Control and Cybernetics

vol. **25** (1996) No. 4

Another simplex-type method for large scale linear programming[1]

by

Jacek Gondzio

Systems Research Institute, Polish Academy of Sciences,
Newelska 6, 01-447 Warsaw, Poland
e-mail: gondzio@ibspan.waw.pl

Abstract: A method is proposed for solving large sparse linear programs. Unlike the well-known simplex method (that makes steps along the edges of polyhedron), the method analysed in this paper takes steps in the directions that belong to the faces of feasible region or cross its interior. More freedom is thus left in their choice. A few remarks concerning the method's implementation are also made. The revised version of the method extensively uses the notion of *working basis*, a nonsingular submatrix of the active part of LP constraint matrix. Throughout the major part of optimization process working basis has size remarkably smaller than the number of constraints, which is beneficial for the efficiency of the method. An experimental implementation of the method is shown to compare favorably with the simplex and primal-dual interior point codes on large set of real-life Netlib test problems.

Keywords: linear programs, large scale optimization, active constraints, working basis.

1. Introduction

We deal in this paper with solving large and sparse linear programming problems

$$\begin{aligned} \text{maximize} \quad & c^T x \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{1}$$

where $c, x \in \mathcal{R}^n, b \in \mathcal{R}^m$ and A is an $m \times n$ matrix.

There exist two computationally attractive approaches to solving (1) i.e. the simplex method (see, e.g., Dantzig, 1963) and the interior point method

[1]This research has been supported by the grant No PB 8 S505 015 05 of the Committee for Scientific Research of Poland.

(see, e.g., Lustig et al., 1992) and several other less efficient or not yet practically verified approaches, some of which have particularly interesting theoretical properties, see e.g.: Cardoso and Climaco (1992), Eiselt and Sandblom (1990), Evtushenko and Zhadan (1992), Fathi and Murty (1989), Gabasov et al. (1984), Korytowski (1990), Mitra et al. (1988) (including many references therein) and Wierzbicki (1993). The reader interested in different variants of the simplex-type pivoting rules is referred to the survey paper of Terlaky and Zhang (1995). Generally, the simplex method goes to the optimum moving along the edges of the polyhedron while the interior point methods pass through its interior.

In this paper another practicable linear programming method is proposed that crosses the interior of the feasible region but stops only on its boundary.

Its key idea is to make steps in the ascent directions that are the *approximate* projections of the objective function gradient onto the face defined by the current set of active constraints or the linear combinations of such directions. Finding an efficient way of computing them constitutes the main difficulty in the method. This is shown to be dominated by the inversion of some full row rank submatrix (*working basis*) of the active constraints matrix A_0. As long as the number of working constraints is remarkably smaller than m, the cost of a single iteration of the method presented is expected to be smaller than that of the simplex method. This is, in particular, the case at the beginning of optimization. We underline that the working basis is a submatrix of the active constraints matrix A_0 and not the normal equations matrix $A_0A_0^T$ that would have to be inverted if the *orthogonal* projections (see, e.g., Rosen, 1961) had to be computed. Even more important, the working basis inverse representation may be inexpensively updated in subsequent iterations of the method. The complexity of such updates is roughly speaking the same as that of the simplex method basis updates, except that the size of the matrix modified is usually smaller.

The approach presented in this paper is closely related to Wolfe's (1963), reduced gradient procedure for convex nonlinear programming. It may also be viewed as an extension of the idea of the projected gradient LP method of Jacquet-Lagreze (1987), a *tableau form* of which was successfully applied to solve some small to medium scale linear programs or as a (revised form) generalization of the adaptive LP method of Gabasov et al. (1984), that, to the best of our knowledge, was implemented only in the context of dense linear algebra and applied to solve only small problems. We underline that we propose a revised version of the method, in implementation of which full advantage of the sparse matrix technology can be taken. The resulting code (experimental and not yet optimized) is shown to compare favorably with the similarly implemented simplex code (using the same approach for handling basis updates).

Let us mention here that the ideas of making steps on the surface of the feasibility region different than those of the simplex method or going through the interior of this region were always present in linear programming techniques. Mitra et al. (1988) give a survey of such methods concluding that most of them remain within the framework of the simplex one. They also discuss computationally attractive approach in which every step through the interior of the feasibility region (supposed to give significant improvement of the objective function) is followed by a *purification procedure* that recovers basic feasible solution.

The method presented in this paper may be viewed as an extension of the one of Mitra et al. (1988) in the sense that we skip their purification procedure and allow steps through the interior of the feasibility region until optimality is reached. Although we lose the nice property of having basic feasible solution, we not only gain a lot of freedom in the choice of next ascent direction, but save time on skipping purification procedure as well. We also operate on smaller working basis than the simplex basis matrix, which makes single iteration of our method less expensive. We pay for it with more involved updates of the working basis in which row and column addition or deletion or row or column exchange are possible. We have thus extended the Schur complement approach of Gondzio (1994a) to handle efficiently three new updates. Additionally, once the problem of exploiting sparsity in the working basis updates has been solved, we are given a lot of freedom in the choice of the strategy to be applied for computing *combined* ascent directions.

The paper is organized as follows. In Section 2, fundamentals of the new approach are addressed. In particular, we concentrate on the method of computing the "best" ascent directions. In Section 3, our algorithm is presented, its convergence is discussed, and some remarks concerning efficient implementation of its logic are given. In Section 4, computational results of the application of the new method to the solution of large scale linear programs from the *Netlib* test collection are presented. Finally, in Section 5 we give our conclusions and discuss possible extensions of the method.

2. Fundamentals of the method

We shall concentrate in this section on the method of computing feasible directions. In particular, we shall show that when compared with the simplex method, remarkably more freedom is left in their choice.

First, let us observe that the linear program (1) differs from the standard MPS formulation (see, e.g., Murtagh, 1981) in which constraints of inequality type are allowed and upper bounds of variables are present. For ease of presentation the latter are omitted.

To simplify the analysis further, we shall assume that m logical (slack, surplus or artificial) variables, each associated with an LP constraint, are added. In this way the constraint matrix gets the form $[A,\ I]$ with a nonsingular diagonal matrix I. We shall take advantage of this particular form when looking for a basis of the null space of the LP constraint matrix in Section 2.1.

We also need an initial feasible solution to start with. Hence, we take any nonnegative x, define a superartificial column $\xi = b - Ax$, and border it to A. The solution $(x, 1)$ is obviously feasible to a new system of linear equations.

