ORSA Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Feature Article—Interior Point Methods for Linear Programming: Computational State of the Art

Irvin J. Lustig, Roy E. Marsten, David F. Shanno,

To cite this article:
Irvin J. Lustig, Roy E. Marsten, David F. Shanno, (1994) Feature Article—Interior Point Methods for Linear Programming: Computational State of the Art. ORSA Journal on Computing 6(1):1-14. https://doi.org/10.1287/ijoc.6.1.1

ORSA Journal on Computing
Vol. 6, No. 1, Winter 1994

0899-1499/94/0601-0001 $01.25

FEATURE ARTICLE

Interior Point Methods for Linear Programming: Computational State of the Art

IRVIN J. LUSTIG / *Program in Statistics and Operations Research, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ 08544; Email: irv@dizzy.cplex.com*

ROY E. MARSTEN / *School of Industrial Engineering and Operations Research, Georgia Institute of Technology, Atlanta, GA 30332*

DAVID F. SHANNO / *Rutgers Center for Operations Research, Rutgers University, New Brunswick, NJ 08903; Email: shanno@dantzig.rutgers.edu*

(Received: December 1992; revised: April 1993, July 1993; accepted: July 1993)

A survey of the significant developments in the field of interior point methods for linear programming is presented, beginning with Karmarkar's projective algorithm and concentrating on the many variants that can be derived from logarithmic barrier methods. Full implementation details of the primal-dual predictor-corrector code OB1 are given, including preprocessing, matrix orderings, and matrix factorization techniques. A computational comparison of OB1 with a state-of-the-art simplex code using eight large models is given. In addition, computational results are presented where OB1 is used to solve two very large models that have never been solved by any simplex code.

Interior point methods for mathematical programming problems were introduced by Frisch[16] and were developed as a tool for nonlinear programming by Fiacco and McCormick.[15] While Fiacco and McCormick noted that their proposed methods could be applied in full measure to linear programming, neither they nor any other researchers at that time seriously proposed that interior point methods would provide a viable alternative to the simplex method for actually solving linear programming problems.

Current interest in interior point methods for linear programming was sparked by the paper by Karmarkar,[27] which used projective transformations to demonstrate a polynomial time complexity bound for linear programming that was far better than any previously known bound. Shortly after the appearance of Karmarkar's paper, newspaper accounts claimed that the new method was significantly faster than existing implementations of the simplex method. Although these claims were never scientifically validated, they did capture the attention of the mathematical programming community, as indicated by the large number of papers (over 1300) published since 1984 (see Kranich[30] for a bibliography). Initial skepticism as to whether interior point algorithms could ever compete with simplex method implementations was quickly replaced by cautious optimism that interior point methods could in fact prove competitive.

Within two years of the publication of Karmarkar's method, an implementation of an interior point method to solve the dual problem, which became known as the dual affine variant (Adler et al.[1]), demonstrated overall superiority over a specific implementation of the simplex method, MINOS 4.0, on a set of linear programming problems collected by David Gay (Gay[18]), and distributed over NETLIB. Shortly thereafter, similar performance was demonstrated for a primal-dual path-following algorithm (McShane, Monma, and Shanno[40]), and serious development of full-blown interior point codes that could be comparatively tested against commercial simplex codes began.

Examination of the comparative performance of simplex and interior point codes had the interesting and important side effect of rekindling interest in the simplex method. During roughly the same period that interior point methods were being studied and implemented, huge advances in the computational efficiency of the simplex method were also achieved, dramatically lowering both the number of iterations and computer time to solve problems. These advances stem from such improvements as better crash basis procedures, better handling of degeneracy, better

Subject classifications: Programming: linear.
Other key words. Linear programming algorithms.

partial pricing, implementation of primal and dual steepest edge algorithms, faster and more stable matrix factorizations, better exploitation of cache memory, and better combined phase 1-phase 2 algorithms. Two new simplex codes, CPLEX (a trademark of CPLEX Optimization, Inc., Incline Village, NV)[9] and IBM's OSL Release 2 simplex code,[26] represent such a major improvement in simplex technology that if the original interior point implementations had been tested against these codes, it might well have discouraged further development of interior point technology. In 1987, the original small NETLIB test set of approximately 50 linear programs had virtually no problems for which interior point codes exhibited a clear advantage over the best simplex codes. This is due to the fact that interior point codes outperform simplex codes only on problems with, in general, thousands of rows and columns, and none of the initial NETLIB problems was sufficiently large.

