] PEROLD A. F., DANTZIG G. B. A Basis Factorization Method for Block Triangular Linear Programs. In: I. S. Duff and G. W. Stewart eds. Sparse Matrix Proceedings 1978 SIAM, Philadelphia 1979, 283–312.
[13] PROPOI A., KRIVONOZHKO V. The simplex method for dynamic linear programs. In: G. B. Dantzig, M. A. H. Dempster and M. Kallio eds. Large-scale linear programming. International Institute for Applied Systems Analysis, CP-81-S1, 1981, 299–366.
[14] RUSZCZYŃSKI A. Modern techniques for linear dynamic and stochastic programs. In: A. Lewandowski and A. Wierzbicki, eds. Theory, Software and Testing Examples for Decission Support Systems. International Institute for Applied Systems Analysis, WP-87-26, 1987, 27–43.
[15] WETS R. Large scale linear programming techniques in stochastic programming. International Institute for Applied Systems Analysis, WP-84-90, 1984.

Received, October 1987.

Modyfikacje metody sympleksów wykorzystujące specjalne cechy dynamicznych i stochastycznych zadań programowania liniowego

W pracy rozważane są dynamiczne i stochastyczne dynamiczne zadania programowania liniowego. Problemy te sformułowane w postaci klasycznych zadań programowania liniowego mają macierze ograniczeń o specyficznych strukturach: schodkowej oraz kątowo-schodkowej, odpowiednio dla zadań dynamicznego oraz stochastycznego. Opracowano modyfikacje metody sympleksów, które wykorzystując specjalne struktury macierzy ograniczeń, umożliwiają istotne zmniejszenie czasu rozwiązywania zadań oraz zmniejszenie wymagań pamięci. Metodę oprogramowano dla mikrokomputera IBM PC. Przedstawiono wyniki obliczeń dla zadań testowych o wymiarach do 1000×2000.

Модификации метода симплексов, использующие особые признаки динамических и стохастических задач линейного программирования

В работе рассматриваются динамические и стохастические задачи линейного программирования. Эти задачи, формулируемые в виде классических задач линейного программирования, имеют матрицы ограничений со специфическими структурами: ступенчатой и угловой ступенчатой, соответствующе для динамических и стохастических задач.

Разработаны модификации метода симплексов, которые используя особые структуры матрицы ограничений, позволяют существенно сократить время решения задач а также снизить требования к памяти. Для реализации метода разработана программа на микрокомпьютер ИБМ ПС. Представлены результаты вычислений для тестирующих задач размерностью до 1000×2000.