As a result of these transformations, we deal with the program

$$\begin{aligned} &\text{maximize} && c^T x + c_L^T s && (2)\\ &\text{subject to} && Ax + s = b, && (3)\\ & && x, s \geq 0, && (4) \end{aligned}$$

in which matrix A has been bordered with a super artificial column ξ (and a new component $-M$ has been added to c) and c_L coefficients are zero for slack or surplus variables associated with the inequality constraints and $-M$ for artificial variables added to the equality constraints. It may easily be shown that if M is sufficiently large, then the optimal solution of (1) may be derived from that of (2)-(4). From now on, we shall denote with n the number of structural variables x in (2)-(4) and with m the number of its constraints.

Let us further suppose that after k iterations of the method a feasible point (x^k, s^k) is determined. Next, we assume that m_B constraints of (3) are active, i.e. slacks associated with them are all zero. These active constraints define submatrix A_0 of matrix A. For simplicity, we assume that rows of A_0 have been reordered to the first m_B positions in A of (3). Additionally, we suppose that some nonsingular $m_B \times m_B$ submatrix of A_0 is given. We shall further call it the working basis B and assume that it is built of the first m_B columns of A_0. Finally, we partition all out of the working basis variables into those which are nonbasic, i.e. lie on their zero bound x_N and strictly positive *superbasic* variables x_S. Summing up, the solution may be partitioned as

$$(x, s) = (x_B, x_N, x_S, s_A, s_N), \tag{5}$$

where $x_B \in \mathcal{R}^{m_B}$ is its basic part, $x_N \in \mathcal{R}^{n_N}$ is its nonbasic part, $x_S \in \mathcal{R}^{n_S}$ is its superbasic part and $s_A \in \mathcal{R}^{m_B}$, $s_N \in \mathcal{R}^{m_N}$ are slack active and nonactive parts, respectively.

We consider all equality constraints and all inequality-type constraints with zero slacks be active, and easily observe that

$$m_B + m_N = m \qquad \text{and} \qquad m_B + n_N + n_S = n. \tag{6}$$

Vector partition (5) determines unique partition of the LP constraint matrix

$$A = \begin{bmatrix} B & N & S \\ C & D & E \end{bmatrix}, \tag{7}$$

where B is an $m_B \times m_B$ nonsingular matrix; all remaining matrices have appropriate dimensions. Let us also observe that current feasible solution (in which we omit superscript k) satisfies

$$x_B \geq 0, \quad x_N = 0, \quad x_S > 0, \quad s_A = 0, \quad s_N \geq 0, \tag{8}$$

and x_B and s_N are strictly positive if degeneracy is not present.

2.1. Feasible directions

Given feasible solution (8), a natural question arises about the feasible directions a step along which will improve the objective function value.

Their construction uses the following scheme. First, we shall expand the set of m rows of (3) with n unit vectors of $\mathcal{R}^{m+n}$ to define a basis of $\mathcal{R}^{m+n}$. Next, we shall build a nonsingular matrix Q of these $m+n$ linearly independent rows and compute its inverse. Selected columns of Q^{-1} will then define a basis of the null space of the LP constraint matrix. Simplex type pricing will, finally, be used to find out which of them are ascent directions.

Let us associate with (7) the following $(m+n) \times (m+n)$ nonsingular matrix

$$Q = \begin{bmatrix} B & N & S & I_{m_B} & \\ C & D & E & & I_{m_N} \\ & I_{n_N} & & & \\ & & I_{n_S} & & \\ & & & I_{m_B} & \end{bmatrix}. \tag{9}$$

It suffices to permute its fifth "block column" to the position two, pushing at the same time the block columns 2, 3 and 4 of one position to the right to obtain an easily invertible, hence nonsingular, block upper triangular matrix. Its first diagonal block will then have the form

$$\tilde{B} = \begin{bmatrix} B & \\ C & I_{m_N} \end{bmatrix}. \tag{10}$$

and will define some simplex method basis for linear program (2)-(4). Observe that $\tilde{B}$ is nonsingular since B is nonsingular. However, it does not have to be a simplex *feasible* basis (it would surely be so if there were no superbasic part present in (7), i.e. if all structural variables that are not associated with the working basis were nonbasic — blocked on their bounds). Let us mention that it is possible to implement the classical revised simplex method with basis matrix represented in an implicit form (10). Powell (1975) implemented such reduced basis approach using PFI (Product Form of Inverse) for handling its updates.

Simple calculations give the following inverse of Q,

$$Q^{-1} = \begin{bmatrix} B^{-1} & & -B^{-1}N & -B^{-1}S & -B^{-1} \\ & & I_{n_N} & & \\ & & & I_{n_S} & \\ & & & & I_{m_B} \\ -CB^{-1} & I_{m_N} & -D+CB^{-1}N & -E+CB^{-1}S & CB^{-1} \end{bmatrix}. \tag{11}$$

Observe that columns of (11) are linearly independent (Q is nonsingular and so is Q^{-1}) and they span $\mathcal{R}^{m+n}$. The jth column of Q^{-1} can be found as

$$q_j = Q^{-1} e_j, \tag{12}$$

where e_j is the jth unit vector of $\mathcal{R}^{m+n}$. Every $q_j, j = 1, 2, ..., m+n$ defines some direction in $\mathcal{R}^{m+n}$. We are concerned with the following two questions:

1. is it a feasible direction for (2)-(4) at the current (feasible) solution?
2. is it an ascent direction?

which shall be answered in this and the following section, respectively.

Every column from the last three blocks of (11) is orthogonal to every row of the first two blocks of Q. In other words, every column of the last three blocks of (11) determines some direction from the null space of matrix $[A, I]$ that defines LP constraints (3). All these directions are obviously linearly independent and span the whole null space of $[A, I]$.

Let us now look closer to the directions defined by (11). For every nonbasic variable x_{Nj}, we determine

$$d_j = \begin{bmatrix} -B^{-1}N_j \\ e_j \\ 0 \\ 0 \\ -D_j + CB^{-1}N_j \end{bmatrix}, \tag{13}$$

a step along which will relax the nonnegativity constraint $x_{Nj} = 0$. For every superbasic variable x_{Sj}, we analogously determine

$$d_j = \begin{bmatrix} -B^{-1}S_j \\ 0 \\ e_j \\ 0 \\ -E_j + CB^{-1}S_j \end{bmatrix}. \tag{14}$$

However, as x_{Sj} is supposed to be strictly positive, both step along d_j and along the negative of d_j are possible. Finally, for every active LP constraint being, say, kth row of A_0, we determine

$$d_k = \begin{bmatrix} -B^{-1}e_k \\ 0 \\ 0 \\ e_k \\ CB^{-1}e_k \end{bmatrix}, \tag{15}$$

a step along which will relax the kth active LP constraint $A_{k.}^T x = b_k$.

