Bull. Nov. Comp. Center, Comp. Science, 22 (2005), 69–77

Parallelizing some problems of the discrete and the non-convex optimization

E.A. Kotel'nikov, G.I. Zabinyako

Abstract. This paper considers aspects of parallelizing of the solution process of problems of the integer linear and quadratic programming as well as of problems of global optimization of the non-convex quadratic programming.

Introduction

Solution to problems of discrete and global non-convex optimization call for their solution large computer costs. It is urgent to work out a special parallel software for solving such problems. Programs in question are being developed in FORTRAN with the use of the MPI parallel programming system.

Section 1 describes application of algorithms with branches and boundaries with one-sided branching as applied to problems of the integer linear and the integer quadratic programming. Parallelizing of the algorithms in question is carried out asynchronously on each processor. As a result, in the parallel version we have a combination of algorithms with one-sided and simultaneous branching. The program realization of algorithms with simultaneous branching is a time-consuming task and requires very big poorly structured arrays for storing the list of estimation problems. The use of the MPI considerably facilitates programming combinations of algorithms with one-sided and simultaneous branching. When executing in the parallel mode has shown the abbreviation of computer costs, i.e., the abbreviation of a general number of iterations made on all the processors for solving auxiliary tasks, as compared to the sequential mode. Thus, due to parallelizing we attain acceleration exceeding the linear one. In the course of parallelizing, processes exchange with short messages against execution of a large number of arithmetic operations, thus providing a high parallelizing efficiency both on computer systems with shared memory and on clusters. Many problems were solved on MVS-1000M and RM-600 E30 in the sequential and the parallel modes, some statistics is given below.

Section 2 presents some problems of the integer linear programming of a special type, i.e., covering problems and partitioning problems. Programs intended for solving the integer linear programming problems of a general type are not quite suitable for solving such problems. Therefore, to solve them, we should use special algorithms.

Section 3 deals with global optimization algorithms as applied to problems of the non-convex quadratic programming. A global maximum of the quadratic function is localized with the help of a decreasing sequence of linear or quadratic majorants of the object function, constructed on subdomains of an admissible domain. Solution to such problems can also be reduced to application of the method of branching and boundaries with one-sided branching, thus allowing us in the parallel mode to attain acceleration exceeding the linear one.

1. Problems of the integer linear and the integer quadratic programming

The problems presented are as follows: *Find a minimum* $f(x)$ *under the constraints*

$$Ax = b \tag{1}$$

$$\alpha \le x \le \beta \tag{2}$$

$$x_j \text{ are integers for } j \in J. \tag{3}$$

Here A is an $m \times n$ matrix; $x, \alpha, \beta \in R^n$; $b \in R^m$; J is a set of indices of integer variables. In problems of the integer linear programming (ILP), a linear function f is set by means of the vector $c \in R^n$. The object function in problems of the integer quadratic programming (IQP) is $f(x) = (Qx, x) + (c, x)$, where Q is a symmetric $n \times n$ matrix and $Q \ge 0$. The algorithms are designed with allowance for sparseness of the matrices A and Q.

1.1. Sequential algorithms. Solution to original problems is reduced to solution of a series of estimation problems. The latter differ from the original problems in that the requirement for integers (3) is absent, and in each i-th estimation problem, conditions (2) are replaced by the conditions $\alpha^i \le x \le \beta^i$. The vectors α^i and β^i are formed according to the rules of the method of branching and boundaries.

The ILP estimation problems are solved by the simplex method. For solution of the IQP problems, the reduced gradient method [1] is applied, where the initial variables x are subdivided into basic, superbasic, and nonbasic. Minimization in a subspace of superbasic variables is made by the linear conjugate gradient method. Selection of a branching variable is frequently done with the use of penalties [2].

The value of a basic variable (in the nonlinear case, the value of a superbasic variable as well) x_j for $j \in J$ in the optimal basis of the current estimation problem will be presented as $x_j = [x_j] + v_j$, where $[x_j]$ is the integer part of x_j. The penalty for an increase x_j by the value $1 - v_j$ is denoted by P_j^+, while for a decrease by the value v_j — by P_j^-.

