Bull. Nov. Comp. Center, Num. Anal., 11 (2002), 103–112
© 2002 NCC Publisher

Linear optimization programs*

G.I. Zabinyako, E.A. Kotelnikov

The paper presents the software for the linear programming problems both for the integer and the mixed-integer linear programming. For the solution of problems of the above types it is possible to exploit a parallel algorithm. Some examples of solution of test problems are given.

1. Objectives

The software for the linear programming (LP) has been developed for solution of problems of the following form: find $\max f(x) = (c, x)$ with the constraints

$$Ax = b, \quad \alpha \le x \le \beta.$$

Here A is $m \times n$ matrix; $c, x, \alpha, \beta \in R^n$; $b \in R^m$. The software programs are designed for the problems with $m \le 32000$. The variables x_j can be bounded either only from below or from above, or they can be free. These variables can also be fixed assuming $\alpha_j = \beta_j$. General constraints can be set as inequalities not putting them into the form of equalities.

In problems of the integer and the mixed-integer linear programming (ILP) the requirement of some variables $x_j \in J$ to be integer and the constraint to apply $|J| \le 32000$ are added.

2. General information about the software

The software has been written in FORTRAN-77, debugged in the operating systems DOS and UNIX. The parallel version of the ILP-algorithm is implemented within the MPI parallel programming system.

The input data of problems are represented in the MPS-format [1]. The software contains the numerical solution procedures and programs for data processing to provide the input, the logical data control, compiling dictionaries and reference tables, the transfer from the external MPS-format to the internal format of the package and forming the output forms.

Attempts to apply the LP for modelling real processes frequently encounter the difficulty which is inconsistency of the system of constraints of

*Supported by the Russian Foundation for Basic Research under Grants 99-07-90422, 01-07-90367.

the model. In the LP package, the version of approximation of an inconsistent system by a consistent one has been realized. In addition to a standard section of the MPS data, a supplementary section for correcting the vector b in the right-hand side has been introduced. The user selects which of b_i are subject to correction and the admissible value of their variation δb_i. If it is b_i which is connected with a constraint in the form of an equality, which is subject to correction, then additional variables u_i and v_i are introduced, and the respective constraint is transformed to the form

$$(a_{i.}, x) = b_i + u_i - v_i, \quad 0 \le u_i \le \delta b_i, \quad 0 \le v_i \le \delta b_i.$$

A constraint on the type of inequalities, for example, $(a_{i.}, x) \le b_i$ is transformed as follows:

$$(a_{i.}, x) \le b_i - v_i, \quad 0 \le v_i \le \delta b_i.$$

The user sets only the name of a constraint corresponding to b_i and the value δb_i, the formation of constraints being automatically done. Then the LP problem of minimization of a sum of the variables u_i and v_i with allowance for the transformed system of constraints is solved.

It is assumed that in the ILP software the problem with integer variables not taken into account is inconsistent, and provision is not made for the possibility to correct constraints.

The column-wise sparse format is used for the matrices, defined by analogy with the row-wise sparse format [2]. The latter can also be used to call the software programs.

The matrix B^{-1} inverse to the basis matrix B is represented in the multiplicative form. To store multiplicators, special data aggregates are used. Multiplicator columns are stored in the 8-byte field. The first 8-byte word of each multiplicator consists of 4-byte integer variables $\dot{I}^-$ and $\dot{I}^+$, being indicators to the previous and the successive multiplicators. After $\dot{I}^-$ and $\dot{I}^+$ a real 8-byte variable G inverse to the leading element is written. Then follow 2-byte integers: iv is the number of the leading element, k is the quantity of nonzero elements of a column without a leading element, $i_1 \dots i_k$ are the numbers of nonzero components. Then 8-byte real variables $\alpha_{i1}, \dots, \alpha_{ik}$ are located, which are nonzero elements of the column.