It is useful to look a little closer at all these directions and to compare them with the directions along which step is made in the primal and dual variants of the simplex method. If B has size m, i.e. it is some (infeasible in general) simplex basis matrix, (13) and (14) become primal simplex directions and (15) becomes dual simplex direction. If m_B is smaller than m, then (13)-(14) and (15) define respectively primal and dual simplex directions for such a relaxation of the LP problem (2)-(4) in which only current active constraints are taken into account.

The method deals thus with both primal and dual information at a time and, as will be shown in the next section, naturally chooses between "primal" and "dual" directions. This seems to be a nice feature that could never be shared by the simplex method (a switch between primal and dual simplex during the optimization process would in general cause the loss of primal/dual feasibility).

2.2. Ascent directions

The answer to the question whether a step in a given feasible direction q_j ensures the increase of the objective function (2) follows the analysis of the scalar product

$$(c^T,\ c_L^T)\ q_j \tag{16}$$

being the analog of the simplex method's reduced cost (cf. Murtagh, 1981). Partitioning the cost coefficients as the LP variables (5) and applying equation (11) and definition (12), we get

$$\begin{aligned}
(c^T, c_L^T)Q^{-1} &= (c_B^T, c_N^T, c_S^T, c_{LA}^T, c_{LN}^T)Q^{-1} \\
&= (c_B^T B^{-1} + (c_{LA}^T - c_{LA}^T BB^{-1}) - c_{LN}^T CB^{-1},\ c_{LN}^T, \\
&\qquad c_N^T + (-c_{LA}^T N + c_{LA}^T BB^{-1}N) - c_{LN}^T D- \\
&\qquad (c_B^T - c_{LN}^T C)B^{-1}N, \\
&\qquad c_S^T + (-c_{LA}^T S + c_{LA}^T BB^{-1}S) - c_{LN}^T E- \\
&\qquad (c_B^T - c_{LN}^T C)B^{-1}S, \\
&\qquad -(c_B^T B^{-1} - c_{LA}^T BB^{-1} - c_{LN}^T CB^{-1})) \\
&= (c_{LA}^T + (c_B^T - c_L^T \begin{bmatrix} B \\ C \end{bmatrix})B^{-1},\ c_{LN}^T, \\
&\qquad (c_N^T - c_L^T \begin{bmatrix} N \\ D \end{bmatrix}) - (c_B^T - c_L^T \begin{bmatrix} B \\ C \end{bmatrix})B^{-1}N, \\
&\qquad (c_S^T - c_L^T \begin{bmatrix} S \\ E \end{bmatrix}) - (c_B^T - c_L^T \begin{bmatrix} B \\ C \end{bmatrix})B^{-1}S, \\
&\qquad -(c_B^T - c_L^T \begin{bmatrix} B \\ C \end{bmatrix})B^{-1}).
\end{aligned} \tag{17}$$

We shall now introduce some useful notation

$$\tilde{c}^T = c^T - c_L^T A, \tag{18}$$

$$\pi^T = \tilde{c}_B^T B^{-1} = (c_B^T - c_L^T \begin{bmatrix} B \\ C \end{bmatrix})B^{-1}. \tag{19}$$

Observe that (18) defines some new cost coefficients for structural variables of the LP problem that account for the penalty imposed on the violated equality constraints (recall that c_L is zero except for artificials associated with equality constraints). For such modified objective function, (19) defines dual variables associated with the active LP constraints. Equation (17) may now be rewritten in a form easy for computations

$$(c^T, c_L^T)Q^{-1} = (c_{LA}^T + \pi^T,\ c_{LN}^T,\ \tilde{c}_N^T - \pi^T N,\ \tilde{c}_S^T - \pi^T S,\ -\pi^T), \tag{20}$$

and the values $(\tilde{c}_N^T - \pi^T N)_j$, $(\tilde{c}_S^T - \pi^T S)_j$ and $(-\pi^T)_k$ price out directions (13), (14) and (15), respectively. In other words, if they are positive, then the step in the appropriate direction associated with them improves the LP objective function. Observe that for superbasic variables with negative reduced cost the step in the direction negative to that of (14) is possible.

3. Formal statement of the method

The analysis performed in the previous section allows to construct the whole family of methods depending on the rules applied to determine the ascent direction. The first choice would be of course: make step in the direction that ensures the largest possible progress measured with the reduced costs (16) (see also (20)). Formally, this gives Algorithm 1 (Fig.1).

Unfortunately, it is not obvious if a finite convergence result for Algorithm 1 can be established. We may note that there exists a finite (although very large) number of possible partitions (5). They are natural analogs to the basic feasible solutions in the classical simplex algorithm. We know that if the simplex method takes always strictly positive steps θ, then it converges in a finite number of steps (it cannot visit twice the same basic feasible solution and the number of basic feasible solutions is finite).

The presence of nonzero superbasic variables x_S in Algorithm 1 is advantageous from the point of view of the practical efficiency of the method. At the same time it is a serious obstacle in the convergence analysis since, in general, there is an infinite number of possible strictly positive solutions x_B and x_S corresponding to a given partition (5). Consequently, it is not evident if Algorithm 1 cannot visit a given partition more than once.

It is possible to impose additional conditions in Algorithm 1 that enable the proof of its finite convergence. Adding a requirement that there are no superbasic variables in partition (5) is one such condition. In this case, however, the method of Algorithm 1 becomes a classical simplex method operating with bases $\tilde{B}$ (cf. (10)) and the notion of working basis is then exploited uniquely in the implicit inverse representation of $\tilde{B}$.

Another possibility is to introduce additional restrictions that force Algorithm 1 to avoid visiting twice the same working basis (or visiting twice the same partition (5)). While this is an interesting theoretical question, it definitely goes beyond the scope of our practically oriented paper.

