SIAM J. OPTIMIZATION
Vol. 2, No. 3, pp. 435–449, August 1992

ON IMPLEMENTING MEHROTRA'S PREDICTOR–CORRECTOR INTERIOR-POINT METHOD FOR LINEAR PROGRAMMING*

IRVIN J. LUSTIG[†], ROY E. MARSTEN[‡], AND DAVID F. SHANNO[§]

Abstract. Mehrotra [*Tech. Report* 90-03, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1990] recently described a predictor–corrector variant of the primal–dual interior-point algorithm for linear programming. This paper describes a full implementation of this algorithm, with extensions for solving problems with free variables and problems with bounds on primal variables. Computational results on the NETLIB test set are given to show that this new method almost always improves the performance of the primal–dual algorithm and that the improvement increases dramatically as the size and complexity of the problem increases. A numerical instability in using Schur complements to remove dense columns is identified, and a numerical remedy is given.

Key words. linear programming, interior-point methods, predictor–corrector algorithms

AMS(MOS) subject classifications. 90C05, 90C06

1. Introduction.

Mehrotra [10] has recently introduced a remarkable higher-order primal–dual logarithmic barrier method for linear programming. He motivates this method as a power series method, but in a nonstandard way. He also introduces a potential function that can be linearly searched to ensure a constant reduction at each step. A numerical algorithm using this method is described, and computational comparisons are given to MINOS 5.3 and OB1, the primal–dual interior-point code of Lustig, Marsten, and Shanno [8]. These new results indicate that the method represents a significant computational advance for linear programming.

The test set that Mehrotra used to evaluate his method is a small subset of the NETLIB [6] test set, excluding all problems with upper-bounded variables, free variables, and ranges. Since most of the large and difficult problems in the NETLIB test set were omitted from Mehrotra's tests, the initial goal of this paper is to describe an implementation extending Mehrotra's algorithm to solve all standard form linear programs. Mehrotra's initial results tend to sell his method short, because its performance, when compared with that of the pure primal–dual algorithm, improves even more as the problems get larger and more complex.

A second goal of this paper is to isolate the effects of various new components of Mehrotra's framework. Mehrotra discusses the results of his code compared only with OB1. However, because OB1 has many safeguards (such as iterative refinement) needed to solve large ill-conditioned problems, and because OB1 incurs overhead to

*Received by the editors May 25, 1990; accepted for publication (in revised form) August 20, 1991.

[†]Department of Civil Engineering and Operations Research, Princeton University, Princeton, New Jersey 08544.

[‡]School of Industrial Engineering and Operations Research, Georgia Institute of Technology, Atlanta, Georgia 30332. The research of this author is partially supported by Office of Naval Research grant N00014-88-K-0104.

[§]Rutgers Center for Operations Research, Rutgers University, New Brunswick, New Jersey 08903. The author is currently a Visiting Fellow at the Department of Civil Engineering and Operations Research, Princeton University, Princeton, New Jersey 08544. The research of this author is sponsored by the U.S. Air Force Office of Scientific Research, Air Force Systems Command, under grant AFOSR-87-0215.

handle upper bounds (which are always assumed to be present), direct comparisons of the two codes and evaluations of the relative effects of components of the algorithm require a common implementation. Once this was accomplished within the framework of OB1, it became easy to compare Mehrotra's algorithms more accurately with the primal–dual algorithms of OB1.

Another purpose of this paper is to develop Mehrotra's algorithm, including bounds, ranges, and free variables, within the framework of a predictor–corrector method. Although Mehrotra's paper [10] uses a power series motivation, Mehrotra himself gave a different motivation for the method in a presentation at a conference in Asilomar. In this presentation, he described a method drawn from predictor–corrector methods for ordinary differential equations. This interpretation is particularly seminal in motivating effective higher-order methods and is extremely clear geometrically in showing how the methods incorporate higher-order information. Therefore, our paper uses the predictor–corrector motivation.

Finally, since we are concerned with solving large, difficult problems, we again confront the enigma of removing dense columns. This is essential for solving two new problems, `fit1p` and `fit2p`, of the NETLIB test set within our memory constraints. We identify an instability that arises from using Schur complements to accomplish this task and suggest a remedy that has worked well in practice.

Section 2 briefly introduces the primal–dual algorithm OB1, with some minor improvements. Section 3 derives the predictor–corrector method for bounded problems and discusses some of the difficulties that had to be overcome. Section 4 deals with dense column removal, and §5 gives computational results indicating that the predictor–corrector method is indeed a major advance in the state of the computational art.

2. The primal–dual method for linear programming.

Lustig, Marsten, and Shanno [8] describe in detail a path-following primal–dual interior-point method for linear programming derived from a logarithmic barrier method. Here we briefly recapitulate the algorithm both for the convenience of the reader and because the first-order conditions will be required to motivate subsequent work on the new primal–dual predictor–corrector method. The primal problem we are concerned with here is

$$\min c^T x \qquad \text{subject to} \qquad Ax = b, \qquad 0 \le x \le u, \tag{1}$$

where some or all of the upper bounds may be infinite. We assume that $A \in \Re^{m\times n}$, $b \in \Re^m$, $c \in \Re^n$, $u \in \Re^n$, and $x \in \Re^n$. Adding slack variables s to transform the upper-bound inequalities to equalities and eliminating the inequality constraints by incorporating them in a logarithmic barrier term appended to the objective function, the Lagrangian for (1) becomes

$$L(x,s,y,w,\mu) = c^T x - \mu \sum_{j=1}^{n} \ln x_j - \mu \sum_{j=1}^{n} \ln s_j - y^T(Ax-b) - w^T(u-x-s). \tag{2}$$

The first-order necessary conditions for a stationary point of (2) are

$$\begin{aligned} Ax &= b, \\ x + s &= u, \\ A^T y - w + z &= c, \\ XZe &= \mu e, \\ SWe &= \mu e, \end{aligned} \tag{3}$$

where X, Z, S, and W are diagonal matrices with the elements x_j, z_j, s_j, and w_j, respectively, and $z \in \Re^n$ is the vector of the dual slack variables.