Fortunately, as simplex codes improved, so did the technology of interior point codes. Also, larger models became available for comparative testing. These models did not easily lend themselves to efficient simplex starting bases and had highly degenerate optimal solutions. Some of these were added to the NETLIB test set. Many of the more interesting models are proprietary models currently in use at the organizations that constructed them. Due to their proprietary nature, these models have not been included in the NETLIB test set, but have been made available in carefully controlled circumstances for comparative testing. Many of these new models are significantly larger than any test models that existed as recently as three years ago.

This paper is organized to provide a brief tutorial on the recent history of interior point methods and implementation details of one successful interior point code. In Section 1, Karmarkar's projective method is described. Section 2 develops logarithmic barrier methods, and uses them to derive primal, dual, and primal-dual path-following algorithms, as well as their affine variants. Section 3 deals with the problem of initial feasibility, leading to contemporary infeasible interior point algorithms, whereas Section 4 shows how to modify the method to efficiently incorporate upper bounds and free variables. In Section 5, the predictor-corrector variant of the primal-dual method is developed, followed by a discussion in Section 6 of how many corrections to make. Section 7 illustrates the dichotomy between algorithms with strong theoretical properties and algorithms that are efficient in practice, and indicates how this gap is beginning to close. Section 8 presents issues involved in implementing an efficient interior point code. Computational results on 10 linear programs and comparisons of the primal-dual predictor-corrector code OB1 (Optimization with Barriers 1) with the OSL simplex code are documented in Section 9. These results show quite clearly that for the majority of large models that we have been able to test, the theoretical advantage in complexity of interior point methods is becoming apparent. Thus, the results of this paper demonstrate that the initial claims of the computational superiority of interior point methods are now beginning to appear. Section 10 closes with topics of current study.

1. Karmarkar's Projective Method

The linear programming problem considered by Karmarkar[27] is

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = 0, \\ & e^T x = 1, \\ & x \geqslant 0, \end{aligned} \tag{1}$$

where A is an $m \times n$ matrix, x and c are n-vectors, and e is the n-vector of all ones. Furthermore, Karmarkar assumes that at x^*, the optimal solution to (1), the optimal objective function value satisfies $c^T x^* = 0$. This is not the standard form for linear programming problems, which is defined in Equation (11) in the next section, but Karmarkar demonstrated how any linear programming problem can be put into this form.

Briefly, Karmarkar's algorithm begins with an initial estimate x^0 to the solution of (1), where $Ax^0 = 0$, $e^T x^0 = 1$, and $x^0 > 0$. The algorithm moves to a new estimate x^1 via the use of projective transformations. Let

$$T(x) = \frac{X_0^{-1} x}{e^T X_0^{-1} x}, \tag{2}$$

where the diagonal matrix X_0 is defined by

$$x_{jj} = x_j^0, \qquad j = 1, \ldots, n,$$

and $T(x)$ is the projective transformation that takes x^0 into $(1/n)e$. Define the matrix B by

$$B = \begin{bmatrix} AX_0 \\ e^T \end{bmatrix} \tag{3}$$

and the vector δ by

$$\delta = -\gamma [I - B^T (BB^T)^{-1} B] X_0 c, \tag{4}$$

where γ is a scalar step parameter. A new point ξ is then defined by

$$\xi = \frac{1}{n} e + \delta, \tag{5}$$

and a new point x^1 by the inverse projective transformation $T^{-1}(\xi)$ defined by

$$x^1 = T^{-1}(\xi) = \frac{X_0 \xi}{e^T X_0 \xi}. \tag{6}$$

Karmarkar then shows that for a proper choice of γ in (4), the algorithm determines an $\tilde{x}$ with $c^T \tilde{x} < 2^{-L} c^T x^0$ in $O(nL)$ iterations, where L is the number of bits needed to represent the entries in A and c.

Karmarkar's original algorithm dealt with the assumption that $c^T x^* = 0$ by means of a sliding objective function. A more elegant way of dealing with this is proposed by Todd and Burrell,[50] who note that if v^* is the optimal value of the objective function, the modified objective function

$$\begin{aligned} \hat{c}^T x &= c^T x - v^* e^T x \\ &= (c - v^* e)^T x \end{aligned} \tag{7}$$

has optimal value 0 (since $e^T x = 1$). Todd and Burrell then use duality theory to determine a convergent sequence of underestimates for the optimal value v^*.

In order to obtain the proof of complexity for his method, Karmarkar introduced the potential function

$$p(x) = \ln c^T x - \sum_{j=1}^{n} \ln x_j. \tag{8}$$

He was able to demonstrate that for the proper choice of γ in (4), the potential function was reduced by at least a constant amount each iteration, leading to the desired complexity result. Although the choice of γ in Karmarkar's initial paper proved much too small in practice, leading to large iteration counts and poor comparisons with the simplex method, many early investigations of Karmarkar's method suggested choosing γ by a line search of $p(x)$. As will be documented later, more effective large step methods have since been developed, obviating the need for line searches of $p(x)$ in general, but for some very difficult problems, these may yet prove of practical value.