Step 0. Initialize

$$k = 0,\ m_B = 0,\ x^0 = (0, t^0),\ s^0 = b - t^0\xi,\ \tilde{c}^T = c^T - c_L^T A.$$

Step 1. Compute dual variables

$$\pi^T = \tilde{c}^T B^{-1}. \tag{21}$$

Step 2. Price out nonbasic and superbasic variables

$$f_N = \tilde{c}_N^T - \pi^T N, \tag{22}$$

$$f_S = \tilde{c}_S^T - \pi^T S. \tag{23}$$

Step 3. Optimality test (and the choice of the ascent direction)

$$(f_N)_{q_1} = \max_{\{j : f_{Nj} > 0\}} f_{Nj}, \tag{24}$$

$$(f_S)_{q_2} = \max_{\{j : f_{Sj} \neq 0\}} |f_{Sj}|, \tag{25}$$

$$f_{q_3} = \max_{\{k : -\pi_k > 0\}} (-\pi_k), \tag{26}$$

$$q = \arg\max \{(f_N)_{q_1},\ (f_S)_{q_2},\ f_{q_3}\}. \tag{27}$$

If $f_q > 0$, then compute direction d_q and go to **Step 4**.
If $f_q = 0$, then stop: optimal solution found.

Step 4. Compute stepsize

$$\theta = \max \{\theta > 0 : x_{k+1} = x_k + \theta d_q \geq 0\}. \tag{28}$$

If $\theta < \infty$, then go to **Step 5**.
If $\theta = \infty$, then stop: unbounded direction found.

Step 5. Make step in direction d_q chosen by (27)

$$x_{k+1} = x_k + \theta d_q. \tag{29}$$

Step 6. Update
Modify matrices B, N, S, C, D and E depending on the ascent direction chosen in **Step 3** and the type of the blocking constraint found in **Step 4**.
$k = k + 1$ and go to **Step 1**.

Figure 1. Algorithm 1

We shall now pass to the brief analysis of computational effort involved in a single iteration of the method. The main difficulty in Algorithm 1 lies in step 6, i.e. the working basis update. Let us then discuss it in more detail.

There are four possible cases of the working basis modifications. They are determined by the type of the direction chosen in step 3 and the type of the blocking constraint found in step 4.

Case 1. The ascent direction was associated with some nonbasic or superbasic variable (see equations (13) and (14)) and the step along it activates as the first some LP constraint from the $[C,\ D,\ E]$ part of (7). Consequently, we have to add the new active constraint to the set of active LP constraints and add the variable q to the set of basic ones. The working basis is thus bordered with a new row and a new column.

Case 2. The ascent direction was associated with some nonbasic or superbasic variable (see equations (13) and (14)) and the step along it activates as the first some nonnegativity constraint of the basic variable $(x_B)_p$. The zeroed basic variable will remain nonbasic in the next iteration and, consequently, must leave the working basis. It is replaced with the variable q. In other words, one column of the working basis is exchanged.

Case 3. The ascent direction was associated with some active LP constraint (direction (15) relaxes this constraint) and the step along it activates as the first some LP constraint from the $[C,\ D,\ E]$ part of (7). Relaxed constraint leaves the set of active constraints and the activated one enters it. The working basis has thus one row exchanged.

Case 4. The ascent direction was associated with some active LP constraint (as in case 3) and the step along it activates as the first some nonnegativity constraint of the basic variable $(x_B)_p$. The deactivated LP constraint must leave the active set and the zeroed variable will remain nonbasic in the next iteration. One row and one column of the working basis have thus to be deleted.

Let us mention that a step along direction (14) may move the superbasic variable to zero, which would not need any working basis update (appropriate variable status has only to be changed).

Summing up, the working basis is subject to one of the following changes in subsequent iterations of Algorithm 1:

- row and column addition,
- column exchange,
- row exchange,
- row and column deletion.

Apart from that, the working basis is used twice at every iteration of the method: to compute dual variables at step 1 and to compute the ascent direction at step 3. The former requires one backward transformation (BTRAN) of

B and the latter needs one forward transformation (FTRAN) of B. (Observe, that whichever direction (13), (14) or (15) has been chosen, the computations may always be organized in such a way that only one equation with B has to be solved). The efficiency of the implementation of Algorithm 1 depends thus strongly on the quality of the method used for working basis inverse representation. The approach applied for this extends the idea of the Schur complement simplex method of Gondzio (1994a), in which the inverse of a large sparse basis matrix has been represented with two other matrices: easily invertible large and sparse fundamental basis built of columns of the LP problem constraint matrix and the small and dense Schur complement that handles the information on the difference between the current and the fundamental bases. For a detailed discussion of this extension the reader is referred to a subsequent paper of Gondzio (1995).

Let us now briefly compare the method presented and the revised version of the simplex method. We quickly conclude that there exist many similarities between them. The main difference, that we expect the most advantages from, is operating on a smaller working basis B. Let us observe that apart from savings obtained in FTRAN and BTRAN operations we also gain on pricing operation performed in step 2 of Algorithm 1 as we do not scan the whole columns of A but only their active parts. An important implementational issue is to organize this step (equations (22) and (23)) row wise rather then column wise as it is the usual practice of the simplex method, see e.g., Murtagh (1981).

The main disadvantage comes from the need of dealing with more involved updates of the working basis. Those, however, are shown in Gondzio (1995) to have comparable complexity to that of the standard column exchange update involved in the simplex method, while smaller size of the working basis is supposed to yield computational savings.

For ease of the method's presentation we have assumed by now that a step is always made along the direction that is the best one in terms of reduced cost (20). We may extend it easily and consider the whole family of methods that use different linear combinations of directions (13), (14) and (15). As long as at most one active constraint is relaxed at a time (i.e. at most one direction of type (15) enters the linear combination), working basis updates remain restricted to the four cases mentioned earlier. Relaxing more than one active LP constraint in a single iteration would inevitably create a need for block updates of the working basis, which we want to avoid.

Let us observe that although all directions defined by (13)-(15) lie on the face of the feasible polyhedron, their linear combinations may enter its relative interior.

Before we pass to the formulation of alternative interior search methods, let us present the approach of Jacquet-Lagreze (1987) in the framework of Section 2. In this method, superbasic part of the ascent direction has always been fixed to

$$d_S = c_S, \tag{30}$$

giving thus the following direction to take step along

$$d = \begin{bmatrix} -B^{-1}Sc_S \\ 0 \\ c_S \\ 0 \\ -Ec_S + CB^{-1}Sc_S \end{bmatrix} = \sum_{j=1}^{n_S} c_{Sj} d_j, \tag{31}$$

where d_j is defined with (14).

