European Journal of Operational Research 40 (1989) 322–328
North-Holland

Theory and Methodology

Computational behavior of a feasible direction method for linear programming

Yahya FATHI
Department of Industrial Engineering and Graduate Program in Operations Research, North Carolina State University, Raleigh, NC 27695, USA

Katta G. MURTY *
Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, MI 48109, USA

Abstract: We discuss a finite method of feasible directions for linear programs. The method begins with a BFS (basic feasible solution) and constructs a profitable direction by combining the updated columns of several nonbasic variables eligible to enter. Moving in this direction as far as possible, while retaining feasibility, leads to a point which is not in general a basic solution of the original problem, but corresponds to a BFS of an augmented problem with a new column. So this is called an *interior move* or a *column adding move*. Next we can either carry another interior move, or a *reduction process* which starts with the present feasible solution and leads to a BFS of the original problem with the same or better objective value. We show that interior moves and reduction processes can be mixed in many ways leading to different methods, all of which can be implemented by maintaining the basis inverse or a factorization of it. Results of a computational experiment are presented.

Keywords: Linear programming, feasible direction methods, interior moves, basic feasible solution, reduction process

1. Introduction

We consider the linear program (LP):

$$\text{Minimize} \quad z(x) = cx$$
$$\text{subject to} \quad Ax = b,$$
$$x \geqslant 0,$$

where A is an m by n matrix, b is a column vector of size m, c is a row vector of size n, and $\text{rank}(A) = m$. Let $\boldsymbol{K}$ denote the set of feasible solutions of (1). If E is any matrix, we let $E_{.j}$ denote its j-th column.

Let x_B be a basic vector, associated with the basis matrix B for (1). Denoting the vector of nonbasic variables by x_D, and the matrix of columns associated with them in (1) by D, we can rearrange the variables in (1) and write the equality constraints in (1) as $Bx_B + Dx_D = b$; and the objective function $z(x)$ as $c_Bx_B + c_Dx_D$. The basic solution of (1) corresponding to the basic vector x_B is $\bar{x} = (\bar{x}_B, \bar{x}_D) = (B^{-1}b, 0)$, and it is a BFS of (1) if $B^{-1}b \geqslant 0$, and an optimum solution if $\bar{c}_D = c_D - c_BB^{-1}D \geqslant 0$. $\bar{c}_D$ is the row vector of nonbasic

Received January 1987; revised May 1988
* Partially supported by NSF grant ECS-8521183.

relative cost coefficients in (1) with respect to the basic vector x_B.

If $\bar{x}$ is a BFS and $\bar{c}_D \ngeq 0$, rearrange the nonbasic variables in the vector x_D into two parts, $x_{D,1}$ and $x_{D,2}$, where $x_{D,1}$ consists of all the variables in x_D with relative cost coefficients $\bar{c}_j < 0$, and $x_{D,2}$ are all the other nonbasic variables in x_D. Let (D_1, D_2), $(\bar{c}_{D,1}, \bar{c}_{D,2})$ and $(c_{D,1}, c_{D,2})$ be the partitions of D, $\bar{c}_D$ and c_D, respectively, corresponding to the partition $(x_{D,1}, x_{D,2})$ of x_D. Rather than considering one variable from $x_{D,1}$ as the entering variable, as in the primal simplex algorithm, the feasible direction methods discussed in this paper consider a subset or all of the variables in $D_{D,1}$, and construct a profitable direction to move by taking a nonnegative combination of their updated columns. Let $\omega \geqslant 0$ be a column vector of nonnegative weights of the same dimension as $x_{D,1}$. The direction of movement chosen is

$$y = (y_B, y_{D,1}, y_{D,2}) = (-B^{-1}D_1\omega, \omega, 0).$$

The next point obtained is

$$\bar{x} + \lambda y = (\bar{x}_B + \lambda y_B, \bar{x}_{D,1} + \lambda y_{D,1}, \bar{x}_{D,2} + \lambda y_{D,2}),$$

where the step length λ is given the largest possible value, θ, that keeps the next point nonnegative. In general, when $\theta > 0$, this move from $\bar{x}$ to the new point $\bar{x} + \theta y$ takes us through the relative interior of a face of $\boldsymbol{K}$.