As a final note on the potential function $p(x)$, Karmarkar's projective method can be derived as a constrained Newton method with respect to $p(x)$. This fact served to place the derivation of Karmarkar's method within the broader class of logarithmic barrier methods to be discussed in Section 2.

An early implementation difficulty with Karmarkar's method involved converting the standard form linear programming problem (11) to Karmarkar homogeneous form (1). Karmarkar's suggested method made the matrix A twice as large. A better method due to Tomlin[51] added a dense row to A. Gay[17] and de Ghellinck and Vial[20] demonstrated how to apply Karmarkar's method to problems in standard form, with a natural method for finding a feasible point.

A potentially more serious problem with Karmarkar's method as a viable computational method arises from a recent paper by Powell,[46] in which he demonstrates a problem with n variables where the number of iterations required by Karmarkar's method is $O(n)$, where n can be made arbitrarily large. Methods to be developed in subsequent sections have complexity of $O(\sqrt{n}\,L)$, and thus appear to have a worst case theoretical advantage.

To the best of our knowledge, there is no fully robust implementation of the projective method that is sufficiently efficient to be compared with state-of-the-art simplex codes. Goffin and Vial[22] and Bahn et al.[3] have successfully used projection methods for cutting plane algorithms, and Yamashita[54] has implemented a dual projective algorithm, but these implementations are prototype codes and they cannot be fairly compared with commercial simplex codes for solving general linear programs. As research intensified on interior point methods, new methods, described in the next section, took precedence in implementation, and although the projective method has continued to figure prominently in complexity analysis, it has been far less important in the development of general implemented algorithms.

2. Logarithmic Barrier Methods

Logarithmic barrier methods were introduced by Frisch[16] and developed by Fiacco and McCormick.[15] Initially, the concentration was on nonlinear problems of the form

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) \geqslant 0, \qquad i = 1, \ldots, m \end{aligned} \tag{9}$$

where $x = (x_1, \ldots, x_n)^T$. Logarithmic barrier methods transform (9) into a sequence of unconstrained problems of the form

$$\text{minimize} \qquad f(x) - \mu_k \sum_{i=1}^{m} \ln g_i(x), \tag{10}$$

where μ_k is a scalar barrier parameter that satisfies $\mu_k > 0$, with $\lim_{k \to \infty} \mu_k = 0$. The algorithm to solve (9) is then

(i) Choose $\mu_0 > 0$, $\epsilon > 0$, and x^0 such that $g_i(x^0) > 0$ for all $1 \leqslant i \leqslant m$.
(ii) Let $x^k = \arg(\min(f(x) - \mu_k \sum_{i=1}^{m} \ln g_i(x)))$.
(iii) If $\mu_k < \epsilon$, then stop. Otherwise, choose $\mu_{k+1} < \mu_k$, set $k = k + 1$, and go to (ii).

Fiacco and McCormick show that when $f(x)$ and $g_i(x)$ meet certain general conditions, the sequence $\{x^k\}$ converges to a solution of (9), and that $\lim_{k \to \infty} \mu_k / g_i(x^k) = \lambda_i$, where λ_i is the optimal Lagrange multiplier associated with $g_i(x)$.

2.1. Primal Log Barrier Methods for Linear Programming

The relationship between logarithmic barrier methods and Karmarkar's method was first noted by Gill et al.[21] They considered the linear programming problem in the standard form

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && Ax = b, \\ & && x \geqslant 0, \end{aligned} \tag{11}$$

transforming the problem to

$$\begin{aligned} &\text{minimize} && c^T x - \mu \sum_{j=1}^{n} \ln x_j \\ &\text{subject to} && Ax = b. \end{aligned} \tag{12}$$

The Lagrangian for (12) is

$$L(x, y, \mu) = c^T x - \mu \sum_{j=1}^{n} \ln x_j - y^T (Ax - b), \tag{13}$$

and the first order conditions for (12) are

$$\begin{aligned} \nabla_x L &= c - \mu X^{-1} e - A^T y = 0, \\ \nabla_y L &= -Ax + b = 0, \end{aligned} \tag{14}$$

where X is the diagonal matrix whose diagonal elements are the variables x_j, $1 \leqslant j \leqslant n$, and is denoted by X_k when evaluated at the iterate x^k.