It seems promising to combine only the ascent directions taking thus advantage of the available pricing information. Following Mitra et al. (1988), we may for example use the reduced costs as weights of each component direction.

In the following section several different strategies of combining the ascent directions are practically verified.

4. Numerical results

The method analysed in Sections 2 and 3 has been experimentally implemented. We shall present in this section preliminary computational results that demonstrate its efficiency on linear programs from the Netlib test collection of Gay (1985).

Table 1 brings some statistics on the LP tests used in our analysis. Its columns contain: problem name and its size before and after presolve analysis (cf. Gondzio, 1994b): number of constraints m, number of variables (excluding slacks) n and number of nonzero elements of the constraint matrix (excluding objective and right-hand side vectors) *nonz*. The last column of Table 1 contains final density of A. Test problems are ordered by the increasing number of constraints.

We start the analysis from a comparison of three different methods applied to solve test problems.

Primal-Dual. The primal-dual logarithmic barrier interior point method with Mehrotra's (1991) predictor-corrector of order 2. This code — HOPDM version 1.0 of June 1992 — has been shown by Altman and Gondzio (1993a;b) to solve most of the test problems in the number of iterations comparable with that of OB1 of Lustig et al. (1992).

Simplex. The simplex method in which Schur complement updates are used to handle the basis inverse representation. The code has been developed in the context of Marsten's (1981) XMP library. LA05 routines of Reid (1982) implementing Bartels-Golub updates of sparse LU factors were replaced in it with the set of routines of Gondzio (1994a) for handling Schur complement basis inverse. This code is slightly faster than XMP (5-10% on the average).

New. The method analysed in this paper (Algorithm 1) with the Schur complement approach extended by Gondzio (1995) to handle all necessary working basis updates.

Table 2 collects information on the performance of all three methods for all test problems. It contains: number of iterations and the 25MHz SUN SPARC station computation time in seconds (excluding the time for model input and solution output). Empty fields in it mean that the problem was too difficult to be solved by a given method.

Although the **New** code used in the experiments is not yet optimized (some improvements still have to be made in it), the results obtained so far seem encouraging. On most of the problems from the test set our experimental code compares favorably with other two LP solvers. The **New** method solved all except 2 of 56 Netlib problems. In 12 cases this implementation was the fastest of all three codes compared. In 26 cases it was faster than the simplex method. We can thus conclude that the new method is an attractive alternative for other two practicable LP approaches.

Table 3 gives a bit of insight into the method analysed. It containes: problem name, number of LP constraints m, number of equality type constraints m_{EQ}, size of the working basis after crash (see e.g., Gould and Reid, 1989) $m_B(st)$, the largest size of the working basis found in the solution process $m_B(max)$, its size in optimum $m_B(opt)$, and the percentage of all LP constraints that are binding at the optimum. Thus, we may conclude that for considerable part of the problems, the size of the working basis remains smaller than m during the whole solution process. Small size of m_B clearly favours **New** code.

Table 4 collects information on the number of different working basis updates: row and column additions *RCadd*, column exchanges *Cexcng*, row exchanges *Rexcng*, and row and column deletions *RCdel*. The sum of these columns does not necessarily give the number of iterations to reach optimality as they also include pivots that were rejected for stability reasons.

Typical update of the standard simplex method, i.e. column replacement is clearly the most frequent one. Let us also observe that modifications following steps along direction (15) (given in the last two columns of Table 4) seldom exceed 10% of all iterations. They offer, however, to Algorithm 1 the possibility of choosing "dual" steps.

Finally, in Table 6 we report results of solving several larger Netlib tests (see Table 5 for their statistics) with different variants of the new method:

One direction. The new method as stated in Algorithm 1.

k **best.** Its modified version in which at every kth iteration a step is made in a direction that is a linear combination of k best ascent directions (best in terms of reduced costs). The reduced costs were used as weights in the composite direction.

Additionally, as a benchmark, we report results of solving these problems with the CPLEX 2.0 LP optimizer of Bixby (1992). CPLEX is a very efficient commercial simplex code. It uses sophisticated linear algebra dedicated to a specific computer architecture and deeply optimized implementation of the logic of the simplex method, which explains its high performance. Analysis of data from Table 6 shows that our code solves most of the linear programs in a comparable number of iterations but it is roughly speaking about 3-8 times slower than CPLEX (a single iteration in our code costs about 3-5 times more than that of CPLEX). We decided to report these results as, in our opinion, they indicate considerable potential of the new approach.

An important conclusion from an analysis of results collected in Table 6 is that the approach presented in this paper needs relatively low number of iterations to reach optimality. We suppose that this feature is a consequence of the possibility of choice between "primal" and "dual" type directions.

On the other hand, we were surprised that the use of composite directions did not help too much. It improved the method's performance only on GREENBEA and GREENBEB problems. However, as all problems collected in Table 5 except WOODW are highly degenerate, we suppose that our code first needs incorporating better anti-cycling rules before final validating composite directions approach. Another difficulty of applying combined directions arises from larger number of pivot rejections that occur during the solution process. We are now investigating the possibility of removing this drawback.

5. Conclusions

We have presented another computationally attractive approach to solving large scale linear programming problems.

The method belongs to the active set family. It extensively uses the notion of working basis B, a nonsingular submatrix of the active (or, strictly speaking, working) part of LP constraint matrix. The size of matrix B remains usually smaller than m throughout major part of the solution process so a single iteration of this new method may be (when carefully implemented) cheaper than that of the simplex method. Although the computations needed in the method proposed are organized in a simplex-like way (BTRAN, PRICE, FTRAN, UPDATE), the method is definitely a nontrivial extension of the simplex.

It has a useful property of dealing naturally with primal and dual information and can easily choose between primal/dual steps. Additionally, it can pass through the interior of feasible region taking steps along directions that are linear combinations of the most attractive ascent directions.

Its experimental implementation proved to be quite fast and there still exist possibilities of its further improvement. Some useful partial results may, for example, be stored to save on computations.

More generally, we suppose that most of the techniques applied in advanced implementations of the simplex method can probably be incorporated into the method analysed in this paper. This, in particular, applies to implementing steepest-edge pricing (see e.g., Forrest and Goldfarb, 1992), using better anti-degeneracy technique and early detection (and avoiding) of unstable pivots. All these possible improvements will be the subject of our future research.