The primal–dual algorithm is derived from the first-order conditions (3) by applying one iteration of Newton's method to find an approximate solution to (3) for a fixed value of μ and continuing until the complementarity $x^T z + s^T w$ is reduced to a predetermined tolerance. In [8] this algorithm was implemented specifically by assuming that a current estimate x, y, z, w, s was available satisfying $x > 0, z > 0, w > 0$, and $s > 0$, with the further restriction that $x + s = u$, i.e., that the upper bounds were always explicitly satisfied. Fixing μ and applying one step of Newton's method to (3) yields the set of equations

$$\begin{aligned} A\Delta x &= b - Ax, \\ \Delta x + \Delta s &= 0, \\ A^T\Delta y + \Delta z - \Delta w &= c - A^T y - z + w, \\ Z\Delta x + X\Delta z &= -XZe + \mu e, \\ W\Delta s + S\Delta w &= -SWe + \mu e. \end{aligned} \tag{4}$$

Defining

$$\Theta = (X^{-1}Z + S^{-1}W)^{-1} \tag{5}$$

and

$$\rho(\mu) = \mu(S^{-1} - X^{-1})e - (W - Z)e, \tag{6}$$

the solution to (4) is

$$\begin{aligned} \Delta y &= (A\Theta A^T)^{-1}[(b - Ax) + A\Theta((c - A^T y - z + w) + \rho(\mu))], \\ \Delta x &= \Theta[A^T\Delta y - \rho(\mu) - (c - A^T y - z + w)], \\ \Delta z &= \mu X^{-1}e - Ze - X^{-1}Z\Delta x, \\ \Delta w &= \mu S^{-1}e - We + S^{-1}W\Delta x, \\ \Delta s &= -\Delta x. \end{aligned} \tag{7}$$

A new point x^*, s^*, y^*, z^*, w^* is then defined by

$$\begin{aligned} x^* &= x + \alpha_P \Delta x, \\ s^* &= s + \alpha_P \Delta s, \\ y^* &= y + \alpha_D \Delta y, \\ z^* &= z + \alpha_D \Delta z, \\ w^* &= w + \alpha_D \Delta w, \end{aligned} \tag{8}$$

where α_P and α_D are respective step lengths in the primal and dual spaces chosen to assure the nonnegativity of the variables x, s, z, and w. At each step the barrier parameter μ is reduced by a method discussed below, and the algorithm continues until the relative duality gap satisfies

$$\frac{c^T x - b^T y + u^T w}{1 + |b^T y - u^T w|} < \epsilon \tag{9}$$

for a user-predetermined ϵ.

In [8] μ is chosen at each step to be

$$\mu = \frac{c^Tx - b^Ty + u^Tw + M\delta_1 + M\delta_2}{\phi(n)}, \tag{10}$$

where

$$\delta_1 = \|b - Ax\| / \|b - Ax^0\|, \tag{11}$$

$$\delta_2 = \|c - A^Ty - z + w\| / \|c - A^Ty^0 - z^0 + w^0\|, \tag{12}$$

$$\phi(n) = \begin{cases} n^2, & \text{if } n \leq 5000, \\ n^{3/2}, & \text{if } n > 5000, \end{cases} \tag{13}$$

and M is an appropriately chosen large constant, which in [8] was given by

$$M = \xi\phi(n) \max\{\max_{1 \leq j \leq n} \{|c_j|\}, \max_{1 \leq i \leq n} \{|b_i|\}\}. \tag{14}$$

A somewhat complex algorithm for choosing x^0, y^0, z^0, and ξ is documented in [8]. In the numerical results given in [8], it was noted that for these choices of x^0, y^0, z^0, and ξ, 69 of the 71 problems of the NETLIB test set tested in [8] converged to eight digits of accuracy, whereas `pilot4` and `capri` required ξ to be reduced in order to obtain that degree of accuracy.

The algorithm corresponding to (7) and (8) that we use to compare with the new predictor–corrector algorithm modifies the ξ of [8] to achieve eight digits of accuracy on all problems with the default settings. Here we choose ξ initially, as in [8], and compute the initial search vector. Denoting this vector by

$$\begin{aligned} \Delta x &= \Delta x_1 + \mu\Delta x_2, \\ \Delta y &= \Delta y_1 + \mu\Delta y_2, \\ \Delta z &= \Delta z_1 + \mu\Delta z_2, \\ \Delta w &= \Delta w_1 + \mu\Delta w_2, \end{aligned} \tag{15}$$

we compute the vector norms

$$\Delta_1 = \|\Delta x_1 + \Delta y_1 + \Delta z_1 + \Delta w_1\|$$

and

$$\Delta_2 = \|\Delta x_2 + \Delta y_2 + \Delta z_2 + \Delta w_2\|. \tag{16}$$

Then, if $\mu\Delta_2 < 0.7\Delta_1$, we increase ξ by a factor of 10, and if $10\Delta_1 < \mu\Delta_2$, we decrease ξ by a factor of 10. Thus, ξ is dynamically adjusted to attempt to bring the lengths of the optimality–feasibility vector Δ_1 and the centering vector Δ_2 to approximately equal magnitudes, where in (16) $\|\cdot\|$ is the l_1 norm.

This readjustment of ξ allowed all previously tested problems, plus 14 of the 15 additional problems recently added to the NETLIB test set, to be solved to 8 digits of accuracy using the default options. Moreover, a comparison between the iteration counts given in §4 for this method and the counts of [8] shows that overall the method is not only more stable, but more efficient. Thus, if a pure primal–dual method is desired, it seems important to choose the initial μ to roughly equate the initial norms of the vectors Δ_1 and Δ_2. However, because of the clear numerical dominance of the method that is documented in the next section, the importance of the pure primal–dual algorithm is questionable at best.