With the assumption that there exists a strictly interior feasible point x^k, i.e., $x^k > 0$ and $Ax^k = b$, Newton's

method is applied to (14) in an attempt to determine a better estimate to the solution of (11). This yields the search direction

$$\Delta x^k = -\frac{1}{\mu_k} X_k P X_k c + X_k P e, \tag{15}$$

where

$$P = \left(I - X_k A^T (A X_k^2 A^T)^{-1} A X_k\right), \tag{16}$$

and the new estimate x^{k+1} to the optimal solution is

$$x^{k+1} = x^k + \alpha_k \Delta x^k \tag{17}$$

for an appropriate step length α_k. The barrier parameter μ_k is then reduced by $\mu_{k+1} = \rho \mu_k$, $0 < \rho < 1$, and the algorithm continues. Note that this is a major departure from the Fiacco–McCormick algorithm in that only one Newton step is taken for each value of μ_k.

The similarity between the directions defined by (15) and δ defined by (4) was noted by Gill et al.,[21] who show that if the algorithm with a search direction defined by (15) is applied to a problem in Karmarkar form (1), then at each iteration there exists a value of μ_k such that the search directions (4) and (15) are identical. Gay[17] then showed how to apply the Karmarkar algorithm to problems in the standard form (11). This research, coupled with the inclusion of the Todd-Burrell parameter underestimating the objective function, led to the result that Karmarkar's method is just a special case of general logarithmic barrier methods (see Gonzaga[23] and Shanno and Bagchi[48]), and also focused attention on the development of logarithmic barrier methods.

Independently of the work on logarithmic barrier methods, Barnes[4] and Vanderbei, Meketon, and Freedman[53] were developing what was to become known as the primal affine method. In this method, Δx^k in (15) is replaced by

$$\Delta x^k = -X_k P X_k c, \tag{18}$$

which is the limiting direction in (15) as $\mu_k \to 0$. It was later discovered that this method had been initially proposed by Dikin[12, 13] more than 15 years before Karmarkar's work.

To understand the relationship between the primal affine and primal logarithmic barrier methods, only the role of the barrier parameter μ_k needs to be considered. It is crucial for interior point methods to remain in the interior of the feasible region, yet from the very beginning, computational experience suggested that choosing α_k in (17) to get very close to the boundary of the region is most efficient. If the problem

$$\begin{aligned} &\text{minimize} && -\sum_{j=1}^{n} \ln(x_j) \\ &\text{subject to} && Ax = b, \end{aligned} \tag{19}$$

that tries to find the analytic center of the feasible region is considered (see Sonnevend[49]), then Newton's method applied to the first order conditions yields

$$\Delta x_k = X_k P e. \tag{20}$$

Thus, the search vector (15) is made up of a centering term to keep away from the boundary, and an affine term that leads toward an optimal solution. As $\mu_k \to 0$, optimality dominates, whereas for large μ_k, the method proceeds across the interior of the feasible region. As shown by den Hertog and Roos,[11] most interior point methods have search directions that are linear combinations of these two vectors.

The paper by Gill et al.[21] actually describes an implementation of a pure primal barrier method, with computational results generally inferior to the MINOS 5.1 simplex code, but better on a set of degenerate problems. These results motivated further computational experimentation with interior point methods because many important problems exhibit degeneracy, often to a large degree, and degeneracy can cause serious problems for simplex codes. The fact that the primal log barrier method solved the set of degenerate problems faster than MINOS was viewed as very encouraging. Because subsequent work has led to much more efficient algorithms, further details of the original algorithm will not be provided here.

As a final note on primal log barrier methods, Anstreicher[2] has shown that the original Fiacco–McCormick implementation of their algorithm has polynomial complexity when applied to linear programs. Thus a polynomial algorithm for these problems existed nearly 20 years before Karmarkar's work.

2.2. Dual Log Barrier Methods

The dual linear programming problem to the standard form primal problem (11) is

$$\begin{aligned} &\text{maximize} && b^T y \\ &\text{subject to} && A^T y \leqslant c. \end{aligned} \tag{21}$$

Adding dual slack variables z_j, $1 \leqslant j \leqslant n$, to (21) gives the equivalent dual form

$$\begin{aligned} &\text{maximize} && b^T y \\ &\text{subject to} && A^T y + z = c, \\ & && z \geqslant 0. \end{aligned} \tag{22}$$