Additionally, it may be more natural to make a bridge from the primal-dual interior point method to this new method than to the simplex one since we do not have to determine the full row rank basis matrix to start with and accept more that m strictly positive variables at the optimum. Let us also observe that adding new LP constraints or variables (as e.g., feasibility or optimality cuts in decomposition approaches) may naturally be handled in this method.

Acknowledgments

The author is grateful to Professor Tamas Terlaky from Delft University of Technology and Professor Uwe Suhl from the Free University of Berlin for pointing out the Mitra et al. and Powell references, respectively.

The author is grateful to Professor Domingos C. Cardoso from the University of Aveiro for many helpful remarks that improved the final version of this paper.

References

ALTMAN, A. and GONDZIO J. (1993A) An efficient implementation of a higher order primal-dual interior point method for large sparse linear programs. *Archives of Control Sciences*, **2**, 1/2, 23–40.

ALTMAN, A. and GONDZIO J. (1993B) HOPDM - A higher order primal-dual method for large scale linear programming. *European Journal of Operational Research*, **66**, 159–161.

BIXBY, R.E. (1992) Implementing the simplex method: the initial basis. *ORSA Journal on Computing*, **4**, 3, 267–284.

CARDOSO, D.M. and CLIMACO, J.C.N. (1992) The generalized simplex method. *Operations Research Letters*, **12**, 5, 337–348.

DANTZIG, G.B.(1963) *Linear Programming and Extensions.* Princeton University Press, Princeton.

EISELT, H.A. and SANDBLOM, C.-L. (1990) Experiments with external pivoting. *Computers and Operations Research*, **17**, 4 , 325–332.

EVTUSHENKO, JU.G. and ZHADAN, W.G. (1992) *Barrier-projective and barrier-Newton computational methods of optimization (linear programming case).* Technical Report, USSR Academy of Sciences Center of Computations, Moscow, USSR (in Russian).

FATHI, Y. and MURTY, K.G. (1989) Computational behavior of a feasible direction method for linear programming. *European Journal of Operational Research*, **40**, 322–328.

FORREST, J.J. and GOLDFARB, D. (1992) Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, **57**, 341–374.

GABASOV, R., KIRILLOVA, F.M. and TIATIUSHKIN, A.I. (1984) *Constructive Aspects of Optimization, part 1.* Izdatelstvo Universitetskoje, Minsk (in Russian).

GAY, D.M. (1985) Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, **13**, 10-12.

GONDZIO, J. (1994A) On exploiting original problem data in the inverse representation of linear programming bases. *ORSA Journal on Computing*, **6**, 2, 193–206.

GONDZIO, J. (1994B) Presolve analysis of linear programs prior to applying an interior point method, Technical Report 1994.3, Department of Management Studies, University of Geneva. Switzerland, February 1994, revised in December 1994, *ORSA Journal on Computing* (to appear).

GONDZIO, J. (1995) *Applying Schur complements for handling general updates of large sparse unsymmetric matrix, Technical Report ZTSW-2-G244/93*, Systems Research Institute, Warsaw, September 1993, revised April 1995.

GOULD, N.I.M. and REID, J. (1989) New crash procedures for large systems of linear constraints. *Mathematical Programming*, **45**, 475–501.

JACQUET-LAGREZE, E. (1987) *A projected gradient method for linear programming, Technical Report No 79.* LAMSADE, University of Paris Dauphine, June, Paris, France.

KORYTOWSKI, A. (1990) Inner search methods for linear programming. *Applicationes Matematicae*, **20**, 2, 307–327.

LUSTIG, I.J., MARSTEN, R.E. and SHANNO, D.F. (1992) On implementing Mehrotra's predictor–corrector interior point method for linear programming. *SIAM Journal on Optimization*, **2**, 3, 435-449.

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

MEHROTRA, S. (1991) *Higher order methods and their performance, Technical Report 90-16R1.* Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, July 1991.

MITRA, G., TAMIZ, M. and YADEGAR, J. (1988) Experimental investigation of an interior search method within a simplex framework. *Communications of the ACM*, **31**, 1474–1482.

MURTAGH, B. (1981) *Advanced Linear Programming, Computation and Practice.* McGraw-Hill, New York.

POWELL, S. (1975) A development of the product form algorithm for the simplex method using reduced transformation vectors. *Mathematical Programming Study*, **4**, 93–107.

REID, J.K. (1982) A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. *Mathematical Programming*, **24**, 55–69.

ROSEN, J. (1961) The gradient projection method for nonlinear programming, I. Linear constraints. *Journal of SIAM*, **8**, 181–197.

TERLAKY, T. and ZHANG, S. (1995) Pivot rules for linear programming: a survey on recent theoretical developments. *Annals of Operations Research*, **46**, 203-233.

WIERZBICKI, A.P. (1993) *Augmented simplex: a modified and parallel version of simplex method based on multiple objective and subdifferential optimization approach, Technical Report 93-XX*. Institute of Automatic Control, Warsaw University of Technology, Warsaw, July 1993.

WOLFE, P. (1963) Methods for nonlinear programming, in: *Recent Advances in Mathematical Programming*, Graves R.L. and Wolfe P. (eds), McGraw-Hill, New York, 1963.