If the new solution $\hat{x} = \bar{x} + \theta y$ is again a BFS of (1), we proceed as before, but in general it will not be a BFS. When $\hat{x}$ is not a BFS of (1), extend the original tableau for (1) by adding a new variable x_{n+1} associated with the original column vector $D_1\omega$ and the original cost coefficient $c_{D,1}\omega$, and use the convention that any feasible solution $(\tilde{x}_B, \tilde{x}_{D,1}, \tilde{x}_{D,2}, \tilde{x}_{n+1})$ for the extended problem *corresponds* to the solution $(\tilde{x}_B, \tilde{x}_{D,1} + \tilde{x}_{n+1}\omega, \tilde{x}_{D,2})$ for the original problem. The move in the original problem from $\bar{x}$ to $\hat{x}$ amounts to bringing x_{n+1} into the basic vector x_B in the extended tableau by means of an ordinary primal simplex pivot step, so that a BFS for the extended tableau is obtained (this BFS corresponds to the feasible solution $\hat{x}$ for the original problem, under the correspondence discussed above) and we can proceed again as before using the extended tableau. Before this pivot step, the relative cost coefficient of x_{n+1} in the extended tableau with respect to the basic vector x_B is of course equal to $\bar{c}_{D,1}\omega$, and it becomes zero after this pivot step is completed. Since the original column and the original cost coefficient of x_{n+1} in the extended tableau are $D_1\omega$ and $c_{D,1}\omega$, respectively, it easily follows that in any basic vector for the extended tableau in which x_{n+1} is a basic variable, the relative cost coefficient of at least one of the original variables in $x_{D,1}$ corresponding to a positive weight in ω, must be nonnegative.

We will call this move from $\bar{x}$ to $\hat{x}$ an *interior move*, or a *column adding move* when it is carried out by extending the tableau with a new variable as discussed above.

When the method is continued in the same manner, new variables will be added, and the tableau will continue to be extended. As soon as one of these new variables leaves the basic vector, it is deleted from the tableau altogether. Because of this, all nonbasic variables at every stage of this process will always be original problem variables in (1).

The above arguement implies that each of these new variables in the current basic vector at any stage of the method corresponds to a positive combination of a different subset of original variables. Also, since any existing new variable in the tableau has to be a basic variable at that stage, the total number of new variables at any stage never exceeds m.

The process of extending the tableau with new variables continues until at some stage either the optimality criterion or the objective unboundedness condition is satisfied for the current augmented problem. If the optimality criterion is satisfied, an optimum solution of the original problem is the one corresponding to the present solution of the current augmented problem. If the objective unboundedness condition is satisfied for the current augmented problem, the objective function in the original problem is unbounded below too. In either case the procedure terminates.

Finite termination of this algorithm has been proved for any choice of values for the weights ω_j as long as the same set of values are used for identical sets of eligible variables throughout the algorithm [7].

The current solution of the extended tableau will always be a BFS of the extended problem at each stage in this algorithm. However, the corresponding solution of the original problem (1) may not be a BFS of (1). If x^+ is the solution of the

original problem (1) corresponding to the current solution at some stage, it is possible to apply a reduction process, beginning with x^+, and obtain a BFS of the original problem (1), $\tilde{x}$, say, satisfying $z(\tilde{x}) \leqslant z(x^+)$, together with a basic vector for (1) associated with it, and start the whole procedure afresh with this basic vector.

In this paper we discuss several possible strategies for mixing the column addition moves and the reduction processes to obtain a variety of methods. We show that each of these methods can be implemented by maintaining the inverse of a basis of order m, or a factorization of it, as in the usual simplex algorithm. In these methods, as long as column adding moves are being carried out, the present basis will always be a basis for the extended tableau at that time, some of its columns being the new columns associated with the new variables at that stage. Whenever the reduction process is carried out, a BFS, say $\bar{x}$, of the original problem will be obtained from the present feasible solution of the original problem, the present basis in the extended problem at that stage will be converted into a basis for the original problem (1) associated with $\tilde{x}$, and all new variables and columns associated with them in the extended tableau will be eliminated.