Renegar[47] initially proposed applying Huard's[25] method of centers to (21) and derived an algorithm with $O(\sqrt{n}\,L)$ complexity in the number of iterations. Subsequently, this work was again shown to be a special case of logarithmic barrier methods (see Gonzaga[24]). Dual methods are easily derived by applying the logarithmic barrier method to (21) by writing the problem

$$\text{maximize} \quad b^T y + \mu \sum_{j=1}^{n} \ln\left(c_j - a_j^T y\right), \tag{23}$$

where a_j is the jth column of the matrix A. The first order conditions are

$$b - \mu A Z^{-1} e = 0, \tag{24}$$

where Z is the $n \times n$ diagonal matrix with elements $z_j = c_j - a_j^T y$. One step of Newton's method yields

$$\Delta y = \frac{1}{\mu}(AZ^{-2}A^T)^{-1}b - (AZ^{-2}A^T)^{-1}AZ^{-1}e, \quad (25)$$

where the first term in (25) represents a step toward optimality and the second term is a centering step in the dual space.

Again, as in the primal case, the dual affine variant is derived by letting $\mu \to 0$ in (25), yielding the direction

$$\Delta y = (AZ^{-2}A^T)^{-1}b. \quad (26)$$

This was also discovered independently of logarithmic barrier methods (see Adler et al.[1]). As previously noted, the dual affine algorithm of [1] was the first implemented algorithm to show superior performance against the MINOS 4.0 simplex code. The dual affine code was approximately four times faster than MINOS (running on defaults) on the NETLIB test set that was available at that time. These results were extremely important in motivating further computational work with the dual affine algorithm (e.g., Marsten et al.[39]). Although this work showed that interior point methods have great potential, the dual affine algorithm was soon replaced by the superior algorithm defined in the next section. In addition, den Hertog et al.[10] have used a dual log barrier method in a cutting plane application.

2.3. Primal-Dual Logarithmic Barrier Methods

The underlying theory of primal-dual interior point methods is due to Megiddo[41] and was originally developed into a convergent algorithm by Kojima, Mizuno, and Yoshise.[29] The algorithm can be easily derived by considering the first order conditions (14) of the primal problem (11), or alternatively, by applying the logarithmic barrier method to the dual problem (22) where dual slack variables have been added. Here, the problem is

$$\begin{aligned} \text{maximize} \quad & b^T y + \mu \sum_{j=1}^{n} \ln z_j \\ \text{subject to} \quad & A^T y + z = c, \end{aligned} \quad (27)$$

with the Lagrangian

$$L(x, y, z, \mu) = b^T y + \mu \sum_{j=1}^{n} \ln z_j - x^T(A^T y + z - c). \quad (28)$$

The first order conditions for (28) are

$$XZe = \mu e, \quad (29a)$$

$$Ax = b, \quad (29b)$$

$$A^T y + z = c, \quad (29c)$$

where X and Z are the previously defined diagonal matrices and e is the n-vector of all ones. Conditions (29b) and (29c) are the usual linear programming optimality conditions of primal and dual feasibility, whereas (29a) is the usual complementarity condition in the limit as $\mu \to 0$.

As before, Newton's method can be applied to the conditions (29), with resulting steps

$$\begin{aligned} \Delta y &= -(AXZ^{-1}A^T)^{-1}AZ^{-1}\nu(\mu), \\ \Delta z &= -A^T\Delta y, \\ \Delta x &= Z^{-1}\nu(\mu) - XZ^{-1}\Delta z, \end{aligned} \quad (30)$$

where $\nu(\mu) = \mu e - XZe$. An affine variant of (30) sets $\mu = 0$ at each step.

In comparing primal, dual, and primal-dual methods, it is first instructive to note that all construct a matrix of the form ADA^T, where D is diagonal. The content of D varies, but the computational work does not. This is also the same type of matrix used by Karmarkar's method.

Given this similarity, there are two immediate advantages that appear when examining the primal-dual method. The first is that for primal feasible x and dual feasible y and z, the exact current duality gap $c^T x - b^T y$ is always known. It can easily be shown that for a feasible point (x, y, z),

$$c^T x - b^T y = x^T z, \quad (31)$$

and thus an excellent measure of how close the given solution is to the optimal is always available. A second advantage of the primal-dual method is that it allows for separate step lengths in the primal and dual spaces, i.e.,

$$\begin{aligned} x^{k+1} &= x^k + \alpha_P^k \Delta x^k, \\ y^{k+1} &= y^k + \alpha_D^k \Delta y^k, \\ z^{k+1} &= z^k + \alpha_D^k \Delta z^k. \end{aligned} \quad (32)$$

This separate step algorithm was first implemented by McShane, Monma, and Shanno[40] and has proven highly efficient in practice, significantly reducing the number of iterations to convergence.

