Operations Research Letters 11 (1992) 195–198
North-Holland

May 1992

Prior reduced fill-in in solving equations in interior point algorithms

John R. Birge
University of Michigan, 1205 Beal Avenue, Ann Arbor, MI 48109-2177, USA

Robert M. Freund
Sloan School of Management, M.I.T., E53-361, Cambridge, MA 02139, USA

Robert Vanderbei
Dept. of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544, USA

Received August 1990
Revised August 1991

The efficiency of interior-point algorithms for linear programming is related to the effort required to factorize the matrix used to solve for the search direction at each iteration. When the linear program is in symmetric form (i.e., the constraints are $Ax \leq b$, $x \geq 0$), then there are two mathematically equivalent forms of the search direction, involving different matrices. One form necessitates factoring a matrix whose sparsity pattern has the same form as that of (AA^T). The other form necessitates factoring a matrix whose sparsity pattern has the same form as that of (A^TA). Depending on the structure of the matrix A, one of these two forms may produce significantly less fill-in than the other. Furthermore, by analyzing the fill-in of both forms prior to starting the iterative phase of the algorithm, the form with the least fill-in can be computed and used throughout the algorithm. Finally, this methodology can be applied to linear programs that are not in symmetric form, that contain both equality and inequality constraints.

interior-point algorithm; linear program; factorization; fill-in

1. Introduction and notation

The efficiency of interior-point algorithms for linear programming is related to the effort required to factorize the matrix used to solve for the search direction at each iteration. When the linear program is in symmetric form (i.e., the constraints are $Ax \leq b$, $x \geq 0$), then there are two mathematically equivalent forms of the search direction, involving different matrices. One form necessitates factoring a matrix whose sparsity pattern has the same form as that of (AA^T). The other form necessitates factoring a matrix whose sparsity pattern has the same form as that of (A^TA). Depending on the structure of the matrix A, one of these two forms may produce significantly less fill-in than the other. Furthermore, by analyzing the fill-in of both forms prior to starting the iterative phase of the algorithm, the form with the least fill-in can be computed and used throughout the algorithm. Finally, this methodology can be applied to linear programs that are not in symmetric form, that contain both equality and inequality constraints.

The notation used is as follows. The vector of ones is represented by $\boldsymbol{e}$, $\boldsymbol{e} = (1, 1, \ldots, 1)^T$. If $\boldsymbol{x}$ and $\boldsymbol{s}$ are vectors, then $\boldsymbol{X}$ and $\boldsymbol{S}$ are the diagonal matrices whose diagonal entries correspond to $\boldsymbol{x}$ and $\boldsymbol{s}$.

Correspondence to: Robert Vanderbei, Dept. of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544, USA.

2. Observation using the Sherman–Morrison–Woodbury formula for linear programs in symmetric form

Consider a linear program in symmetric form:

SFP

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^{\mathrm{T}}x \\ \text{s.t.} \quad & Ax \le b, \\ & x \ge 0, \end{aligned}$$

which can alternatively be written as

SFP′:

$$\begin{aligned} \underset{x,s}{\text{maximize}} \quad & c^{\mathrm{T}}x + 0^{\mathrm{T}}s \\ \text{s.t} \quad & Ax + Is = b, \\ & (x, s) \ge 0. \end{aligned}$$

Suppose (x, s) is a current interior feasible solution, i.e., $Ax + s = b$, $x > 0$, $s > 0$. Virtually all known interior-point algorithms compute the next direction $d = (d_x, d_s)$ as a linear combination of the affine-scaling direction (see Vanderbei et al. [8], Barnes [2], Dikin [4]) and the Newton centering direction, see Gonzaga [6], also Den Hertog and Roos [3], also Karmarkar [7]. For problem SFP, the affine-scaling direction is the solution (up to scalar multiple) to the problem

$$\begin{aligned} \underset{d_x,d_s}{\text{maximize}} \quad & c^{\mathrm{T}}d_x \\ \text{s.t} \quad & Ad_x + d_s = 0, \\ & d_x^{\mathrm{T}}X^{-2}d_x + d_s^{\mathrm{T}}S^{-2}d_s \le 1. \end{aligned}$$

Eliminating d_s in the above and applying the Karush–Kuhn–Tucker conditions yields the affine-scaling direction (up to a scalar multiple)

$$d_x = (A^{\mathrm{T}}S^{-2}A + X^{-2})^{-1}X^{-1}(Xc) \quad \text{(affine scaling)}. \tag{1a}$$

The Newton centering direction is derived by computing the Newton step from (x, s) in the centering problem

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} \ln x_j + \sum_{i=1}^{m} \ln s_i \\ \text{s.t.} \quad & Ax + s = b, \\ & (x, s) > 0. \end{aligned}$$

Again eliminating the s variables and computing the Newton direction in the x variables yields

$$d_x = (A^{\mathrm{T}}S^{-2}A + X^{-2})^{-1}X^{-1}(e - XA^{\mathrm{T}}S^{-1}e) \quad \text{(Newton centering)}. \tag{1b}$$

However, an alternate form of (1a) and (1b) can be derived using the following result.