We also present the results of a computational experiment.

2. The reduction process

At some stage of the algorithm suppose we have an extended tableau with r new variables $x_{n+1}, \ldots, x_{n+r}$. Let $X = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+r})^T$ be the vector of variables in this extended tableau. Let X_B denote the present basic vector and B the corresponding basis in the present extended tableau. Let $\bar{X} = (\bar{x}_1, \ldots, \bar{x}_n, \bar{x}_{n+1}, \ldots, \bar{x}_{n+r})^T$ denote the associated BFS of the present extended problem and let $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n)^T$ be the feasible solution of the original problem corresponding to $\bar{X}$. All the new variables $x_{n+1}, \ldots, x_{n+r}$ are basic variables in X_B.

Let

$\Gamma = \{A_{.j}: 1 \leqslant j \leqslant n$, and $A_{.j}$ is either a column of B or appears with a positive weight in one or more of the columns of new variables in $X_B\}$,

and

$$J(\tilde{x}) = \{j: \tilde{x}_j > 0\}.$$

Clearly for each $j \in J(\tilde{x})$, $A_{.j} \in \Gamma$. The reduction process goes through several steps. In each step at least one column is eliminated from Γ, and changes are made in the present feasible solution for (1) and the present basis, and the basis inverse is updated. The reduction process terminates by either detecting the unboundedness of the objective function in (1), or when the present extended columns are eliminated and the basis consists of original columns in (1) only. We will now describe the first step in this process.

Look for a column $A_{.j}$ for some $j \in J(\tilde{x})$ that appears with a positive weight in the weighted sum corresponding to a column of B associated with a new variable. If no such column exists, X_B must be a degenerate feasible basic vector for the current extended tableau, and in the associated BFS $\bar{X}$ the values of all new variables, namely $\bar{x}_{n+1}, \ldots, \bar{x}_{n+r}$, are zero. This implies that the present feasible solution $\tilde{x}$ is a degenerate BFS of the original problem (1). To convert the present basis B into a basis of the original problem (1) associated with $\tilde{x}$, replace each of the new variables in the present basic vector X_B by an original problem variable in (1) that can replace it, one after another. This takes exactly r degenerate pivot steps, at the end of which we will have a basic vector for the original problem (1) associated with the BFS $\tilde{x}$, and the associated basis inverse; terminate the reduction process.

On the other hand, if one or more such columns exist, select one of them, say $A_{.j_1}$. Let $(\beta_1, \ldots, \beta_m)^T = B^{-1}A_{.j_1}$. Then $A_{.j_1} - B(\beta_1, \ldots, \beta_m)^T = 0$ is a linear dependence relation for the set of vectors Γ. Suppose this relation is $\sum_{j \in \Gamma} \alpha_j A_{.j} = 0$, where $\alpha = (\alpha_j: A_{.j} \in \Gamma) \neq 0$. Define $x(\lambda) = (x_j(\lambda): j = 1, \ldots, n)$, where

$$x_j(\lambda) = \begin{cases} \tilde{x}_j + \lambda\alpha_j & \text{for } j \in \Gamma, \\ 0 & \text{otherwise.} \end{cases}$$

Find θ_1 and θ_2, the minimum and the maximum values of λ that keep $x(\lambda) \geqslant 0$. So $\theta_1 \leqslant 0$ and $\theta_2 \geqslant 0$.

Define $z(\alpha) = \sum(c_j\alpha_j: j \in \Gamma)$. Since $z(x(\lambda)) = z(\tilde{x}) + \lambda z(\alpha)$, if either $\theta_1 = -\infty$ and $z(\alpha) > 0$, or $\theta_2 = +\infty$ and $z(\alpha) < 0$, then $z(x)$ is unbounded

below on $\boldsymbol{K}$ and we terminate. If this unboundedness criterion is not satisfied, choose

$$\theta = \begin{cases} \text{minimum}\{|\theta_1|, |\theta_2|\} & \text{if } z(\alpha) = 0, \\ \theta_1 & \text{if } z(\alpha) > 0, \\ \theta_2 & \text{if } z(\alpha) < 0. \end{cases}$$