The algorithms of solution to the ILP and the IQP problems consist of the following steps:

Step 0. Set $i = 0$, $k = 0$, the incumbent value $r^0 = +\infty$, $\alpha^0 = \alpha$, $\beta^0 = \beta$.

Step 1. Solve the estimation problem. Let x^i be an optimal solution and $f^i = f(x^i)$. If $f^i \geq r^i$ or a system of constraints is inconsistent, assign $r^{i+1} = r^i$, $i = i + 1$ and go to Step 3.

Let $f^i < r^i$. If the vector x^i meets the demands of integerness of the components x_j^i for $j \in J$, then assign $r^{i+1} = f^i$, $i = i + 1$. If $f^i = f^0$, go to Step 5, otherwise, to Step 3.

Step 2. If $f^i < r^i$, but the integerness conditions are not fulfilled, then for those basic (and superbasic in the nonlinear case) variables x_j, $j \in J$, whose x_j^i is not integer, define the penalties P_j^+, P_j^-. Among them find a minimum penalty $P_{\min}$. If $f^i + P_{\min} \geq r^i$, go to Step 3.

For $f^i + P_{\min} < r^i$ perform branching in the variable x_j, corresponding to a maximum penalty P_j^+ or P_j^-. Choose the least from P_j^+ and P_j^-.

Let $P_j^- \leq P_j^+$, then choose the next problem corresponding to P_j^-. Assign $k = k+1$, put on the list of estimation problems the one, corresponding to the penalty P_j^+. In this case check, whether $f^i + P_j^+ \geq r^i$, then mark the branch, corresponding to P_j^+. Change the upper boundary of the variable x_j, setting it equal to $[x_j^i]$, and go to Step 1.

If $P_j^+ < P_j^-$, then the list stores the problem, corresponding to P_j^-, and in the problem in question the lower boundary of the variable x_j is set equal to $[x_j] + 1$.

Step 3. If $k = 0$, go to Step 5. For $k > 0$, if the branch is marked or $f^i + P^i \geq r^i$, where P^i is the penalty P_j^+ or P_j^-, go to Step 4. Otherwise formulate a problem alternative to the one, formed at Step 2 for $f^i + P_{\min} < r^i$. Mark the corresponding branch and go to Step 1.

Step 4. Set two-sided constraints on the variable x_j, using the information from the list of problems. Assign $k = k - 1$ and go to Step 3.

Step 5. Termination.

1.2. Parallel algorithms. Amongst the processor elements we distinguish the one with zero number, which, in addition to execution of the algorithm with branching and boundaries with one-sided branching carries out some scheduler functions. In the course of solution, some reference data, necessary for arranging parallel computations, are being stored. At the initial stage,

the data about the problem are read from the external memory by zero processor and are transferred to all the rest. Further, zero estimation problem (the original problem with no allowance for integer features) is solved on zero processor, other processors being in expectation.

In the ILP problems, for loading any processor, the indices of basic and nonbasic variables and a part of the list of estimation problems are transferred to its main memory. The values of elements of integer arrays and a part of the list of estimation problems to be transferred are determined by the level k which starts the solution to the first after loading estimation problem. Based on the list of basic variables, a matrix – inverse to the basic one is constructed, and execution of the algorithm with branching and boundaries with one-sided branching starts with the level k. In the IQP problems, it is necessary, in addition, to transfer the array of indices of superbasic variables and a real array of values of superbasic variables.

To regulate the loading of processor elements, an array of pairs (i, k_i) is stored in zero processor, where i is the number of a processor and k_i is the level of an unmarked branch at the i-th processor. At each of the processors an attempt is made to prepare k_i with the least value (which corresponds to the highest unmarked branch at a given processor). If a certain i-th processor is at rest, this is marked with assigning $k_i = -1$.

Let us consider a case when the i-th processor completes the fulfilment of the algorithm — zero level has been attained — or the processor has not been loaded yet. If the number of a processor element $i > 0$, then the i-th processor transfers a call for loading to zero processor. After receiving a call, a pair (j, k_j) with a minimum value $k_j > 0$ is selected at zero processor, and a message about the need of loading the i-th processor is transferred to the j-th processor. As a result, at the j-th processor, a branch of the level k_j is marked, and the loading data are transferred to the i-th processor. If $k_j > 0$ does not exist, then the i-th process is in expectation. When completing the fulfilment of the algorithm on zero processor, the number of the j-th processor for loading is determined in a similar way.