Proposition.

$$(A^{\mathrm{T}}S^{-2}A + X^{-2})^{-1}X^{-1} = X\left[I - XA^{\mathrm{T}}(AX^2A^{\mathrm{T}} + S^2)^{-1}AX\right]. \tag{2}$$

Proof. Direct multiplication reveals that

$$X(A^{\mathrm{T}}S^{-2}A + X^{-2})\, X\left[I - XA^{\mathrm{T}}(AX^2A^{\mathrm{T}} + S^2)^{-1}AX\right] = I,$$

and therefore

$$\left[X(A^{\mathrm{T}}S^{-2}A + X^{-2})\right]^{-1} = X\left[I - XA^{\mathrm{T}}(AX^2A^{\mathrm{T}} + S^2)^{-1}AX\right]$$

which is equivalent to (2). □

Equation (2) can also be viewed as a modified instance of the Sherman–Morrison–Woodbury formula.

Using (2), an alternate form of (1a) and (1b) is

$$d_x = X\left[I - XA^{\mathrm{T}}(AX^2A^{\mathrm{T}} + S^2)^{-1}AX\right](Xc) \quad \text{(affine scaling)} \tag{3a}$$

and

$$d_x = X\left[I - XA^{\mathrm{T}}(AX^2A^{\mathrm{T}} + S^2)^{-1}AX\right] \times (e - XA^{\mathrm{T}}S^{-1}e) \quad \text{(Newton centering)}. \tag{3b}$$

3. Comparisons

Note in (1) that the major computational burden in computing d_x is the solution of equations involving the matrix $(A^{\mathrm{T}}S^{-2}A + X^{-2})$, whose sparsity depends on the sparsity of the matrix $A^{\mathrm{T}}A$. Also, if A is $m \times n$ (and for problem SFP we could have $m \ge n$ or $m \le n$), then the equation system to be solved is $n \times n$. In contrast, the

major computational burden in computing d_x using (3) lies in solving equations involving the matrix $(AX^2A^T + S^2)$, whose sparsity depends on the sparsity of the matrix AA^T, and the system of equations to be solved is $m \times m$.

In deciding whether to compute d_x by (1) or by (3), one consideration is the size of the respective systems, either $m \times m$ or $n \times n$. Another consideration is the fill-in in the factors of the respective systems, if they are to be solved by working with the Cholesky factorization. Prior to starting the iterative phase of the algorithm for SFP, one can analyze the fill-in in both (A^TA) and (AA^T) to reveal the extent of the fill-in in the factors of each system. If the fill-in in one of the systems is significantly less than in the other system, then the system with less fill-in should be chosen.

In particular, if A has any number of dense rows, A^TA will be dense and so the computation of d_x from (3) would be preferred. Similarly, if A has a number of dense columns, AA^T will be dense and so the computation of d_x from (1) would be preferred.

4. Extensions to problems not in symmetric form

Many linear programming problems are cast in the more general form:

RP

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & c^Tx \\ \text{s.t} \quad & Ax + s = b, \\ & Px = q, \\ & (x, s) \geq 0, \end{aligned}$$

where s are slack variables on the $Ax \leq b$ constraints, and there are a relatively small number of other constraints $Px = q$. The variables s then can be viewed as a partial basis for the system

$$\begin{bmatrix} A & I \\ P & 0 \end{bmatrix} \begin{pmatrix} x \\ s \end{pmatrix} = \begin{pmatrix} b \\ q \end{pmatrix}.$$

We now illustrate how the methodology presented in the last two sections can be extended to the case of problem RP. For simplicity, we will work with the affine scaling direction. The extension to the Newton-center direction follows similarly. The affine-scaling direction for RP is the solution $d = (d_x, d_s)$ to the program

$$\begin{aligned} \underset{d_x, d_s}{\text{maximize}} \quad & c^Td_x \\ \text{s.t.} \quad & Ad_x + d_s = 0, \\ & Pd_x = 0, \\ & d_x^TX^{-2}d_x + d_s^TS^{-2}d_s \leq 1. \end{aligned}$$

Eliminating d_s in the above and letting

$$Q = A^TS^{-2}A + X^{-2}, \tag{4}$$

the affine scaling direction in the x-coordinates (up to a scalar multiple) is

$$d_x = Q^{-1}c - Q^{-1}P^T(PQ^{-1}P^T)^{-1}PQ^{-1}c. \tag{5}$$

Note that the major computational burden in computing d_x in (5) lies in solving systems involving the matrices Q and $(PQ^{-1}P^T)$. If the number of rows of P is relatively small, then solving equations involving the matrix $(PQ^{-1}P^T)$ should not be significant, in comparison to the effort involved in solving equations involving the matrix Q.