It can be verified that $x(\theta) = (x_j(\theta))$ is a feasible solution of (1) satisfying $z(x(\theta)) \leqslant z(\tilde{x})$. θ_1 and θ_2 are the minimum and maximum values of λ that keep $x(\lambda) \geqslant 0$, so they are obtained by the usual minimum or maximum ratio computation as in the simplex algorithm. Let $L \subset \Gamma$ be the subset of $A_{.j} \in \Gamma$ such that j ties in the definition of θ_1 or θ_2, whichever is equal to θ. It can be verified that $x_j(\theta) = 0$ for each j such that $A_{.j} \in L$. We will now drop each of the vectors $A_{.j}$ in L from Γ, one at a time, each time updating B^{-1} appropriately. To drop $A_{.u} \in L$ do the following:

(i) If $A_{.u}$ lies in the weighted sum corresponding to a column in B then let that column be $B_{.v}$ and suppose $B_{.v} = \sum_{j \in \Delta} \omega_j' A_{.j}$, where $\omega_j' > 0$ for each $j \in \Delta$. Let $B'_{.v} = \sum_{(j \in \Delta, j \neq u)} \omega_j' A_{.j}$. If $B'_{.v} = 0$, $z(x)$ is unbounded below on $\boldsymbol{K}$ then terminate. Otherwise, replace $B_{.v}$ by $B'_{.v}$ and update B^{-1} corresponding to this change, as in the revised simplex method, by an appropriate pivot operation. Repeat the same for each column of B corresponding to a new variable that contains $A_{.u}$ with a positive weight.

(ii) If $A_{.u}$ appears as a column of B by itself, there must be a column $B_{.v}$ of B corresponding to a new variable, for which $\beta_v \neq 0$. Suppose $B_{.v} = \sum_{j \in \Delta} \omega' A_{.j}$. Replace $A_{.u}$ in B by $A_{.s}$ for some $s \in \Delta$, say, and then replace $B_{.v}$ by $B'_{.v} = \sum_{(j \in \Delta, j \neq s)} \omega_j' A_{.j}$, and update B^{-1} accordingly by the appropriate pivot steps.

When this work is completed, drop $A_{.u}$ from Γ. If all the columns in the current matrix B are individual columns of A in (1), B is a basis for (1), and the present feasible solution $x(\theta)$ must be a BFS of (1) corresponding to it, and it satisfies $z(x(\theta)) \leqslant z(\tilde{x})$ then terminate the reduction process. Otherwise, with the current Γ, B, its inverse, and the feasible solution of (1) corresponding to them, $x(\theta)$, go back to another step in the reduction process.

So, this whole reduction process of moving from $\tilde{x}$ to a BFS of (1) with the same or better objective value can be carried out using pivot steps and updating the basis inverse as in the usual revised simplex algorithm.

3. The choice of weights in a column adding move

At some stage of this algorithm, let B be the present feasible basis associated with the basic vector X_B for the current extended tableau. So, some of the basic variables in X_B may be new variables introduced in earlier stages. Let X_D be the vector of nonbasic variables at this stage. We know that all these nonbasic variables are original problem variables in (1). Let $\bar{c}_D$ be the vector of nonbasic relative cost coefficients in the present tableau. $\bar{X} = (\bar{X}_B = B^{-1}b, \bar{X}_D = 0)$ is the present BFS of the extended tableau, and let $\tilde{x}$ be the feasible solution of the original problem (1) corresponding to it. If $\bar{c}_D \geqslant 0$, let $(X_{D,1}, X_{D,2})$ be the partition of the nonbasic vector X_D with the corresponding partition $(\bar{c}_{D,1}, \bar{c}_{D,2})$ of $\bar{c}_D$ such that $\bar{c}_{D,1} < 0$ and $\bar{c}_{D,2} \geqslant 0$. Our feasible direction method requires the weight vector $\omega_{D,1} \geqslant 0$ corresponding to $X_{D,1}$, to determine the new column to be introduced into the tableau at this stage. The overall computational requirements of the methods depend on the choice of weights used in each iteration of the algorithm. In determining the value for the weights ω_j we have the freedom to select any subset of the eligible variables (those in $X_{D,1}$) at this stage and set their weights at positive levels. We call these variables *the entering variables*, and denote their set of subscripts by S. The rest of the variables in $X_{D,1}$ are ignored during this particular movement, i.e., their weights are set equal to zero. A particular strategy that we have used in our computational experiments is to limit the number of entering variables in an iteration to some preselected positive integer p. In this strategy, whenever the number of eligible variables exceeds p, we simply pick the p among them with the most negative relative cost coefficients to be the entering variables. In our experiments we used the following choice rules to obtain the weights ω_j, for the entering variables. Here, for each $j \in S$, $\bar{c}_j$ denotes the present relative cost coefficient of x_j and $\bar{A}_{.j} = (\bar{a}_{ij}: i = 1, \ldots, m) = B^{-1}A_{.j}$, the present updated column of x_j.