Thus far, the choice of the step length parameter has not been addressed. In all implementations that take a few iterations, a ratio test is first used to determine the largest steps that can be taken before either some x_j or, respectively, some z_j becomes negative. Let these respective maximum steps be denoted as $\hat{\alpha}_P$ and $\hat{\alpha}_D$. The subsequent step is then chosen to be a constant multiple ρ of the maximum step, i.e.,

$$\begin{aligned} \alpha_P &= \rho\hat{\alpha}_P, \\ \alpha_D &= \rho\hat{\alpha}_D. \end{aligned} \quad (33)$$

In our computational experience, $\rho = 0.95$ (or even $\rho = 0.9$) seems to be the largest possible safe step for a primal or dual affine variant, although a primal-dual affine variant is not practical for any value of ρ. However for (30), which contains the centering parameter μ, the value $\rho = 0.99995$ works extremely well in practice, with an additional condition that $\alpha_P \leq 1$ and $\alpha_D \leq 1$. Centering does allow for longer steps, and this largely accounts for the computational superiority of methods using barrier parameters as opposed to affine variants.

Published in ORSA Journal on Computing on February 01, 1994 as DOI: 10.1287/ijoc.6.1.1.
This article has not been copyedited or formatted. The final version may differ from this version.

3. Initial Feasibility

The primal methods developed in the previous section assume that an initial feasible point x^0 is available that satisfies $x^0 > 0$ and $Ax^0 = b$. This assumption can be handled by introducing an artificial column into (11). Let $x^0 > 0$ be specified. Then the problem

$$\begin{aligned} \text{minimize} \quad & c^T x + c_a x_a \\ \text{subject to} \quad & Ax + (b - Ax^0)x_a = b, \\ & x, x_a \geqslant 0, \end{aligned} \tag{34}$$

has an initial solution $x = x^0$ and $x_a = 1$. For c_a sufficiently large, it can be shown that (34) has the same solution set as (11) assuming that (11) has a feasible solution.

Similarly, an artificial column can be added to the dual problem (22), yielding the equivalent problem

$$\begin{aligned} \text{maximize} \quad & b^T y + b_a y_a \\ \text{subject to} \quad & A^T y + z + y_a(A^T y^0 + z^0 - c) = c, \\ & z \geqslant 0, \qquad y_a \leqslant 0. \end{aligned} \tag{35}$$

Here, $y_a = -1$ for an initial solution. Finally, a combination of the above techniques can be used to provide a primal problem

$$\begin{aligned} \text{minimize} \quad & c^T x + c_a x_a \\ \text{subject to} \quad & Ax + d_P x_a = b, \\ & d_D^T x + x_b = b_a, \\ & x, x_a, x_b \geqslant 0, \end{aligned} \tag{36}$$

and its dual

$$\begin{aligned} \text{maximize} \quad & b^T y + b_a y_a \\ \text{subject to} \quad & A^T y + d_D y_a + z = c, \\ & d_P^T y + z_a = c_a, \\ & y_a + z_b = 0, \\ & z, z_a, z_b \geqslant 0, \end{aligned} \tag{37}$$

where $d_P = b - Ax^0$ and $d_D = A^T y^0 + z^0 - c$, both of which are feasible at the initial point.

A numerically unpleasant feature of formulations (34)–(37) is the need to include artificial variables and their attendant large costs in the problem to be solved. Lustig[32] considers the possibility of allowing b_a and c_a to simultaneously go to infinity in (36) and (37), and shows that well defined limiting directions Δx, Δy, and Δz exist, which is not true for either the primal or the dual case. These results are not duplicated here since a preferred equivalent interpretation is given below.

The primal-dual search directions (30) are derived by applying Newton's method to the first order conditions (29), where under the assumption of feasibility we have the Newton system

$$\begin{aligned} Z\Delta x + X\Delta z &= \mu e - XZe, \\ A\Delta x &= 0, \\ A^T\Delta y + \Delta z &= 0. \end{aligned} \tag{38}$$

If we do not assume that the point (x, y, z) is feasible, applying Newton's method to (29) yields the system

$$\begin{aligned} Z\Delta x + X\Delta z &= \mu e - XZe, \\ A\Delta x &= b - Ax, \\ A^T\Delta y + \Delta z &= c - A^T y - z, \end{aligned} \tag{39}$$

which has the solution

$$\begin{aligned} \Delta y &= -(AXZ^{-1}A^T)^{-1}(AZ^{-1}\nu(\mu) - AXZ^{-1}r_D - r_P), \\ \Delta z &= -A^T\Delta y + r_D, \\ \Delta x &= Z^{-1}\nu(\mu) - Z^{-1}X\Delta z, \end{aligned} \tag{40}$$

where $r_D = c - A^T y - z$ and $r_P = b - Ax$. These search directions are the directions derived by Lustig.[32] Clearly, Newton's method can be applied in similar fashion to infeasible primal or dual methods. All computational results in Section 9 for the primal-dual method are based on Newton's method applied to the infeasible problem without the use of artificial variables.