One method for treating Q^{-1} is to form Q directly as in (4) and then to factorize Q accordingly. Here we see from (4) that the sparsity pattern of Q is identical to the sparsity pattern of the matrix A^TA. An alternative strategy for solving systems involving Q is to observe through (2) that

$$Q^{-1} = X\left[I - XA^T(AX^2A^T + S^2)^{-1}AX\right]X. \tag{6}$$

Solving systems involving Q using (6) requires factorizing $(AX^2A^T + S^2)$, whose sparsity pattern depends on the sparsity pattern of AA^T. Prior to starting the iterative phase of the algorithm for RP, one can analyze the fill-in in the matrices A^TA and AA^T to reveal the extent of the fill-in in the factors before choosing to use (4) or (6) in solving for d_x in (5). The other comments in Section 3 regarding this strategy remain valid for this case as well.

The strategy of choosing to solve for the interior point direction via the better of the two methods discussed here was first reported in Arantes and Birge [1], where computation time was reduced by at least 75%. Herein this strategy is tested on a subset of the netlib suite of linear programming problems (Gay [5]). Table 1 below shows the arithmetic operations needed to solve

Table 1
Arithemetic operations in solving for interior point direction (in millions)

netlib problem	Solve via $(AX^2A^T + S^2)$	Solve via $(A^TS^{-2}A + X^{-2})$	Speed-up factor in better method
degen3	15.	313.	21
bnl2	14.	384.	27
25fv47	2.6	35.	13
cycle	5.7	22	4
d2q06c	33	481	15
fit1p	86	0.59	146
fit2p	9079	4.3	2111
agg	0.68	0.23	3

for the interior point direction on eight netlib problems from the suite. These problems were chosen somewhat at random from among the larger netlib problems to illustrate the potential of choosing the better of the two methods. The tests were performed on an IBM RS-6000. The ordering heuristic used was minimum degree with mass elimination of all indistinguishable nodes. As the table shows, the advantage of using the better of the two strategies varies, obviously due to the structure of the individual problems. Nevertheless, for some problems the speed-up is quite large, on the order of 100 (146 for fit1p) or higher (2111 for fit2p).

Finally, we note that a broad class of solution strategies for solving interior point equations that includes the two mentioned here as special cases appears in Vanderbei [9].

Acknowledgment

The work of John Birge was supported in part under Grant EECS-885101 from the National Science Foundation.

References

[1] J. Arantes and J. Birge, "Computational results using interior point methods for two-stage stochastic programs", presented at TIMS/ORSA Joint National Meeting, Las Vegas, NE, 1990.

[2] E.R. Barnes, "A variation on Karmarkar's algorithm for solving linear programming problems", *Math. Programming* **36**, 174–182 (1990).

[3] D. Den Hertog, and C. Roos, "A survey of search directions in interior point methods for linear programming", Report 89-65, Delft University of Technology, Delft, The Netherlands, 1989.

[4] I.I. Dikin, "Iterative solution of problems of linear and quadratic programming", *Dokl. Akad. Nauk SSSR* **174**, 747–748 (1967).

[5] D.M. Gay, "Electronic mail distribution of linear programming test problems, *Mathematical Programming Society COAL Newsletter* **13**, 10–12 (1985).

[6] C.C. Gonzaga, "Search directions for interior linear programming methods", Memorandum No. UCB/ERL M87/44, Electronic Research Laboratory, College of Engineering, University of California, Berkeley, CA, 1987.

[7] N. Karmarkar, "A new polynomial time algorithm for linear Programming", *Combinatorica* **4**, 373–395 (1984).

[8] R.J. Vanderbei, M.S. Meketon and B.A. Freedman, "A modification of Karmarkar's linear programming algorithm", *Algorithmica* **1**, 395–407 (1986).

[9] R.J. Vanderbei, "Dense columns and interior point methods for LP", presentation at 14th International Symposium on Mathematical Programming, Amsterdam, The Netherlands, 1991.