3. Mehrotra's predictor–corrector method. Mehrotra [10] introduces a power series variant of the primal–dual algorithm without considering explicit bounds. This algorithm was initially described by Mehrotra [9] without any given motivation. Here we derive a version of the method described in [9] including bounded variables but, as previously noted, using the predictor–corrector motivation. This algorithm can also be viewed as an extension of one of the algorithms presented by Mehrotra in [10]. The method again uses the logarithmic barrier Lagrangian (2) to derive the first-order conditions (3). Rather than applying Newton's method to (3) to generate correction terms to the current estimate, we substitute the new point into (3) directly, yielding

$$\begin{aligned} A(x+\Delta x) &= b, \\ (x+\Delta x)+(s+\Delta s) &= u, \\ A^T(y+\Delta y)-(w+\Delta w)+(z+\Delta z) &= c, \\ (X+\Delta X)(Z+\Delta Z)e &= \mu e, \\ (S+\Delta S)(W+\Delta W)e &= \mu e, \end{aligned} \tag{17}$$

where ΔX, ΔZ, ΔS, and ΔW are diagonal matrices having elements Δx, Δz, Δs, and Δw, respectively. Simple algebra reduces (17) to the equivalent system

$$A\Delta x = b - Ax, \tag{18a}$$
$$\Delta x + \Delta s = u - x - s, \tag{18b}$$
$$A^T\Delta y - \Delta w + \Delta z = c - A^Ty + w - z, \tag{18c}$$
$$X\Delta z + Z\Delta x = \mu e - XZe - \Delta X\Delta Ze, \tag{18d}$$
$$S\Delta w + W\Delta s = \mu e - SWe - \Delta S\Delta We. \tag{18e}$$

The left-hand side of (18) is identical to (4), while the right-hand side has two distinct differences. The first deals with the equation defining the upper bounds. In the algorithm of §2 we always choose x^0 and s^0 so that $x^0 > 0$, $s^0 > 0$, and $x^0+s^0 = u$. Thus the right-hand side of equation (18b) is always zero. For reasons to be discussed later, this proves to be computationally unstable for the predictor–corrector method on problems with small upper bounds. Thus we assume only that $x^0 > 0$ and $s^0 > 0$ but not that the upper-bound constraint $x^0 + s^0 = u$ is satisfied initially. Rather, we allow the method to iterate to bound feasibility in precisely the same manner it iterates to primal and dual feasibility.

The major difference between (4) and (18) is the presence of the nonlinear terms $\Delta X\Delta Ze$ and $\Delta S\Delta We$ in the right-hand side of (18d) and (18e). Thus (18) implicitly defines the step Δx, Δy, Δz, Δs, Δw. To determine a step approximately satisfying (18), Mehrotra suggests first solving the defining equations for the primal–dual affine direction:

$$\begin{aligned} A\Delta\hat{x} &= b - Ax, \\ \Delta\hat{x} + \Delta\hat{s} &= u - x - s, \\ A^T\Delta\hat{y} - \Delta\hat{w} + \Delta\hat{z} &= c - A^Ty + w - z, \\ X\Delta\hat{z} + Z\Delta\hat{x} &= -XZe, \\ S\Delta\hat{w} + W\Delta\hat{s} &= -SWe. \end{aligned} \tag{19}$$

These directions are then used in two distinct ways: to approximate the nonlinear terms in the right-hand side of (18) and to dynamically estimate μ.

To estimate μ, Mehrotra performs the standard ratio test on both the primal and dual variables to determine the step that would actually be taken if the primal–dual affine direction defined by (19) were used. Thus, we define

$$\begin{aligned}\hat{\delta}_P &= \min\left\{\min_j\left\{\frac{x_j}{-\Delta\hat{x}_j}, \Delta\hat{x}_j < 0\right\}, \min_j\left\{\frac{s_j}{-\Delta\hat{s}_j}, \Delta\hat{s}_j < 0\right\}\right\},\\ \hat{\delta}_D &= \min\left\{\min_j\left\{\frac{z_j}{-\Delta\hat{z}_j}, \Delta\hat{z}_j < 0\right\}, \min_j\left\{\frac{w_j}{-\Delta\hat{w}_j}, \Delta\hat{w}_j < 0\right\}\right\},\end{aligned} \tag{20}$$

and let

$$\begin{aligned}\delta_P &= 0.99995\hat{\delta}_P,\\ \delta_D &= 0.99995\hat{\delta}_D.\end{aligned}$$

Then the new complementarity gap that would result from a step in the affine direction is

$$\hat{g} = (x + \delta_P\Delta\hat{x})^T(z + \delta_D\Delta\hat{z}) + (s + \delta_P\Delta\hat{s})^T(w + \delta_D\Delta\hat{w}). \tag{21}$$

Mehrotra's estimate in [9] for μ, generalized to include lower bounds, is then

$$\mu = \left(\frac{\hat{g}}{x^Tz + s^Tw}\right)^2\left(\frac{\hat{g}}{n}\right), \tag{22}$$

which chooses a small μ when good progress can be made in the affine direction and a large μ when the affine direction produces little improvement. This is appealing, since poor progress in the affine direction generally indicates the need for more centering and hence a larger value of μ. In [10] Mehrotra uses a cubic rather than a quadratic multiplier, but comparison of the iteration counts of §5 with those of [10] indicates that this appears to make little difference.

In the implementation tested and documented in §5, we have essentially adopted this same algorithm for choosing μ with one minor difference. We found that choosing μ by (22) can result in numerically unstable systems as the optimum is approached on poorly conditioned problems, such as the `pilot` models. Thus when the absolute complementarity $x^Tz + s^Tw \geq 1$, we define μ by (22), but when $x^Tz + s^Tw < 1$, we define

$$\mu = (x^Tz + s^Tw)/\phi(n), \tag{23}$$

where $\phi(n)$ is defined by (13). In practice this proved totally satisfactory and far more stable than always choosing μ by (22).