Rule 1: $\omega_j = 1$ for all $j \in S$.

Rule 2: $\omega_j = -\bar{c}_j$ for all $j \in S$.

Rule 3: $\omega_j = -\bar{c}_j$ (minimum$\{\bar{b}_i/\bar{a}_{ij}: i$ such that $\bar{a}_{ij} > 0\}$) for all $j \in S$.

Rule 2 is similar to the weights chosen in the reduced gradient method [8]. Rule 3 is computationally expensive but could lead to better directions of movement.

We will use the notation p/q to define the strategy employed in an implementation, p denoting the maximum number of eligible variables allowed to enter in each iteration, and q indicates the rule used to determine the weights ω_j, for the selected entering variables ($q = 1$, 2 or 3). Notice that the 1/1 startegy is the one used by the well known simplex algorithm.

4. The feasible direction methods

4.1. *The pure interior strategy: The column adding method*

In the implementation using this strategy, in each iteration the tableau is extended by adding a new variable whose column is the weighted combination of the columns of the entering variables selected in that iteration, until at some stage one of the two termination conditions, optimality or unboundedness, is satisfied.

4.2. *Mixed strategies*

This implementation begins with a BFS of the original problem (1), and starts in the same manner as the column adding method. After some selected number of iterations of this process, or after a certain prespecified condition is satisfied (we will call this condition G; several possible choices for this are discussed later), we stop the column adding process and identify the feasible solution $\tilde{x}$ of the original problem (1) corresponding to the present BFS of the current augmented problem. We then carry out the reduction process discussed in Section 2, beginning with $\tilde{x}$. This process terminates with either discovering that $z(x)$ in unbounded below on $\boldsymbol{K}$, or with a BFS $\bar{x}$ for (1), and the inverse of an associated basis B for (1), satisfying $z(\bar{x}) \leqslant z(\tilde{x})$. We resume the column adding process with the feasible basis B and the associated BFS $\bar{x}$. Finite termination of this implementation can also be guaranteed [7].

Condition G, which determines the time to stop the interior movements and start a reduction process, has a major impact on the overall computational requirements (measured by the total number of pivot steps before termination) of this method. We now present the alternative strategies for condition G used in our computational experiments (these are denoted by G_1, G_2 and G_3).

G_1–Pure interior strategy. This is the strategy discussed above. This strategy never uses the reduction process and proceeds with the column adding method until termination.

G_2–Persistent reduction strategy. Every interior movement is followed by a reduction process. In this strategy each iteration begins with a BFS of (1), goes through an interior move and a reduction process and ends with a BFS of (1) if the unboundedness condition is not satisfied.

G_3–Strategy based on objective reduction. This strategy consists of consecutive interior moves until an interior move is made during which the value of the objective function does not make a 'substantial improvement'. At this time the algorithm stops the interior moves and starts a reduction process to obtain a BFS of the original problem (or determine that the problem is unbounded). The term 'substantial improvement' is open to interpretation and could be defined in a number of ways. A definition we used in our experiments is the following: Let z_1 be the value of the objective function before an interior move and z_2 be its value after the move. We define the result of the move a 'substantial improvement' if $|z_1 - z_2| > t|z_1|$.