If at a certain processor a new value of the incumbent r is received, this value is transferred to all the other processors. When receiving a new incumbent at each processor, the values f^i and P_j, corresponding to the level k, with which the algorithm fulfilment at a given processor has started, were selected from the list of problems. The condition $f^i + P_j \geq r$ is either equal to P_j, P_j^+ or P_j^- is checked. If this inequality is fulfilled at a certain processor, then at this very processor the fulfilment of the algorithm is completed and the call for loading is done.

The obtained value of r is used for data revision of marked and unmarked branches in the list of problems. It may appear that at a certain processor j an unmarked branch intended for loading some other processor is marked. In this case, the information about a change of k_j is transferred

to zero processor. In order that the data for loading some other processor be prepared in the j-th process, it is needed to make some supplementary calculations, and the value k_j can then have not the highest level among unmarked branches.

The problem has been solved if zero level is attained in all processor elements, or the incumbent value, equal to f^0, is received at a certain processor.

The input data of the problem are transferred by the blocking MPI function `MPI_BCAST`. In the sequel, all processes are carried out asynchronously, and exchanges are done by unblocking functions. The same MPI functions are used for solution to the ILP and the IQP problems.

1.3. Examples of solution to problems. Numerical experiments were carried out on MVS-1000M, 10 processors being employed in the parallel program. Table 1 represents statistical data on solution to the ILP problems. The following notation is used: m is the number of lines in the matrix of constraints; n is a general number of variables among which n_i is that of integer; it_1 is the number of iterations for solution to problems by the sequential algorithm; it_{10} is the total number of iterations fulfilled on all the processors by the parallel algorithm; t_1 is the time (in seconds) needed for solution to a problem by the sequential algorithm; t_{10} is the time needed by the parallel algorithm.

Table 1

Problem	m	n	n_i	it_1	it_{10}	t_1	t_{10}	t_1/t_{10}
1	91	104	58	7547308	2504731	634	31	20.45
2	91	104	58	7547308	2732125	634	44	14.41
3	97	1989	1989	3540594	2136474	1315	104	12.64
4	363	1298	1254	9850308	10108694	3983	584	6.82
5	1248	1224	720	>120000000	126305890	133448	13560	9.82
6	1392	1224	408	19804205	17061570	20551	1887	10.89
7	2176	6000	6000	44597720	25133600	50398	3715	13.57

The first and the second lines in the table present the results of solution to the same problem. Problems of small dimensions (with small values of m) are characterized by essential variations of the total number of iterations and the time needed for problem solution in the parallel version for different program startups.

In Problem 5, an optimum has been found by means of the sequential program, however, the optimality was not substantiated for a given number of iterations.

Table 2 shows the distributions of the number of iterations in estimation problems: the number of iterations needed for solving a null estimation

Table 2

Number of iterations	Problem						
	1	2	3	4	5	6	7
Null estimation	97	97	565	283	995	2043	1066
Average in sequential mode	2.8	2.8	328.4	23.7	28.6	32.1	38.1
Average in parallel mode	2.7	2.8	350.5	24.5	37.7	29.2	26.9

problem, the average number of iterations of solution to estimation problems in the sequential mode, and the same in the parallel mode.

Table 2 explains the reason of large variations in the statistics from Table 1 for the Problems 1 and 2 — the interprocessor exchanges are primary in this problem.

Problem 3 is distinct for large mean values of iterations in estimation problems. This is an example of a problem that is unsuitable for the simplex method. In most of iterations of the simplex method, zero offsets are formed.

When testing the IQP programs, similar relations for sequential and parallel algorithms are obtained. For more detail about programs and their testing the reader is referred to [3, 4].