Problem	Original size			After presolve			
	m	n	nonz	m	n	nonz	density (%)
AFIRO	27	32	83	21	29	72	11.80
ADLITTLE	56	97	383	53	95	365	7.25
SCSD1	77	760	2388	77	760	2388	4.08
RECIPE	91	154	628	71	109	411	5.31
SHARE2B	96	79	694	92	79	632	8.70
SHARE1B	117	225	1151	107	217	990	4.26
SCAGR7	129	140	420	95	138	350	2.67
GROW7	140	301	2612	140	301	2612	6.20
SCSD6	147	1350	4316	147	1350	4316	2.17
FORPLAN	161	418	4494	131	399	4200	8.03
BEACONFD	173	262	3375	72	129	668	7.19
ISRAEL	174	142	2269	163	142	2258	9.76
VTP-BASE	198	186	802	53	83	281	6.39
SC205	205	203	551	203	202	550	1.34
BRANDY	220	249	2148	121	203	1735	7.06
E226	223	282	2578	161	260	2169	5.18
BORE3D	233	314	1427	128	159	615	3.02
CAPRI	271	351	1804	249	331	1535	1.86
GROW15	300	645	5620	300	645	5620	2.90
SCTAP1	300	480	1692	269	452	1557	1.28
BANDM	305	472	2494	245	400	1783	1.82
SCFXM1	330	457	2589	273	430	2335	1.99
STAIR	356	391	3684	356	391	3684	2.64
STANDATA	359	1059	2974	301	954	2627	0.91
SCORPION	388	358	1426	233	295	718	1.04
SCSD8	397	2750	8584	397	2750	8584	0.79
ETAMACRO	400	606	2060	333	541	1849	1.03
SHIP04S	402	1458	4352	241	1291	3823	1.23
SHIP04L	402	2118	6332	317	1915	5695	0.94
PILOT4	410	1058	7119	393	970	6797	1.78
GROW22	440	946	8252	440	946	8252	1.98
STANDMPS	467	1059	3622	403	1026	3143	0.76
SCAGR25	471	500	1554	347	498	1322	0.77
SCRS8	490	1169	3182	447	1131	2827	0.56
SEBA	515	1028	4352	328	334	2280	2.08
FFFFF800	524	854	6227	466	817	5425	1.42
SHELL	536	1525	3056	487	1448	2902	0.41
GFRD-PNC	616	1092	2377	590	1066	2325	0.37
SCFXM2	660	914	5183	546	860	4675	1.00
NESM	662	2748	13078	646	2676	12926	0.75
PILOT-WE	722	2791	9398	703	2593	9049	0.50
SHIP08S	778	2387	7114	326	1632	4795	0.90
SHIP08L	778	4283	12802	520	3149	9346	0.57
25FV47	821	1571	10400	776	1544	10079	0.84
CZPROB	929	3294	9983	689	2770	8337	0.44
PILOT-JA	940	1765	13060	827	1679	12404	0.89
PILOTNOV	975	1968	12186	865	1907	11736	0.71
SCFXM3	990	1371	7777	819	1290	7015	0.66
SCTAP2	1090	1880	6714	977	1768	6159	0.36
SHIP12S	1151	2763	8178	417	1996	5823	0.70
SHIP12L	1151	5427	16170	687	4224	12507	0.43
SIERRA	1227	2016	7252	1135	2016	6968	0.30
GANGES	1309	1681	6912	1120	1489	6396	0.38
PILOT	1441	3449	41092	1384	3382	40725	0.87
SCTAP3	1480	2480	8874	1346	2356	8229	0.26
80BAU3B	2262	9301	20413	2020	8851	19390	0.11

Table 1. Collection of test problems

Problem	Primal-Dual		Simplex		New	
	Iters	Time	Iters	Time	Iters	Time
AFIRO	6	0.52	9	0.08	7	0.12
ADLITTLE	11	1.84	132	1.06	109	1.01
SCSD1	8	3.84	238	2.39	253	4.75
RECIPE	11	2.06	46	0.27	30	0.53
SHARE2B	12	2.36	146	1.64	127	1.59
SHARE1B	21	4.94	330	4.44	194	3.32
SCAGR7	13	2.08	111	1.16	98	1.27
GROW7	13	6.32	725	24.44	206	10.23
SCSD6	10	7.70	640	9.26	789	21.58
FORPLAN	32	17.90	626	11.29	598	17.68
BEACONFD	7	4.32	118	1.86	27	1.34
ISRAEL	28	20.76	362	6.56	181	3.42
VTP-BASE	38	10.12	315	4.52	58	0.69
SC205	10	2.46	50	0.92	152	2.93
BRANDY	19	7.20	405	8.07	364	7.90
E226	21	9.98	866	17.11	588	13.87
BORE3D	18	6.50	464	8.62	45	1.21
CAPRI	20	11.54	342	5.97	309	7.76
GROW15	14	13.20	2006	273.69	365	57.84
SCTAP1	17	7.58	339	5.76	315	7.34
BANDM	17	8.58	669	17.49	377	13.01
SCFXM1	20	11.76	508	11.01	490	13.75
STAIR	18	31.88	1238	75.61	353	34.02
STANDATA	13	9.58	270	5.27	76	2.84
SCORPION	11	4.18	222	6.98	81	2.28
SCSD8	9	13.78	1076	33.23	3196	220.79
ETAMACRO	33	34.32	833	17.31	1394	41.03
SHIP04S	12	10.40	181	3.97	170	5.95
SHIP04L	12	15.53	259	5.93	206	9.60
PILOT4	42	81.93	3530	440.92	2543	276.65
GROW22	16	21.28	-	-	515	177.34
STANDMPS	16	14.74	448	11.66	827	29.46
SCAGR25	17	7.84	623	14.19	522	16.62
SCRS8	23	18.27	884	24.13	697	29.12
SEBA	26	69.88	14	0.85	15	2.21
FFFFF800	39	65.21	612	19.63	738	33.41
SHELL	30	25.09	615	15.34	519	22.31
GFRD-PNC	17	11.21	536	14.85	489	18.55
SCFXM2	26	29.81	958	37.72	998	49.60
NESM	49	169.79	9134	413.34	3172	233.62
PILOT-WE	52	108.90	13738	2014.91	10878	1301.55
SHIP08S	13	17.41	377	13.17	285	11.44
SHIP08L	14	34.00	492	18.04	458	28.58
25FV47	39	149.88	8096	811.57	8798	1175.68
CZPROB	49	80.16	2023	88.14	1427	103.15
PILOT-JA	43	338.08	-	-	-	-
PILOTNOV	25	180.98	-	-	5379	990.68
SCFXM3	24	42.81	1482	83.58	1458	102.16
SCTAP2	14	35.65	1094	48.27	799	46.19
SHIP12S	17	23.78	438	21.18	315	14.80
SHIP12L	16	49.46	903	46.87	526	48.98
SIERRA	22	56.01	1816	109.16	776	47.56
GANGES	18	59.73	807	61.77	1002	85.96
PILOT	77	4210.55	-	-	-	-
SCTAP3	15	51.75	1367	83.13	974	72.74
80BAU3B	54	393.15	-	-	13771	2169.91