In our experiments we used three different values for t, which are 0.1, 0.01 and 0.001, respectively. In any future reference to this strategy we use the notation $G_3(t)$ to emphasize the dependence of this strategy on the parameter t. This definition and these particular values of t happen to be appropriate for the models under consideration in our experiments. The problem solver can provide any other appropriate definition of 'substantial improvement' for the specific problem under consideration.

5. Computational experiments

From the discussions in the previous sections it is clear that certain algorithmic strategies and parameter values must be specified prior to using the feasible direction method to solve a linear programming problem. We have conducted a

limited series of computational experiments to study the impact of different strategies and parameter values on the overall computational requirements of the algorithm. This section contains the results of these experiments. All the programs were written by the authors in FORTRAN (version 4.3) and ran on a VAX 11/750 computer in the Department of Industrial Engineering at North Carolina State University.

In the context of our discussions, an algorithm is completely defined by specifying its interior-reduction movement strategy (G_1 through G_3 as discussed in Section 4.2) and its choice of weights (p/q as discussed in Section 3). We combine these notations to specify a particular algorithm as $G/p/q$. For instance, $G_2/5/3$ represents an algorithm that performs a reduction operation after every interior movement, uses no more than 5 columns of the matrix A to generate a new column, and uses Rule 3 to determine the weights ω_j. Notice that $G_1/1/1$ represents the standard simplex method.

5.1. Test problems

We have used two different linear programming models in these experiments. We refer to these models as Test Problem 1 and Test Problem 2, respectively. Following is a brief description of each of these models.

Test Problem 1. Following Cirina [2], we have adopted the LP model used by Khun and Quandt [5] and by Avis and Chvatal [1] as our Test Problem 1. We have considered the following problem:

$$\text{Maximize} \quad z = \sum_{j=1}^{n} x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij}x_j \leqslant 10^4, \quad i = 1,\ldots,m,$$

$$x_j \geqslant 0, \quad j = 1,\ldots,n,$$

with m taken from $\{10, 20, 30\}$ and n taken from $\{10, 20, 30, 40\}$ with $m \leqslant n$ and a_{ij} taken at random from the set $\{1, 2,\ldots,1000\}$.

Test Problem 2. This model is a variation of the generalized transportation problem as follows:

$$\text{Maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}$$

$$\text{subject to} \quad \sum_{i=1}^{m} a_{ij}x_{ij} \leqslant b_j, \quad j = 1,\ldots,n,$$

$$\sum_{j=1}^{n} x_{ij} \leqslant r_i, \quad i = 1,\ldots,m,$$

$$x_{ij} \geqslant 0, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n,$$

with m and n taken from $\{4, 6, 8\}$, and a_{ij}, b_j and r_i taken at random from the sets $\{1, 2,\ldots,20\}$, $\{10m, 11m,\ldots,100m\}$ and $\{2n, 3n,\ldots,10n\}$, respectively.

5.2. Experiments and the results

For each test problem, and for each selected combination of m and n (9 combinations for each test problem), we generated 10 to 20 different problems at random; each problem was then solved by the standard simplex method ($G_1/1/1$) as well as by a variety of other strategies.

During the course of these experiments it became clear that the overall computational requirements of the algorithm strongly depends on the choice of the parameter values and algorithmic strategies used. Certain strategies proved to be quite inefficient for all instances of a particular model (notably $G_1/n/2$ and $G_1/n/3$ for Test Problem 1), while certain other strategies showed promise of being more efficient than the standard simplex method. During these experiments we also discovered considerable fluctuations in the overall computational requirements of each strategy on different problems of the same size and similar structure in our test series. Some strategies, although very efficient on some instances, were quite inefficient on others. $G_2/p/2$ and $G_2/p/3$ with the values of p between 2 and 5 were in this category.