The actual new step $\Delta x, \Delta y, \Delta z, \Delta s, \Delta w$ is then chosen as the solution to

$$\begin{aligned}A\Delta x &= b - Ax,\\ \Delta x + \Delta s &= u - x - s,\\ A^T\Delta y - \Delta w + \Delta z &= c - A^Ty + w - z,\\ X\Delta z + Z\Delta x &= \mu e - XZe - \Delta\hat{X}\Delta\hat{Z}e,\\ S\Delta w + W\Delta s &= \mu e - SWe - \Delta\hat{S}\Delta\hat{W}e.\end{aligned} \tag{24}$$

Clearly, all that has changed from (4) is the right-hand side, so the matrix algebra remains the same as in the solution (7). Ratio tests identical to (20) are now done using $\Delta x, \Delta y, \Delta z, \Delta s$, and Δw to determine actual step sizes α_P and α_D, and the actual new point x^*, y^*, z^*, s^*, w^* is defined by (8).

After the coefficient matrix $A\Theta A^T$ has been factored, the additional work of the predictor–corrector method is in the extra backsolve to compute the affine direction and the extra ratio test used to compute μ. What is gained from this extra work is approximate second-order information concerning the trajectory from the current estimate to the optimal point as μ is varied continuously.

To see this, note that the full primal–dual affine correction terms, $\Delta\hat{X}\Delta\hat{Z}e$ and $\Delta\hat{S}\Delta\hat{W}e$, are added to the right-hand side of (24). The step lengths δ_P and δ_D defined by (20) are used to compute μ but not to modify the correction. Thus the correction added is one that would result from taking a full step of length 1 in the affine direction. Whenever a primal and dual full step of length 1 can be taken, primal feasibility and dual feasibility are achieved exactly within the numerical accuracy of the computations. Now the solution to (24) can be written as

$$\begin{aligned} \Delta y &= \Delta\hat{y} + c_y, \\ \Delta x &= \Delta\hat{x} + c_x, \\ \Delta z &= \Delta\hat{z} + c_z, \\ \Delta w &= \Delta\hat{w} + c_w, \\ \Delta s &= \Delta\hat{s} + c_s, \end{aligned} \tag{25}$$

where $\Delta\hat{x}, \Delta\hat{y}, \Delta\hat{z}, \Delta\hat{s}, \Delta\hat{w}$ are the solution to (19) and the correction terms c_x, c_y, c_z, c_s, and c_w satisfy

$$\begin{aligned} Ac_x &= 0, \\ c_x + c_s &= 0, \\ A^Tc_y + c_z - c_w &= 0, \\ Xc_z + Zc_x &= \mu e - \Delta\hat{X}\Delta\hat{Z}e, \\ Sc_w + Wc_s &= \mu e - \Delta\hat{S}\Delta\hat{W}e. \end{aligned} \tag{26}$$

Now, if the full step of 1 were achieved on this affine step, the new complementarity would be precisely

$$\begin{aligned} &(x+\Delta\hat{x})^T(z+\Delta\hat{z}) + (s+\Delta\hat{s})^T(w+\Delta\hat{w}) \\ &\quad = x^Tz + \Delta\hat{x}^Tz + \Delta\hat{z}^Tx + \Delta\hat{x}^T\Delta\hat{z} + s^Tw + \Delta\hat{s}^Tw + \Delta\hat{w}^Ts + \Delta\hat{s}^T\Delta\hat{w} \\ &\quad = \Delta\hat{x}^T\Delta\hat{z} + \Delta\hat{s}^T\Delta\hat{w}, \end{aligned}$$

as by the definition of $\Delta\hat{x}, \Delta\hat{s}, \Delta\hat{z}, \Delta\hat{w}$ in (19),

$$x^Tz + \Delta\hat{x}^Tz + \Delta\hat{z}^Tx = s^Tw + \Delta\hat{z}^Tw + \Delta\hat{w}^Ts = 0.$$

Thus (26) could describe a centered Newton step from the point $x+\Delta\hat{x}$, $y+\Delta\hat{y}$, $z+\Delta\hat{z}$, $s+\Delta\hat{s}$, $w+\Delta\hat{w}$ except that the corrections $\Delta\hat{x}$, $\Delta\hat{y}$, $\Delta\hat{z}$, $\Delta\hat{s}$, $\Delta\hat{w}$ have not been added to the diagonal matrices on the left-hand side of (26). Thus the correction terms are a Newton step from the point achieved by a full affine step, but using the

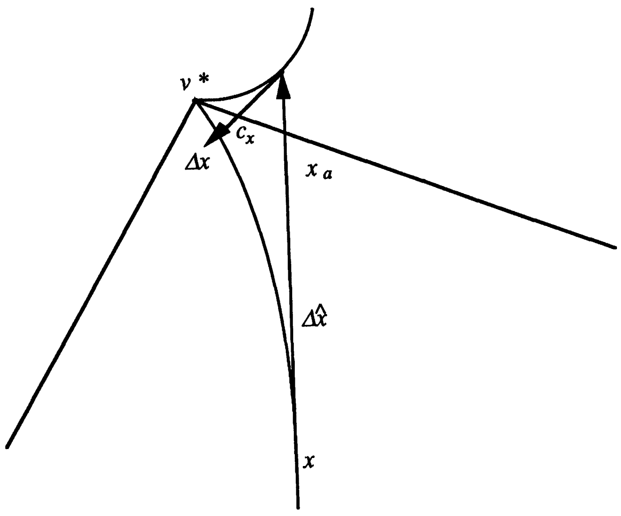

FIG. 1. *Diagram of direction vectors.*

second-derivative matrix at the current point x, y, z, w, s as an approximation to the Hessian at the point corresponding to a full affine step.

The effect of this is demonstrated in Fig. 1, shown in terms of the primal variables x. Here the vertex v^* is the desired optimum. The affine direction $\Delta\hat{x}$ from the current estimate x is a vector tangent to the continuous trajectory as $\mu \to 0$, as is shown in the figure. The point x_a is the new point predicted by the affine variant with a step of length 1, which by the previous analysis must lie outside the feasible region. The correction term c_x that is added at x_a is not tangent to the trajectory through x_a for two reasons. First, a centering term corresponding to $\mu > 0$ is added, rotating c_x toward the center of the polytope. Second, the second-derivative matrix used to calculate c_x is computed at x and not at x_a. Hence, even without a centering term, the vector c_x would not be exactly tangent at x_a. However, the curvature of the trajectories defined through each point by continuously varying μ is clearly estimated by the method. The results of §5 show this to be extremely effective computationally.