2. The ILP problems of a special form

The following problems are considered:

$$\text{Minimize} \quad \sum_{j \in J} c_j x_j$$

under the conditions

$$\sum_{j \in J_i} x_j \geq 1, \quad i \in I^1, \qquad \sum_{j \in J_i} x_j = 1, \quad i \in I^2,$$

$$x_j = 0 \text{ or } 1, \quad j \in J,$$

where $J = \{1, \ldots, n\}$, introduce $I = I^1 \cup I^2$, then J_i, $i \in I$, define a subset of the indices J of those variables which participate in the i-th constraint. Denote by I_j a set of constraints indices with participation of the variables x_j, i.e., $I_j = \{i \in I : j \in J_i\}$. It is assumed that $c_j > 0$ for all j, where $I_j \subseteq I^1$. If $I^1 = \emptyset$, then we have a partitioning problem, and for $I^2 = \emptyset$ — a covering problem.

This class of large-dimension problems has long-resisted solution with the help of the ILP algorithms of a general type. The difficulty is in that the basic solutions are strongly degenerate in estimation problems of the linear programming.

Algorithms and software for the given type of problems are constructed on the basis of the methods proposed in [5, 6].

If in the original problem the requirement that $x_j = 0$ or 1 is replaced by the condition $x_j \geq 0$ for all j, then the dual problem to thus obtained one takes the form:

$$\text{Maximize} \quad z_l(u) = \sum_{i \in I} u_i$$

under the conditions

$$\sum_{i \in I_j} u_i \leq c_j, \quad j \in J, \qquad u_i \geq 0, \quad i \in I^1.$$

We introduce the upper $z_u(x)$ and the lower $z_l(u)$ boundaries for the object function in the method of branching and boundaries with branching that we use for solution of problems. Various heuristics are used for obtaining the estimations of boundaries. In this case, of primary importance is maximization of the lower estimation with the help of the subgradient method [7]. In fact, efforts are made to replace solution of estimation problems by application of the subgradient method to the dual problem, whose computer costs are much lower. Using the subgradient method it is required to find a maximum

$$z_l(u) = \sum_{i \in I} u_i + \sum_{j \in J} \min_{x_j = 0 \text{ or } 1} \Big(c_j - \sum_{i \in I_j} u_i \Big) x_j$$

under the conditions $u_i \geq 0$ for $i \in I^1$.

The choice of variables of branching x_j is made based on the ordering by priorities depending on the estimations u_i of the lines of constraints, and j are selected from J_i in a certain sequence. The algorithms do not possess such a parallelism as the method of branches and boundaries with one-sided branching. Here it is appears difficult to gain a uniform loading of all processors in the parallel mode.

Currently, is the OR-Library [8], there are available practical problems with a number of variables exceeding one million and with a number of constraints up to several thousands. Solution to such problems surely demands large computer costs.

3. The search for a global extremum in non-convex quadratic programming

Consider the problem

$$\text{global}\max f(x) = x^T Q x + c^T x \tag{4}$$

on a limited polyhedron given by the conditions

$$Ax = b, \qquad \alpha \leq x \leq \beta. \tag{5}$$

Here A is an $m \times n$ matrix, $b \in R^m$, $c, x, \alpha, \beta \in R^n$, Q is a symmetric non-negative definite $n \times n$ matrix.

Transform problem (4), (5) in the following manner. If Q is a positive definite matrix and LDL^T is its Choletsky factorization, then having changed the variables $L^T x = y$, obtain the problem

$$\text{global}\max \psi(x,y) = y^T D y + c^T x \tag{6}$$

under conditions (5) and

$$x - (L^T)^{-1} y = 0. \tag{7}$$

If Q is a alternating signs matrix, then applying to it the procedure of the modified Choletsky factorization [9], obtain $Q = LDL^T - D_1$, where L is the lower triangular matrix, D is a diagonal matrix with positive entries on the diagonal, D_1 is a diagonal matrix with non-negative elements on the diagonal. After changing the variables $y = L^T x$ obtain the problem

$$\text{global}\max \psi(x,y) = y^T D y - x^T D_1 x + c^T x \tag{8}$$

under conditions (5), (7).

Further, independent of attributes of the matrix Q, find the limit of changing u_i, v_i of the variables y_i, where $y^T = (y_1, y_2, \ldots, y_n)$, having solved $2n$ problems of the linear programming $u_i = \min y_i$, $v_i = \max y_i$ under conditions (5), (7). Denote by G a set, defined by these conditions and on it define majorants of the functions ψ, set in (6) and (8).