The algorithms are governed by the numerical parameters which indicate to the frequency of addressing to the reinversion procedure, specify the accuracy of fulfilling the conditions according to the direct and the dual variables in the LP problems, define the constraints on selection of the leading elements, etc. The user sets the names of two files, the first one containing statistical data on the problem and its solution, and the second file – the information which is necessary for resuming the solution of the problem if needed. Options of setting the parameters are written down into files with fixed names (different for the LP and the ILP). The ILP programs provide for some approaches to be exploited by certain options to obtain approximate solutions.

3. The simplex method procedures

Below we briefly present characteristics of the basic procedures of the simplex method and some examples of solution to poorly-scaled problems of the LP.

3.1. Correction of matrices inverse to the basic matrices. Let at a recurrent iteration of the simplex-method, the p-th column of the basic matrix B be replaced by the column $a_{\cdot q}$ of the matrix A. The matrix $\bar{B} = B+(a_{\cdot q}-Be_p)e_p^T$ becomes basic, where e_p is the p-th unit vector in R^m. The matrix $\bar{B}^{-1}$ is calculated by the Forrest–Tomlin method [3] supplemented with operations to control the numerical stability. For the newly formed multiplicator U_k^{-1}, the smallness of the leading element module is controlled. The maximum value of modules of the elements U_k^{-1} is compared to the maximum of modules of the elements in $\bar{B}$. Such verifications can result in taking the decision on the unrecurrent appeal to the reinveresion procedure.

Before we appeal to the correction procedure, the vector $s = \pm B^{-1}a_{\cdot q}$ and the number of the leading row p are determined, then the value $s_{\max} = \max_{i=1,\dots,m} |s_i|$ is calculated. If $s_{\max}/|s_p| > M$, where M is a sufficiently large positive number, the unrecurrent appeal to the reinversion procedure is done, or the decision on the fact that the column $a_{\cdot q}$ at several immediate iterations should be kept as non-basic is taken.

3.2. The reinversion procedure. In the algorithm of the inverse matrix renewal [4] we make an attempt to minimize the filling-up of the multiplicative presentation of the matrix $B^{-1} = U^{-1}L^{-1}$ with non-zero elements, where each of the matrices U^{-1} and L^{-1} is the product of the corresponding multiplicators. This is a heuristic method of reduction of the matrix B by rearranging rows and columns so that this matrix be as close as possible to the bottom-triangular shape.

The columns which after rearrangements have no non-zero elements above the main diagonal are called *normal*, while the rest ones – the *columns-spikes*. When selecting the leading elements, the normal columns generate only L^{-1}-multiplicators, and the column-spikes – L^{-1} and U^{-1}-multiplicators. The normal column multiplicator is completely defined by values of elements of the respective column of the matrix A. Therefore only the number of a column from A and the number of the leading row are stored in the multiplicator field.

In the software in question, the algorithm has been implemented, in which in addition to heuristics [4] an attempt is made to apply standard operations of controlling the numerical stability when selecting the leading elements [5].

3.3. Calculation of the step. Let at a recurrent k-th iteration of the simplex-method the direction $s = \pm B^{-1}a_{\cdot q}$ be defined. The procedure of calculation of the step λ along s is based on the algorithm from [6]. First the maximum value λ_1 is defined such that $\alpha_j - \delta_k \leq x_j + \lambda_1 s_j \leq \beta_j + \delta_k$, where x_j are basic variables, δ_k is a small positive value which is increased at each iteration of the simplex-method. When selecting the leading row p, an attempt is made to maximize the value $|s_p|$ using the relations $\lambda_2 \leq \lambda_1$ and $\alpha_j \leq x_j + \lambda_2 s_j \leq \beta_j$. As a result, $\lambda = \max\{\lambda_2, \tau/|s_p|\}$, where τ is a certain positive value, $\tau \ll \delta_0$. At the $(k+1)$-th iteration $\delta_{k+1} = \delta_k + \tau$.

After reinversion, it may appear that some basic variables x_j do not satisfy the two-sided constraints. The relations $\alpha_j \leq x_j \leq \beta_j$ are reconstructed with the help of solution to the LP problem in whose object function $c_j = 0$ with $\alpha_j \leq x_j \leq \beta_j$, $c_j = 1$ with $x_j < \alpha$ and $c_j = -1$ for $x_j > \beta_j$. The initial value λ_1 of the step in the direction s is determined as

$$\lambda_1 = \arg\min_{\mu>0} \sum_j (\max\{0, \alpha_j - x_j - \mu s_j\} + \max\{0, x_j + \mu s_j - \beta_j\}).$$

The leading row p and the step λ are selected in the manner similar to the above-considered version.