Higher-order methods for interior-point algorithms have been proposed by Bayer and Lagarias [2] and implemented by Adler, Karmarkar, Resende, and Veiga [1] and Domich, Boggs, Rogers, and Witzgall [5]. Each of these implementations uses power series estimates to the trajectory at x and shows that higher-order methods reduce the iteration count, with occasional significant reductions. However, Adler et al. report that the increased costs of computing the higher-order terms generally eliminate any effective advantage gained from the lower iteration counts, whereas Domich et al. do not provide comparative timings. As previously noted, Mehrotra [10] uses a different power series derivation for the predictor–corrector method.

The results of §5 show that the predictor–corrector method almost always reduces the iteration count and usually reduces computation time. Furthermore, as problem size and complexity increase, the improvements in both iteration count and execution time become greater. Thus the predictor–corrector method is a higher-order method that is generally very computationally efficient.

Another small variant in the algorithm involves the choice of step size. Zhang, Tapia, and Dennis [11] have recently shown that for nondegenerate problems the primal–dual method will have an asymptotic quadratic rate of convergence if a step of length 1 is taken as the optimum is approached. Thus, if $0.99995\hat{\delta}_P > 1$ or $0.99995\hat{\delta}_D > 1$, we restrict the step length to 1. Further, if δ_P and δ_D are both equal to 1, a correction is not added on this iteration, but the affine direction is accepted with a step length of 1, thus attaining primal and dual feasibility in one step in the affine direction.

As a final note on this section, we have stated that this implementation of the predictor–corrector algorithm allows bound infeasibility. This proved to be necessary for the predictor–corrector method as it is more sensitive to the starting point than the pure primal–dual method. Although it is theoretically always easy to maintain bounds exactly, the predictor–corrector method performs best with relatively large initial estimates to the primal variables x. Some models of the NETLIB test set, notably the `pilot` models, have upper bounds of 10^{-5} for some variables, and hence maintaining bound feasibility requires very small initial estimates to x. For these estimates, the method has proved quite unstable and often fails to converge because of numerical problems. Allowing the initial estimates for x and s to violate the bounds resolves these problems and is quite efficient in practice.

4. Yet again, dense columns.

In Choi, Monma, and Shanno [4] the use of Schur complements to eliminate dense columns from A to assure a sufficiently sparse factorization of $A\Theta A^T$ is discussed. In [8] the authors discuss the effects of this in greater detail and document numerical stability problems with the algorithm. It is the purpose of this section to identify the instability, suggest a possible remedy, and advise that the remedy be used with extreme caution.

The nature of the problem can be seen easily from the following simple linear programming problem:

$$(27)\qquad \begin{array}{ll} \min & -x_1 \\ \text{subject to} & x_1 + x_2 = 2, \\ & x_1 + x_3 = 1, \\ & x_1,\ x_2,\ x_3 \ge 0. \end{array}$$

This has the solution $x_1 = 1$ and $x_2 = 1$. If we consider the first column to be dense, the resulting sparse matrix factored during the Schur complement procedure is

$$(28)\qquad \begin{aligned} A_s\Theta_s A_s^T &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_2/z_2 & 0 \\ 0 & x_3/z_3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} x_2/z_2 & 0 \\ 0 & x_3/z_3 \end{bmatrix}. \end{aligned}$$

As the solution is approached, $x_3 \to 0$ and this matrix becomes rank deficient. This introduces sufficient instability into the Schur complement procedures to make it impossible at times to achieve the desired accuracy. Further, this is not a contrived example, for often the removal of dense columns leaves all remaining elements of one or more rows identically zero. Thus an artificial variable must be added to each of these rows to assure that $A_s\Theta_s A_s^T$ is not rank deficient. Since these artificial variables must be driven to zero, rank deficiency always results as the optimum is approached, and the desired accuracy is almost never attained.

To attempt to alleviate this problem when Schur complements are being used, the algorithm monitors the spread between the largest and the smallest diagonal elements of the factorization

$$(29)\qquad L_s D_s L_s^T = A_s\Theta_s A_s^T.$$

When this spread becomes large, all smaller diagonal elements are set to 1 and the remaining elements of these columns are set to 0. These Cholesky factors are used as preconditioners for a preconditioned conjugate gradient method [1] with the dense column added to the matrix A. This eliminates rank deficiency and removes much of the instability. Although, in general, Schur complements are more efficient than preconditioned conjugate gradients, in this case preconditioned conjugate gradients allowed for accurate solutions while Schur complements did not.

For example, the linear program (27) is easily solved in 10 iterations by the primal–dual method of §2 without dense columns removed and by the same method combined with the preconditioned conjugate gradient method with the single dense column removed. However, failure occurs because of numerical instability when (27) is solved by using pure Schur complements. Interestingly, (27) is easily solved with the single dense column removed by using the predictor–corrector algorithm of §3 and Schur complements, indicating that the new method may be much more stable than pure primal-dual algorithms. On problem **seba**, when columns of length 50 or more are removed and the minimum local fill-in ordering for the matrix $A_s\Theta_sA_s^T$ is used, the predictor–corrector method can achieve only seven digits of accuracy before failing at the 19th iteration. However, the pure primal–dual method easily achieves eight digits of accuracy in 28 iterations.

It is indeed instructive to examine how the predictor–corrector method performs on **seba** under different options. Five runs were executed and are identified in Table 1 as **minfil** for Schur complements with the minimum local fill-in ordering, **mmind** for Schur complements with the multiple minimum degree ordering, **nodense** for a pure Cholesky factorization with no dense columns removed, **switch** for Schur complements with the minimum local fill-in ordering switching to preconditioned conjugate gradients when the spread of the diagonal elements was greater than 10^{14}, and **conjgrad** for removing dense columns by the preconditioned conjugate gradient method at each iteration. The times reported are simply solution times.