Table 2. Comparison of the efficiency of the methods analysed

Problem	m	m_{EQ}	$m_B(st)$	$m_B(max)$	$m_B(opt)$	$\frac{m_B(max)}{m}$ (%)
AFIRO	21	6	6	10	10	48
ADLITTLE	53	14	13	43	43	81
SCSD1	77	77	77	77	77	100
RECIPE	71	50	50	50	50	70
SHARE2B	92	13	13	52	50	54
SHARE1B	107	81	73	89	86	80
SCAGR7	95	83	83	92	86	91
GROW7	140	140	140	140	140	100
SCSD6	147	147	147	147	147	100
FORPLAN	131	89	88	115	112	85
BEACONFD	72	70	70	70	70	97
ISRAEL	163	0	0	67	65	40
VTP-BASE	53	43	43	46	46	87
SC205	203	90	90	192	192	95
BRANDY	121	98	92	112	112	93
E226	161	29	27	115	109	68
BORE3D	128	127	125	127	127	99
CAPRI	249	126	131	211	211	85
GROW15	300	300	300	300	300	100
SCTAP1	269	113	113	196	196	73
BANDM	245	245	240	245	245	100
SCFXM1	273	172	166	226	225	82
STAIR	356	209	208	353	350	98
ANDATA	301	160	160	161	161	53
SCORPION	233	203	203	228	227	97
SCSD8	397	397	396	397	397	100
ETAMACRO	333	207	207	313	309	93
SHIP04S	241	201	201	217	216	90
SHIP04L	317	277	277	289	289	91
PILOT4	393	274	262	375	375	95
GROW22	440	440	440	440	440	100
STANDMPS	403	268	267	293	289	72
SCAGR25	347	299	299	325	310	89
SCRS8	447	353	353	432	425	95
SEBA	328	321	321	326	326	99
FFFFF800	466	313	308	390	389	83
SHELL	487	485	482	486	485	100
GFRD-PNC	590	522	496	558	548	93
SCFXM2	546	344	332	452	451	83
NESM	646	472	472	524	523	81
PILOT-WE	703	566	546	679	679	97
SHIP08S	326	254	254	277	276	85
SHIP08L	520	448	448	469	469	90
25FV47	776	490	484	636	616	79
CZPROB	689	662	662	679	679	99
PILOT-JA	827	591	557	?	?	?
PILOTNOV	865	632	586	833	833	96
SCFXM3	819	516	498	682	680	83
SCTAP2	977	442	442	666	655	67
SHIP12S	417	316	314	354	354	85
SHIP12L	687	586	584	621	621	90
SIERRA	1135	523	503	568	562	50
GANGES	1120	1096	1052	1097	1097	98
PILOT	1384	208	199	?	?	?
SCTAP3	1346	589	589	932	915	68
80BAU3B	2020	0	0	1869	1865	92

Table 3. Changes of the working basis dimension

Problem	iters	RCadd	Cexcng	Rexcng	RCdel
AFIRO	7	4	3	0	0
ADLITTLE	109	38	60	4	9
SCSD1	253	0	253	0	0
RECIPE	30	0	30	0	0
SHARE2B	127	54	52	5	23
SHARE1B	194	26	154	1	15
SCAGR7	98	17	63	4	17
GROW7	206	0	206	0	0
SCSD6	789	0	789	0	0
FORPLAN	598	79	428	28	67
BEACONFD	27	0	27	0	0
ISRAEL	181	81	66	19	17
VTP-BASE	58	4	53	0	1
SC205	152	104	31	14	3
BRANDY	364	36	311	2	21
E226	588	170	288	39	98
BORE3D	45	2	43	0	0
CAPRI	309	104	170	9	29
GROW15	365	0	365	0	0
SCTAP1	315	110	169	9	31
BANDM	377	5	372	0	0
SCFXM1	490	103	329	16	49
STAIR	353	158	164	13	16
STANDATA	76	1	75	0	0
SCORPION	81	27	51	1	3
SCSD8	3196	1	3195	0	0
ETAMACRO	1394	195	1044	59	129
SHIP04S	170	28	128	1	16
SHIP04L	206	19	180	0	11
PILOT4	2543	159	2319	13	51
GROW22	515	0	513	0	0
STANDMPS	827	82	669	13	70
SCAGR25	522	63	381	28	79
SCRS8	697	139	459	32	85
SEBA	15	5	8	2	0
FFFFF800	738	155	486	23	89
SHELL	519	4	514	0	1
GFRD-PNC	489	69	403	0	17
SCFXM2	998	211	654	43	97
NESM	3172	262	2627	56	276
PILOT-WE	10878	591	9737	77	486
SHIP08S	285	37	233	0	23
SHIP08L	458	36	405	2	26
25FV47	8798	1034	6817	154	924
CZPROB	1427	66	1312	0	60
PILOT-JA	-	?	?	?	?
PILOTNOV	5379	401	4971	42	113
SCFXM3	1458	351	884	54	178
SCTAP2	799	279	433	22	74
SHIP12S	315	56	237	5	22
SHIP12L	526	70	408	14	44
SIERRA	776	97	546	4	41
GANGES	1002	46	960	0	0
PILOT	-	?	?	?	?
SCTAP3	974	410	460	23	84
80BAU3B	13771	3688	8047	214	1930

Table 4. Number of different update cases

Problem	Original size			After presolve			
	m	n	nonz	m	n	nonz	density (%)
80BAU3B	2262	9799	21002	2020	8851	19390	0.11
BNL1	643	1175	5121	558	1109	4619	0.74
BNL2	2324	3489	13999	1886	3055	12564	0.22
FIT1P	627	1677	9868	627	1655	9846	0.95
FIT2P	3000	13525	50284	3000	13525	50284	0.12
GREENBEA	2392	5405	30877	1947	4005	23417	0.30
GREENBEB	2392	5405	30877	1940	3988	23346	0.30
STOCFOR2	2157	2031	8343	2129	2015	8255	0.19
WOODW	1098	8405	37474	711	5355	23080	0.61

Table 5. Collection of larger Netlib test problems

Problem	Cplex		One direct.		2 best		3 best		5 best	
	Iters	Time	Iters	Time	Iters	Time	Iters	Time	Iters	Time
80BAU3B	10635	174	13771	1337	14373	1351	14113	1514	14848	1387
BNL1	2320	30	3223	99	3645	121	3521	114	3631	117
BNL2	4563	156	8229	828	9191	942	10412	1050	11813	1302
FIT1P	810	12	629	52	756	62	890	72	789	60
FIT2P	12246	925	14787	6051	16283	6592	17634	6840	17411	6791
GREENBEA	7621	317	19525	2890	17391	2520	15017	2201	14971	2199
GREENBEB	6056	253	15658	2287	15016	2133	13291	1810	13721	1921
STOCFOR2	968	25	873	64	948	74	936	73	983	75
WOODW	1765	28	1179	86	1318	92	1359	93	1353	92

all times are for 33MHz SUN SPARCstation.

Table 6. Comparison of different variants of the new method