3.4. Some examples of solution to the LP-problems. As examples we have selected five poorly-scaled problems from a set of the NETLIB tests [7]. Table 1 contains: dimensions, nz-the number of non-zeroes in the matrix, modulo values of the minimum and the maximum elements a_{ij} of the matrix A.

It has not been possible to obtain solutions to the problems presented in Table 1 with a reasonable accuracy without counterbalance algorithm of the constraint matrices elements. In the LP-software, the counterbalance procedure from [8] has been implemented. Table 2 presents the results of the solution to problems employing the counterbalance. There is an option

Table 1

N	Problem name	m	n	nz	$\min\|a_{ij}\|$	$\max\|a_{ij}\|$
1	perold	625	1376	6018	5.3E−5	2.4E4
2	pilot.ja	940	1988	14698	2.0E−6	5.8E6
3	pilot.we	722	2789	9126	1.4E−4	4.7E4
4	pilot4	410	1000	5141	3.7E−5	2.8E4
5	pilotnov	975	2172	13057	2.0E−6	5.8E6

Table 2

N	it	t	$\|x^*\|_1$	$\|y^*\|_1$	h_r	h_c	V_1	V_0
1	4408	261	4.4E3	8.9E1	7.3E−7	7.7E−10	31399	23230
2	5405	578	4.3E3	7.7E1	8.9E−7	9.2E−10	67151	49272
3	4334	315	6.3E3	2.1E6	3.0E−11	1.5E−12	29543	21095
4	1290	39	7.8E3	1.2E2	6.8E−7	3.9E−11	20346	13123
5	1593	180	5.3E4	2.3E1	1.3E−6	3.5E−15	60124	45168

in the file specifying the controlling parameters values in which it is possible to indicate whether the scaling is necessary. When solving the problems, the frequency of addressing to the reinversion procedure was set equal to 50, i.e., after carrying out 50 recurrent recalculations of the inverse matrix. In the course of solution due to computing errors there arose additional addressings to this procedure. Thus, for Problem 1 there were 46 unrecurrent addressings, while for Problems 2 and 3 there was one for each.

Estimates of errors and norms of the optimal values of the direct $\|x^*\|_1$ and the dual variables $\|y^*\|_1$ are given in Table 2 for the original scales (i.e., after carrying out the inverse transformations corresponding to the counterbalance).

Remark. N is the number of a problem; it is the number of iteration of solution to problem; t is time of solution in sec. on IBM PC with the clock frequency 75 MHz; $h_r = \max_i |(a_{i\cdot}, x^*) - b_i|$; $h_c = \max_{j\in J_B} |(a_{\cdot j}, y^*) - c_j|$, where $a_{\cdot j}$ is the j-th column of the matrix A, and j runs the list of basic columns; c_j is the j-th component of the object function vector c; V_1 is the maximum number of double words occupied with multiplicative presentation of inverse matrices in the course of the problem solution; V_0 is the maximum memory under multiplicators after the reinversion.

Different versions of solution to the NETLIB problems and the detailed information about the algorithms are presented in [9].

4. The integer linear programming

In the ILP programs, the method of branching and boundaries with one-sided branching has been implemented. Let us briefly dwell on the sequential algorithm which is considered in good detail in [10] and [11].

4.1. The algorithm with branching and boundaries with one-sided branching. The algorithm is based on solving a series of the LP estimation problems. The branching variable is selected by penalties [1, 12]. Let us present the basic variable x_j for $j \in J$ in the optimal basis of the recurrent estimation LP problem in the form $x_j = [x_j] + v_j$, where $[x_j]$ is the integer of x_j. Let P_j^+ stand for the penalty for the increase of x_j by the value $1-v_j$, and P_j^- for the decrease by the value v_j.