TABLE 1
Timings on SEBA.

Method	CPU seconds	Accuracy
minfil	14.60	10^{-7}
mmind	14.70	10^{-8}
nodense	388.70	10^{-8}
switch	32.44	10^{-8}
conjgrad	86.85	10^{-8}

Clearly, removal of dense columns on **seba** is valuable. Each of the four methods that removed dense columns took approximately 0.15 seconds to do the ordering, independent of whether it was minimum local fill-in or multiple minimum degree. However, minimum local fill-in without removal of the dense columns required 91.05 seconds to do the ordering. The fact that the same predictor–corrector algorithm achieved different accuracies with different orderings indicates the sensitivity of the algorithm to ill-conditioned matrices. Finally, the results clearly show that even for problems with a few dense columns (**seba** has 14 dense columns), preconditioned conjugate gradient methods are much slower than Cholesky factorizations and should be used sparingly. In view of this, the **switch** option appears sensible when accuracy is being lost using Schur complements, but further experimentation is needed to determine the optimal time to switch from Schur complements to preconditioned conjugate gradients.

For the NETLIB test set we needed to remove dense columns for only four problems: **israel, seba, fit1p**, and **fit2p**. Problems **israel** and **seba** can both be solved without removing dense columns. However, **fit1p** and especially **fit2p** must have dense columns removed in order to be solved, due to their sufficient size and density. Problems **fit1p** and **fit2p** demonstrate that the capability of removing dense columns is not just more efficient, but at times is actually required. In view of this need, the computational results of the next section give the results for **israel, seba, fit1p**, and **fit2p** with pure Schur complements. However, only seven digits of accuracy were achieved for **seba**. Eight were achieved by using the switch option.

5. Computational results. OB1, a modularized FORTRAN-77 code that was developed to implement the primal–dual barrier method, is documented by Lustig, Marsten, and Shanno [8]. This code for the pure barrier method was modified to alter the initial μ, as described in §2, but otherwise remains as in [8] with the single further exception that at each step we move to a point corresponding to 0.99995 of the distance to the boundary, rather than 0.995.

The predictor–corrector method described in §3 was implemented within the same software, for which the method is selected by a user-specified parameter. As noted in §3, the predictor–corrector method is quite sensitive to the initial guess to the optimal solution. Following Mehrotra [9], an initial estimate to the primal variables x was chosen to be

$$\tilde{x} = A^T(AA^T)^{-1}b. \tag{30}$$

We then define

$$\xi_1 = \max(-\min_{1\le j\le n} \tilde{x}_j, 100, \|b\|/100) \quad \text{and} \quad \xi_2 = 1 + \|c\|, \tag{31}$$

where $\|\cdot\|$ is the l_1 norm.

Then, for each $j = 1, \cdots, n$,

$$x_j^0 = \max(\tilde{x}_j, \xi_1) \quad \text{and} \quad s_j^0 = \max(\xi_1, u_j - x_j^0). \tag{32}$$

We set $y^0 = 0$ and the pair z^0, w^0 to satisfy

$$\begin{aligned} z_j &= c_j + \xi_2, & w_j &= \xi_2, & &\text{if } c_j > \xi_2,\\ z_j &= -c_j, & w_j &= -2c_j, & &\text{if } c_j < -\xi_2,\\ z_j &= c_j + \xi_2, & w_j &= \xi_2, & &\text{if } 0 \le c_j < \xi_2,\\ z_j &= \xi_2, & w_j &= -c_j + \xi_2, & &\text{if } -\xi_2 \le c_j \le 0. \end{aligned} \tag{33}$$

In addition, if we have upper bounds satisfying $u_j < 0.001$ for any j, the components of x^0 are all set to 100. We believe that further experimentation along the lines of Mehrotra [9] would yield a more stable algorithm to determine the starting point.

Besides adding the predictor–corrector method to OB1, we implemented the option of ordering the matrix $A\Theta A^T$ by either multiple minimum degree or minimum local fill-in. These are discussed in detail by de Carvalho [3]. His tests show that overall performance on large problems indicates a definite preference for minimum local fill-in. We tested both orderings as well, and the results of our experiments on a large test set totally reinforce his conclusion, although no strong inference can be drawn from testing on small problems.

The numerical experiments conducted here were done on 86 problems of the expanded NETLIB [6] test set. We did not solve problems `stocfor3` or `truss` because they require substantial time to generate the MPS file from FORTRAN code. The problem `stocfor3` is large enough to be difficult to solve in our computing environment. All experiments were conducted on a Silicon Graphics 4D/70 workstation with 16 megabytes of memory and one processor. The code was compiled with the MIPS `f77` compiler using options `-O2` and `-Olimit 800`.

The main results compare the relative efficiency of the primal–dual and predictor–corrector algorithms. The results are documented in Tables 2 and 3. Solution times do not include preprocessing time or the time to compute the minimum local fill-in ordering. Each method solved 85 of the 86 test problems with the default settings. As previously noted for `seba`, once dense columns of more than 50 elements were eliminated, the predictor–corrector method attained only seven digits of accuracy. The primal–dual method achieved only seven digits of accuracy on `fit2p` with the standard default and dense columns again removed, but easily achieved eight digits of accuracy with the value of ξ in (14) raised from the default of 0.1 to 1. Both methods in this comparison used the minimum local fill-in ordering.