4. Free Variables and Upper Bounds

All of the derivations of interior point methods assume $x_j \geqslant 0$ for all $j = 1, \ldots, n$. When free variables are contained in a problem, some type of translation of the variables is necessary. A standard transformation splits free variables as

$$x_j = x_j^+ - x_j^-, \tag{41}$$

where $x_j^+ \geqslant 0$ and $x_j^- \geqslant 0$. If this transformation is used and the simplex method is applied, it is easy to show that at most one of x_j^+ and x_j^- will be basic at any one time. However, because interior point methods attempt to find points that are as far from the boundary as possible, this can lead to both x_j^+ and x_j^- becoming extremely large. In the OB1 code, documented in Sections 8 and 9, free variables are split as in (41). In addition, both variables are translated so that, at each iteration, the smaller of the two variables is set to a constant although the difference remains unchanged. To date, this technique has worked well in practice, even for problems with hundreds of free variables.

Simplex codes handle upper bounds in a simple and straightforward manner. As shown in Choi, Monma, and Shanno,[8] upper bounds are just as easily handled by primal-dual interior point methods. The linear program with bounded variables is

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b, \\ & 0 \leqslant x \leqslant u, \end{aligned} \tag{42}$$

where some or all of the components of u may be infinite. The first order conditions are

$$\begin{aligned} Ax &= b, \\ x + s &= u, \\ A^T y + z - w &= c, \\ XZe &= \mu e, \\ SWe &= \mu e, \end{aligned} \tag{43}$$

Published in ORSA Journal on Computing on February 01, 1994 as DOI: 10.1287/ijoc.6.1.1.
This article has not been copyedited or formatted. The final version may differ from this version.

and the search directions resulting from applying Newton's method to (43) are

$$\begin{aligned}
\Delta y &= (A\Theta A^T)^{-1}[(b - Ax) \\
&\quad + A\Theta((c - A^T y - z + w) + \rho(\mu))], \\
\Delta x &= \Theta[A^T \Delta y - \rho(\mu) - (c - A^T y - z + w)], \\
\Delta z &= \mu X^{-1} e - Ze - X^{-1} Z \Delta x, \\
\Delta w &= \mu S^{-1} e - We + S^{-1} W \Delta x, \\
\Delta s &= -\Delta x, \qquad (44)
\end{aligned}$$

where $\Theta = (X^{-1}Z + S^{-1}W)^{-1}$ and $\rho(\mu) = \mu(S^{-1} - X^{-1})e - (W - Z)e$. Therefore, only the diagonal matrix and the right-hand side change, but the essential computational work remains the same. Note that in the derivation presented by Choi, Monma, and Shanno,[8] it is assumed that the current iterate always satisfies $x_j + s_j = u_j$ when $u_j < \infty$.

5. Mehrotra's Predictor–Corrector Method

The primal-dual algorithm for problems with bounded variables, described in the previous three sections, represents the first algorithm developed in OB1, as documented by Lustig, Marsten, and Shanno.[34] Shortly thereafter, Mehrotra[43] proposed a predictor–corrector method that can be derived directly from the first order conditions (29). By substituting $x + \Delta x$, $y + \Delta y$, and $z + \Delta z$ in (29), it is then desired that the new estimate satisfies

$$\begin{aligned}
(X + \Delta X)(Z + \Delta Z)e &= \mu e, \\
A(x + \Delta x) &= b, \\
A^T(y + \Delta y) + z + \Delta z &= c. \qquad (45)
\end{aligned}$$

Collecting terms gives the system

$$X\Delta z + Z\Delta x = \mu e - XZe - \Delta X \Delta Z e, \qquad (46a)$$

$$A\Delta x = b - Ax, \qquad (46b)$$

$$A^T \Delta y + \Delta z = c - A^T y - z, \qquad (46c)$$

where ΔX and ΔZ are $n \times n$ diagonal matrices with elements Δx_j and Δz_j, respectively. Examination shows that (46) is identical to (39) with the exception of the nonlinear term $\Delta X \Delta Z e$ in (46a). Mehrotra proposed first solving the affine system