Let x^i be the optimum to the i-th estimation problem, and $f^i = f(x^i)$ and r^i – the value of the incumbent corresponding to the given time instant. If an admissible integer solution is still to be found, then $r^i = -\infty$. For those basic variables $x_j, j \in J$, where x_j^i is non-integer, calculate the penalties P_j^+, P_j^-. Among them, define the minimum penalty $P_{\min}$. If $f^i - P_{\min} \leq r^i$, then the i-th estimation problem is selected from a higher level in the list of problems.

At $f^i - P_{\min} > r^i$, the branching is realized with respect to the basic variable x_j, to which corresponds the maximum penalty P_j^+ or P_j^-. The schemes of the method of branching and boundaries with one-sided branching make possible to use a compact form of lists of estimation problems. Let at a certain level k, the variable x_j with the penalties P_j^- and P_j^+ be selected for branching. If $P_j^- \leq P_j^+$, then the problem corresponding to P_j^- is selected as a recurrent estimation problem, and it is necessary to store the information about the alternative problem in the list of estimation problems (in our case in the auxiliary array h). To this end, the assignments $h(1,k) = j$; $h(2,k) = \beta_j^i$ are performed, where β_j^i is the upper boundary of the variable x_j in the i-th estimation problem at the preceding level $(k-1)$; $h(3,k) = 1$ if $f^i - P_j^+ \leq r^i$ and $h(3,k) = 0$ otherwise (if $h(3,k) = 1$, we will consider the corresponding branch to be marked); $h(4,k) = f^i$; $h(5,k) = P_j^+$. At $P_j^+ < P_j^-$, the estimation problem, corresponding to the penalty P_j^+ is selected, and in the array h the following values are stored: $h(1,k) = -j$; $h(2,k) = \alpha_j^i$; $h(3,k)$ is either equal to 0 or 1 depending on fulfilling the inequalities $f^i - P_j^- \leq r^i$; $h(4,k) = f^i$; $h(5,k) = P_j^-$.

The schemes with one-sided branching allow one to easily go from one estimation problem to another. If the fixed value $x_j^i = \alpha_j^i = \beta_j^i$ is assigned to the variable selected for branching, then the fictitious variable is introduced into the basis, to provide non-degeneracy of the basic matrix. Otherwise the optimal basis of the preceding problem is used. Solution to any estimation problem except for the problem corresponding to $i = 0$ starts with verification of tolerances of the basic variables. If constraints are violated, then the algorithm of minimization of the piecewise-linear function to provide tolerance is applied (see Section 3.3). Only zero values are considered admissible for fictitious variables.

It is possible to modify the scheme of problem solution using controlling parameters. For example, to shorten the selection procedure, the user can set the estimation $\tilde{f}$ of the value of the object function. If the optimum f^i of the estimation problem appears less than $\tilde{f}$, then in the sequel it is not used as parent. For obtaining approximate solutions, the input parameter δ_f is provided. The process of solving the problem terminates if the recurrent incumbent r^i satisfies the inequality $(f^0 - r^i)/|f^0| \leq \delta_f$.

4.2. Parallel algorithm. The parallel algorithm is so structured that at each of processor elements the algorithm with branching and boundaries with one-sided branchings is implemented in the asynchronous mode. Among the processor elements, an element with zero number is distinguished. The data on a problem are read from the external memory by zero processor and are transferred to all the rest. Then the original problem is solved on zero processor with no allowance for integer features, while all the other processes are expecting at this time instant. In the course of solution the reference information, needed for organizing the parallel computations is being accumulated in zero element.

For loading any processor (including zero processor in the sequel) the arrays ISB, ISN and, partially, h are transferred to its main memory. The integer array ISB of m components contains the numbers of basic variables, the integer array ISN of n dimension for non-basic variables shows at which boundary – the upper or the lower – are variables, and for the basic variables in ISN, the numbers of positions in the basis are written. A part of the array h to be transferred, is determined by the level at which the solution of estimation problems starts.