The predictor–corrector method outperformed the primal–dual method on iteration count for 85 of the 86 problems. More importantly, it outperformed the primal–dual in execution time for 71 of the 86 problems. As expected, the percentage decrease in iterations is always greater than the percentage decrease in run time because of the extra back solve and ratio test required. However, the percentage decrease in run time is still quite impressive, especially on large, difficult problems. Yet the run time percentages can still be improved. Both methods test to assure that $A\Delta x$ is sufficiently close to $b - Ax$, which entails calculating $A\Delta x$ at each iteration. When this test fails, iterative refinement is invoked. Here, we use the specific form of iterative refinement devised for least-squares problems [7]. We have found this to be an important safety feature for large, difficult problems, but on most problems of the NETLIB test set it represents unnecessary overhead and could be removed. It should be noted that on many problems the number of nonzeros in A and L are of the same order of magnitude; hence, this test can be as expensive as a forward and backward pass through L. OB1 has been designed to solve difficult problems with little intervention from the user. Therefore, we prefer to present results on an algorithm designed for maximum stability. For the user who never requires iterative refinement, the computational superiority of the predictor–corrector method may be even more pronounced. The one unmistakable conclusion, with or without iterative refinement, is that the predictor–corrector algorithm is substantially superior to the pure primal–dual algorithm and that this superiority grows with problem size and complexity.

Iteration counts of the algorithms documented here are comparable to Mehrotra's algorithm [10], even though small differences occur on individual problems. Thus Mehrotra's somewhat greater computational advantage in terms of computation time over the primal–dual algorithms of OB1 is due almost entirely to the extra overhead imposed by the check for iterative refinement, inclusion of bounds, and other safeguards we found necessary for larger, more complex problems. The one check that we have not included is one to assure that his potential function is reduced sufficiently at

TABLE 2
Computational results for OB1 (A–N).

Problem	No. of	No. of	No. of	Primal/Dual Meth.		Pred./Corr. Meth.		Dynamic μ	
Name	Rows	Cols.	Nonzeros	Its.	Sol. Time	Its.	Sol. Time	Its.	Sol. Time
25fv47	821	1571	10400	47	148.48	25	92.41	46	167.55
80bau3b	2262	9799	21002	70	363.72	38	253.98	64	425.97
adlittle	56	97	383	16	0.70	12	0.73	17	1.03
afiro	27	32	83	13	0.26	9	0.28	11	0.32
agg	488	163	2410	28	19.24	24	19.52	29	23.65
agg2	516	302	4284	27	46.34	18	36.22	27	53.62
agg3	516	302	4300	26	44.82	17	34.50	25	49.99
bandm	305	472	2494	28	7.53	17	5.87	28	9.41
beaconfd	173	262	3375	18	3.39	10	2.45	14	3.31
blend	74	83	491	15	1.07	14	1.26	18	1.61
bnl1	643	1175	5121	56	54.05	27	32.77	49	58.49
bnl2	2324	3489	13999	59	1037.78	33	632.11	61	1159.18
boeing1	351	384	3485	38	15.72	24	12.84	40	21.08
boeing2	166	143	1196	27	3.39	14	2.41	23	3.75
bore3d	233	315	1429	39	4.69	18	2.87	27	4.28
brandy	220	249	2148	28	6.41	19	5.35	24	6.77
capri	271	353	1767	44	16.07	18	8.43	29	13.48
cycle	1903	2857	20720	51	196.09	30	137.74	48	218.43
czprob	929	3523	10669	59	48.58	35	40.09	53	60.14
d2q06c	2171	5167	32417	53	804.24	31	525.11	51	856.96
degen2	444	534	3978	24	38.44	14	26.31	23	42.02
degen3	1503	1818	24646	31	766.81	20	551.83	40	1082.83
e226	223	282	2578	27	7.01	22	7.19	34	11.07
etamacro	400	688	2409	45	37.61	29	28.80	50	49.30
fffff800	524	854	6227	60	79.15	28	42.99	44	67.21
finnis	497	614	2310	41	13.96	26	11.78	42	18.88
fit1d	24	1026	13404	22	16.19	18	17.16	28	26.39
fit1p	627	1677	9868	22	34.01	16	29.91	23	42.89
fit2d	25	10500	129018	47	320.21	24	219.82	40	361.11
fit2p	3000	13525	50284	32	302.19	18	206.91	31	366.76
forplan	161	421	4563	40	16.71	21	10.71	30	15.20
ganges	1309	1681	6912	33	43.49	16	27.10	30	50.13
gfrdpnc	616	1092	2377	27	8.29	18	8.07	27	11.97
greenbea	2392	5405	30877	62	290.71	41	218.21	61	340.34
greenbeb	2392	5405	30877	70	287.16	33	167.31	60	301.41
grow15	300	645	5620	26	14.33	16	11.61	20	14.42
grow22	440	946	8252	29	23.73	16	17.44	21	22.40
grow7	140	301	2612	22	5.49	14	4.54	19	6.12
israel	174	142	2269	30	13.61	23	12.19	33	17.44
kb2	43	41	286	23	0.91	15	0.81	24	1.24
lotfi	153	308	1078	34	4.46	16	2.94	24	4.30
nesm	662	2923	13288	66	148.57	30	86.46	57	161.93

each step. Since our only difficulties with convergence occur through failures of linear algebra, we have not found this test to be necessary. However, this test may be required later if we have unexplained computational difficulties. Also, work is underway to create a stripped-down version of OB1 that allows the user to choose speed rather than safety if experience indicates this will be satisfactory for a given problem. Thus our results and Mehrotra's are totally compatible for an efficient algorithm restricted

TABLE 3
Computational results for OB1 (P–W).