The majorant of the function, defined in (6), is a linear function

$$M(x,y) = (u+v)^T D y + c^T x - u^T D v, \tag{9}$$

where $u^T = (u_1, u_2, \ldots, u_n)$, $v^T = (v_1, v_2, \ldots, v_n)$, and the majorant of the function ψ set in (8), is the following quadratic function

$$M(x,y) = (u+v)^T D y + c^T x - x^T D_1 x - u^T D v. \tag{10}$$

The majorants $M(x,y)$ possess the following features. Let $(x^*, y^*) \in G$ be a point, where at least one component y_i^* meets the condition $u_i < y_i^* < v_i$. Consider two subsets of the set G:

$$G_i^- = \big\{(x,y) \in G : \ u_i \le y_i \le y_i^*\big\}, \quad G_i^+ = \big\{(x,y) \in G : \ y_i^* \le y_i \le v_i\big\}.$$

On the sets G_i^-, G_i^+ define the majorants M^-, M^+ of the function ψ by rule (9), if Q is a positive definite matrix, and by rule (10) if Q is a alternating signs matrix. Then it is easy to check that the following statements are valid:

$$\forall (x,y) \in G_i^- \quad M(x,y) - M^-(x,y) = d_i(v_i - y_i^*)(y_i - u_i) \ge 0,$$

$$\forall (x,y) \in G_i^+ \quad M(x,y) - M^+(x,y) = d_i(y_i^* - u_i)(v_i - y_i) \ge 0,$$

$$M^-(x^*, y^*) = M^+(x^*, y^*).$$

Here d_i are diagonal entries of the matrix D.

Based on the above features of majorants, it appears possible to construct an algorithm of the method of branching and boundaries with one-sided branching for the search for a global maximum of the function f on the set G. At the k-th level of the method, a set of admissible solutions G_k is partitioned to the two subsets G_k^- and G_k^+ in such a way that the majorants M_k, M_k^-, M_k^+ of the function ψ on G_k, G_k^-, G_k^+ satisfy, respectively, the conditions $M_k \geq M_k^-$ on G_k^-, $M_k \geq M_k^+$ on G_k^+, $M_k^- = M_k^+$ on $G_k^- \cap G_k^+$. As an object function of the estimation problem of the next level one chooses either M_k^- or M_k^+, therefore if Q is a positive definite matrix, then at each level of the method in question, an estimation problem of the linear programming is solved, or if Q is a alternating signs symmetric matrix, then an estimation problem of the quadratic programming is solved. The algorithm is described in detail in [10].

References

[1] Murtaf B. Modern Linear Programming. Theory and Practice. — Moscow: Mir, 1984.

[2] Kovalev M.M. Discrete Optimization (Integer Programming). — Minsk: Beloruss University, 1977.

[3] Zabinyako G.I., Kotel'nikov E.A. Linear optimization programs // NCC Bulletin. Series: Num. Anal. — Novosibirsk: NCC Publisher, 2002. — Iss. 11. — P. 103–112.

[4] Zabinyako G.I., Kotel'nikov E.A. Parallel algorithm of the integer quadratic programming // Computer Technologies. — 2004. — Vol. 9, No. 1. — P. 34–41.

[5] 5) Fisher M.L., Kedia P. Optimal solution of set covering/partitioning problems using dual heuristics // Manag. Sci. — 1990. — Vol. 36, No. 6. — P. 674–688.

[6] Balas E., Carrera M.C. A dynamic subgradient-based branch and bound procedure for set covering // Oper. Res. — 1996. — Vol. 44, No. 6. — P. 875–890.

[7] Polijak. Minimization of unsmooth functionals // Computational Mathematics and Mathematical Physics. — 1969. — Vol. 9, No. 3. — P. 509–521.

[8] Beasley J.E. OR-Library: distributing test problems by electronic mail // J. Oper. Res. Soc. — 1990. — Vol. 41, No. 11. — P. 1069–1072.

[9] Gill F., Murray W., Right M. Practical Optimization. — Moscow: Mir, 1975.

[10] Kotel'nikov E.A. The search for a global maximum of the quadratic function with linear constraints // Siberian J. Comput. Math. — Novosibirsk, 2004. — Vol. 7, No. 4. — P. 327–334.