Let it be necessary to load a certain processor from zero processor. Assume a variable x_j be selected for branching at the level k of zero processor and $P_j^- \leq P_j^+$. The estimation problem corresponding to the penalty P_j^- will be solved on zero processor. If $f^i - P_j^+ > r^i$, the data are transferred to the target processor element. At zero processor element, the branch corresponding to P_j^+ is marked (1 is assigned to the element $h(3,k)$).

At the receiving side, all the unmarked branches, having the level higher than k, are marked. The matrix B^{-1} is constructed based on the list of ISB by means of the reinversion procedure.

The data of ISB, ISN, and h are sufficient for the formation of the first estimation problem at a given processor element. Then the algorithm with branching and boundaries with one-sided branching is carried out.

In the parallel algorithm it is desirable, where possible, to begin the calculation process with a higher branch at each processor element. To this end, at each of processor elements the arrays ISB and ISN are stored, which are transferred to a certain other processor as soon as the call comes.

If a new incumbent r^i is received at a certain processor, this information is transferred to all the other. After receiving r^i, the fulfillment of the inequalities $h(4,k) - h(5,k) \leq r^i$ is checked up, where k corresponds to the level with which the fulfillment of the algorithm at a given processor element started. In those cases when this inequality holds, the call for a new assignment is done.

The obtained value of r^i is used for the revision of marked and unmarked branches in the array h. It may appear that the information in the arrays ISB, ISN becomes out of date. In this case, new versions of ISB, ISN are prepared, and the message about the change of the level corresponding to these arrays is sent to the zero processor.

The problem is considered to be solved if the incumbent $r^i = f^0$ is obtained, or zero level is attained in all the processors.

The data of the problem are transferred by the blocked MPI function BCAST [13]. Further all the processes are carried out asynchronously, and the data are exchanged by unblocked functions.

4.3. Examples of solution to the ILP problems. Large sets of test problems in different application areas of the ILP have been stored in the MPS-format. To check up the validity of the software, the tests were selected from: `ftp.camm.rice.edu/pub/plople/bixby/miplib/miplib3/`

Table 3 demonstrates the dimensions of the problems.

Table 3

N	Problem name	m	m_e	n	n_i	n_b	nz
1	air03	124	124	10757	10757	10757	91028
2	bell3a	123	—	133	71	39	347
3	bell5	91	—	104	58	30	266
4	blend2	274	89	353	264	231	1409
5	dcmulti	290	78	548	75	75	1315
6	fiber	363	363	1298	1254	1254	2944
7	gen	780	150	870	150	144	2592
8	gesa3	1368	48	1152	384	216	4944
9	gesa3_0	1224	120	1152	672	336	3622
10	l152lav	97	96	1989	1989	1989	9922

Remark. N is the number of a problem; m is the total amount of constraints, of which m_e is that on equalities; n is the number of variables of which n_i is that of integer, and n_b is that of the Boolean variables; nz is the number of nonzero elements in the matrix A.

Table 4 represents some characteristics of the solution process of problems by the sequential algorithm. The ILP-problems were solved on the multiprocessor common memory computer RM600.

Remark. N is the number of a problem; it_i is the quantity of iterations of the solution of a problem, k_{rep} is the total number of addressings to the reinversion procedure, k_i is the number of incumbents obtained in the course of solution, t_1 is the time (in seconds) needed for solution.

Table 4

N	it	k_{rep}	k_i	t_1	N	it	k_{rep}	k_i	t_1
1	1421	28	1	17.04	6	36293085	721839	52	68141.
2	552937	12154	5	251.03	7	118468	2374	3	409.46
3	8105266	579981	558	3282.6	8	257365	5132	10	1465.1
4	305445	6772	13	281.56	9	482728	9623	37	2218.2
5	390673	7813	29	446.93	10	2502135	50064	22	5184.7

Table 5

N	it_0	it_1	it_2	it_3	$s(it)$	$s(\text{rep})$	t
1	1421	468	0	0	1889	38	19.58
2	140675	138439	138245	138624	555983	12362	64.06
3	473266	468836	519309	519050	1980461	148020	206.72
4	41205	40588	39584	40781	162158	3732	38.71
5	68906	53846	47463	48759	218974	4554	84.80
6	6485799	5704399	5595746	5393408	23179352	484614	12933.
7	42795	34692	31914	27926	137327	2801	146.05
8	92309	49356	42829	55231	239725	5073	529.28
9	204444	136010	127949	132077	600480	12282	922.21
10	463937	430534	456976	426948	1778395	35612	1071.2