Problem	No. of	No. of	No. of	Primal/Dual Meth.		Pred./Corr. Meth.		Dynamic μ	
Name	Rows	Cols.	Nonzeros	Its.	Sol. Time	Its.	Sol. Time	Its.	Sol. Time
perold	625	1376	6018	62	178.54	33	104.66	55	173.56
pilot4	410	1000	5141	56	71.72	36	54.63	50	78.63
pilot.ja	940	1988	14698	66	537.75	46	408.00	56	500.32
pilotnov	975	2172	13057	36	369.98	20	230.03	29	330.19
pilot	1441	3652	43167	67	3813.52	29	1784.35	63	3798.77
pilot.we	722	2789	9126	57	94.67	46	100.60	56	120.59
recipe	91	180	663	17	1.17	10	0.96	13	1.22
sc105	105	103	280	16	0.78	10	0.66	14	0.89
sc205	205	203	551	21	1.83	11	1.31	15	1.83
sc50a	50	48	130	14	0.42	10	0.37	13	0.47
sc50b	50	48	118	12	0.36	8	0.30	11	0.43
scagr25	471	500	1554	24	4.86	16	4.51	24	6.65
scagr7	129	140	420	20	1.16	12	0.87	18	1.40
scfxm1	330	457	2589	29	9.11	17	7.03	26	10.48
scfxm2	660	914	5183	36	23.30	19	16.07	32	26.73
scfxm3	990	1371	7777	37	36.27	20	25.70	33	41.74
scorpion	388	358	1426	21	3.25	14	2.94	21	4.32
scrs8	490	1169	3182	36	15.73	27	15.84	48	27.82
scsd1	77	760	2388	12	2.52	11	3.01	13	3.63
scsd6	147	1350	4316	15	5.67	12	6.10	17	8.46
scsd8	397	2750	8584	14	11.18	10	10.98	15	16.08
sctap1	300	480	1692	21	4.28	15	4.12	25	6.77
sctap2	1090	1880	6714	23	23.85	20	26.73	27	36.41
sctap3	1480	2480	8874	24	34.13	17	32.24	30	55.52
seba	515	1028	4352	28	15.38	19	36.88	29	21.51
share1b	117	225	1151	36	3.55	20	2.68	35	4.71
share2b	96	79	694	18	1.37	12	1.21	18	1.72
shell	536	1775	3556	31	13.67	21	12.96	32	19.58
ship04l	402	2118	6332	25	13.20	15	11.02	23	16.57
ship04s	402	1458	4352	24	8.62	15	7.47	22	10.72
ship08l	778	4283	12802	26	23.92	16	20.65	26	32.62
ship08s	778	2387	7114	25	12.02	14	9.50	24	15.82
ship12l	1151	5427	16170	29	35.35	18	30.58	28	46.53
ship12s	1151	2763	8178	30	17.52	18	14.71	28	22.34
sierra	1227	2036	7302	31	40.14	18	31.41	30	51.25
stair	356	467	3856	25	25.87	16	19.75	23	27.92
standata	359	1075	3031	18	6.13	15	7.04	24	11.05
standmps	467	1075	3679	28	12.19	24	13.97	34	19.60
stocfor1	117	111	447	18	1.09	19	1.46	21	1.63
stocfor2	2157	2031	8343	34	51.78	22	44.14	36	71.28
tuff	333	587	4520	45	30.13	19	15.69	31	25.05
vtp.base	198	203	908	24	1.17	13	0.91	15	1.09
wood1p	244	2594	70215	22	108.18	14	80.66	20	119.66
woodw	1098	8405	37474	35	156.05	20	108.83	34	181.96

to his simple test set.

The predictor–corrector algorithm introduces two new concepts, namely, the correction term and the dynamic choice of μ by (22). It is interesting to study the effects of each concept by eliminating the corrective term and comparing the primal–dual of §2 with a pure primal–dual using the μ given by (22). The results in Tables 2 and

3 (columns labeled "Dynamic μ") clearly demonstrate that the improvement in the algorithm is attributed largely to the correction term rather than the choice of μ. Nevertheless, we found that the predictor–corrector worked best with the algorithm for μ documented in §3, rather than the simpler μ of §2. Therefore, whereas the dynamic μ given by (22) has a largely negative effect on the primal–dual algorithm in terms of execution time, it significantly helps the predictor–corrector algorithm. For the version labeled "Dynamic μ," numerical difficulties prevented convergence to eight digits of accuracy on problems **greenbea** and **pilot.we**.

On the Silicon Graphics 4D/70, our tests strongly substantiated de Carvalho's conclusion [3] that minimum local fill-in generally outperforms multiple minimum degree. However, this result is very architecture dependent and is certainly invalid for vector architectures such as the CRAY Y-MP. Careful testing must be done on any specific architecture to determine the correct default algorithm.

In conclusion, the predictor–corrector algorithm of §3 implemented with the minimum local fill-in ordering is a substantial improvement over the primal–dual algorithm of [8]. Compared with other interior-point methods, the predictor–corrector algorithm is to date the most computationally efficient method for solving large-scale linear programs.

REFERENCES

[1] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga, *An implementation of Karmarkar's algorithm for linear programming*, Math. Programming, 44 (1989), pp. 297–325.

[2] D. Bayer and J. Lagarias, *The nonlinear geometry of linear programming* I. *Affine and projective scaling trajectories*, Trans. Amer. Math. Soc., 314 (1989), pp. 499–526.

[3] M. de Carvalho, *On the minimization of work needed to factor a symmetric positive definite matrix*, Tech. Report ORC 87-14, Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA, 1987.

[4] I. C. Choi, C. L. Monma, and D. F. Shanno, *Further development of a primal–dual interior point method*, ORSA J. Comput., 2 (1990), pp. 304–311.

[5] P. D. Domich, P. T. Boggs, J. E. Rogers, and C. Witzgall, *Optimizing over* 3-*dimensional subspaces in an interior point method for linear programming*, Linear Algebra Appl., 152 (1989), pp. 315–342.

[6] D. M. Gay, *Electronic mail distribution of linear programming test problems*, Mathematical Programming Society COAL Newsletter, No. 13 (December 1985), pp. 10–12.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983, pp. 182–183.

[8] I. J. Lustig, R. E. Marsten, and D. F. Shanno, *Computational experience with a primal-dual interior point method for linear programming*, Linear Algebra Appl., 152 (1989), pp. 191–222.

[9] S. Mehrotra, *On finding a vertex solution using interior point methods*, Linear Algebra Appl., 152 (1990), pp. 233–253.

[10] ———, *On the implementation of a (primal–dual) interior point method*, Tech. Report 90-03, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL 1990.

[11] Y. Zhang, R. A. Tapia, and J. E. Dennis, *On the superlinear and quadratic convergence of primal-dual interior point linear programming algorithms*, Tech. Report 90-6, Department of Mathematical Sciences, Rice University, Houston, TX, 1990.