For each of the two models, however, we discovered certain strategies that showed relatively consistent behavior across the entire set of problems of that model in the test series. For Test Problem 1, strategies $G_3/p/2$ and $G_3/p/3$ with $2 \leqslant p \leqslant 5$ were consistent performers, with their average number of pivot operations about 15%

Table 1
Average number of pivot operations required by different algorithms on 20 randomly generated problems in each size. (Test Problem 1)

Size m	10	10	10	10	20	20	20	30	30
Strategy n	10	20	30	40	20	30	40	30	40
$G_1/1/1$ (simplex)	9.8	18.0	25.5	26.2	32.1	30.1	39.6	58.2	62.5
$G_3(0.01)/2/3$	10.8	16.2	20.3	32.8	29.0	27.0	35.2	47.1	54.1
$G_3(0.01)/3/3$	11.2	14.7	23.4	25.0	33.0	32.2	34.1	50.4	56.6

Table 2
Average number of pivot operations required by different algorithms on 10 randomly generated problems in each size. (Test Problem 2)

Size m	4	4	4	6	6	6	8	8	8
Strategy n	4	6	8	4	6	8	4	6	8
$G_1/1/1$ (simplex)	9.5	11.2	12.5	11.5	12.4	13.6	14.3	14.0	18.0
$G_1/4/3$	4.3	6.7	9.0	8.0	7.4	8.2	8.7	8.0	8.3
$G_1/6/3$	5.8	5.3	11.3	9.8	6.0	8.0	9.3	8.0	8.4

less than that of the standard simplex method on the problems in the test series. For Test Problem 2, strategies $G_1/p/3$ for the values of p between 4 and 6 were the consistent performers, with their average number of pivot operations about 40% less than that of the standard simplex method on the problems in the test series. It should be noted that, due to the special structure of the latter model, most of the problems in this group that we generated had multiple optimal solutions. In all these cases the interior algorithm terminated with a nonbasic optimal solution of the problem.

Table 1 shows the average number of pivot operations for a few selected strategies (including the $G_1/1/1$, of course) on 20 different problems in each of the 9 sizes under consideration for Test Problem 1. Table 2 contains similar information for Test Problem 2.

We expect the savings to be more pronounced on larger problems.

5.3. Other experiments

The diversity of the strategies that can be used in the feasible direction method makes a comprehensive experimental study of this algorithm quite extensive, specially since the computational requirements of different strategies seem to depend on the structure of the problem under investigation. As of now, aside from our own experiments, we are aware of two other experimental studies. In [6] Mitra et al. report on a variety of G_2 type strategies, although they use a different reduction technique. The results of their investigations indicate that for the problems in their experiments, especially for the larger problems, certain strategies of the feasible direction method are superior to the standard simplex method. In [4] Eislet et al. report similar results. The strategies they use, however, involve a return to the standard simplex method after a certain amount of interior movements. For details see [3].

References

[1] Avis, D., and Chvátal, V., "Notes on Bland's pivoting rule", *Mathematical Programming Study* 8 (1978) 24–30.
[2] Cirina, M., "Remarks on a recent simplex pivoting rule", *Rapporto* 7/4(1), Dipartimento di Informatica, University di Torino, Torino, Italy, 1984.
[3] Eislet, H.A., and Sandblom, C.L., "External pivoting in the primal simplex algorithm", Technical Report, Department of Quantitative Methods, Faculty of Commerce and Administration, Concordia University, 1984.
[4] Eislet, H.A., Sandblom, C.L., and DeMarr, R., "Computational experience with external pivoting", *COAL Newsletter* 12 (1985) 16–20.
[5] Kuhn, H.W., and Quandt, R.E., "An experimental study of the simplex method", *Proceedings of Symposia in Applied Mathematics* 15 (1962) 107–124.
[6] Mitra, G., Tamiz, M., and Yadegar, J., "Experimental investigation of an interior search method within a simplex framework", Technical Report 6/86, Department of Mathematics and Statistics, Brunel University, 1986.
[7] Murty, K.G., and Fathi, Y., "A feasible direction method for linear programming", *Operations Research Letters* 3/3 (1984) 121–127.
[8] Wolfe, P., "Methods of nonlinear programming", in: R.L. Graves and P. Wolfe (eds.), *Recent Advances in Mathematical Programming*, McGraw-Hill, New York, 1963.