Table 6

N	1	2	3	4	5	6	7	8	9	10
$it/s(it)$	0.75	0.99	4.09	1.88	1.78	1.56	0.86	1.07	0.80	1.41
t_1/t	0.87	3.92	15.88	7.27	5.27	5.27	2.80	2.77	2.40	4.84

Table 5 contains characteristics of the solution process of problems by the parallel algorithm. In the parallel version, four processor elements, each for 1 process, were used.

Remark. N is the number of a problem; it_i is the quantity of iterations performed on the i-th processor element; $s(it)$ is the total number of iterations on all the processor elements; $s(\text{rep})$ is the total number of reinversions; t is time (in seconds) needed for the fulfillment of the parallel algorithm.

Table 6 allows us to compare the efficiency of solution to problems by the sequential and the parallel algorithms.

In the table, $it/s(it)$ shows the ratio between the number of iterations in the sequential algorithm and the total number of iterations performed on the four processors. The ratio t_1/t shows the acceleration gained as a result of parallelization. As seen from the data of the tables, Problem 1 represents an example of the big-size problem, which is easy to solve, and it makes no sense to use parallelization.

The process of solution of the ILP-problems is unstable. The number of iterations and other characteristics of the computational process are essentially dependent on the selection of the initial values of controlling parameters. The common property of instability of the process of solution is dramatized in certain problems by the following factors: non-uniqueness of solutions of the ILP estimation problems, ill-conditioning of the basic matrices. When solving the problems, the same initial values of controlling parameters were selected both for the parallel version of calculations and in calculations in the one-processor mode.

References

[1] Murtagh B. Advanced Linear Programming. Theory and Practice. – Moscow: Mir, 1984.

[2] Pissanetsky S. Technology of Sparse Matrices. – Moscow: Mir, 1988.

[3] Forrest J.J.H., Tomlin J.A. Updated triangular factors of the basis to maintain sparsity in the product form simplex method // Math. Programming. – 1972. – Vol. 2, № 3. – P. 263–278.

[4] Hellerman E., Rarick D. Reinversion with the preassigned pivot procedure // Math. Programming. – 1972. – Vol. 1, № 2. – P. 195–216.

[5] Zabinyako G.I. Algorithm of reinversion of the simplex-method // Proc. ICMMG, Ser. Sistemnoe modelirovanie. – Novosibirsk, 1998. – Vyp. 5 (23). – P. 59–73.

[6] Gill P.E., Murray W., Saunders M.A., Wright M.A. A practical anticyclone procedure for linearly constrained optimization // Math. Programming. Ser. B. – 1989. – Vol. 45, № 3. – P. 437–474.

[7] Gay D.M. Electronic mail distribution of linear programming test problems // Math. Programming Society COAL Newsletter. – 1985. – Vol. 13. – P. 10–12.

[8] Gill P.E., Murray W., Write M. Practical Optimization. – Moscow: Mir, 1985.

[9] Zabinyako G.I., Kotelnikov E.A., Kobkova T.M., Rozhin V.E. The linear programming programs LP-VC. – Novosibirsk, 1995. – (Report / SB RAS. Computing Center; GR № 01.9.30001317, № 02.9.5000357).

[10] Zabinyako G.I. Program package of the integer linear programming // Diskret. analiz i issled. operatsii. – 1999. – Ser. 2, Vol. 6, № 2. – P. 32–41.

[11] Zabinyako G.I. Program package of the integer linear programming. – Novosibirsk, 1998. – (Report / SB RAS. ICMMG; GR № 01.9.30 001317, № 02.9.80 005512).

[12] Kovalev M.M. Discrete Optimization (Integer Programming). – Minsk: Izd. Belorus. Univ., 1977.

[13] Korneev V.D. Parallel Programming in MPI. – Novosibirsk: Izd. SO RAN, 2000.