$$\begin{aligned}
X\Delta \hat{z} + Z\Delta \hat{x} &= -XZe, \\
A\Delta \hat{x} &= b - Ax, \\
A^T \Delta \hat{y} + \Delta \hat{z} &= c - A^T y - z, \qquad (47)
\end{aligned}$$

and then substituting the vectors $\Delta \hat{x}$ and $\Delta \hat{z}$ found by solving (47) for the $\Delta X \Delta Z e$ term in the right-hand side of (46). Furthermore, he suggested testing the reduction in complementarity $(x + \alpha_P \Delta \hat{x})^T (z + \alpha_D \Delta \hat{z})$, where α_P and α_D are again chosen to insure $x > 0$ and $z > 0$. If we let

$$\hat{g} = (x + \alpha_P \Delta \hat{x})^T (z + \alpha_D \Delta \hat{z}), \qquad (48)$$

then Mehrotra's estimate for μ is

$$\mu = \left(\frac{\hat{g}}{x^T z}\right)^2 \left(\frac{\hat{g}}{n}\right). \qquad (49)$$

This chooses μ to be small when the affine direction produces a large decrease in complementarity and chooses μ to be large otherwise. The predictor-corrector algorithm, with a minor variant of Mehrotra's choice of μ, is the current algorithm implemented in OB1 (see Lustig, Marsten, and Shanno[38]).

6. How Many Corrections?

In [43], Mehrotra motivates the predictor-corrector method as a power series. Clearly, the algorithm of the previous section can easily be extended to a higher order power series by continuing to substitute at each step the Δx and Δz terms found by solving (46) back into the right-hand side of (46a) so that the algorithm is using multiple corrections. For example, when multiple corrections are applied to the NETLIB test problem `afiro` at a primal and dual feasible point, we obtain Table I. Assuming that the goal is to achieve the maximum reduction in complementarity, the optimal number of corrections is four in this case. After four corrections, the power series begins to diverge.

Although multiple corrections often produce lower complementarity, each correction requires a new solution to (46). Because the matrix does not change, only a single factorization needs to be done, but multiple solutions to $(A\Theta A^T)\Delta y = r$ are required. Multiple corrections have been extensively studied by Carpenter et al.[7] This study concludes that the most efficient number of corrections for a general algorithm is one.

Furthermore, even a single correction can increase complementarity. Lustig, Marsten, and Shanno[36] derive a test to determine whether the correction term should be computed or a straight primal-dual step should be taken. Although this test has little computational effect on any problem that has been solved to date, it is important in terms of guaranteed convergence, which is discussed in the next section.

7. Theoretical Issues

As noted in Section 1, Karmarkar's algorithm was motivated by his desire to find a method for linear program-

Table I. Effect of Multiple Corrections on `Afiro`

Total Corrections	Steplength	Complementarity After Step
0	0.626	466
1	0.685	392
2	0.750	311
3	0.827	215
4	0.886	141
5	0.773	282
6	0.784	268
7	0.206	988
8	0.200	996
9	0.016	1230

Published in ORSA Journal on Computing on February 01, 1994 as DOI: 10.1287/ijoc.6.1.1.
This article has not been copyedited or formatted. The final version may differ from this version.

ming with provably good complexity. Renegar's dual method of centers[47] was similarly motivated. Since the inception of Karmarkar's complexity analysis, literally hundreds of papers have been written on the complexity of different variants of projective and logarithmic barrier methods. No attempt is made to survey these results here. The interested reader is referred to the excellent survey by Gonzaga.[24]

Until recently, there has been a large gulf between theory and practice. The original paper by Kojima, Megiddo, and Mizuno[28] on the primal-dual method proved an $O(nL)$ complexity with a rather convoluted choice of step length. Monteiro and Adler[45] quickly improved this to an $O(\sqrt{n}\,L)$ complexity result with $\alpha_P = \alpha_D = 1$ at each step, where μ is reduced so slowly from iteration to iteration that the algorithm is hopelessly inefficient. Further results by Ye, Gonzaga, and others (see Gonzaga[24]) began to allow for larger steps, but these variants still did not correspond to implemented algorithms.

Furthermore, theoretical results predict that the number of iterations will grow slowly. To date, the $O(\sqrt{n}\,L)$ bound is the best achieved. In practice, the number is even smaller. As the results in Section 9 will demonstrate, the number of iterations remains very small, and appears to be $O((\log n)L)$. Lustig, Marsten, and Shanno[33] present empirical evidence demonstrating this behavior.

The complexity issue became further clouded with the removal of artificial variables and the introduction of infeasible interior point algorithms. All of the complexity results had been derived using strictly feasible points. In fact, not only was the complexity of infeasible interior point algorithms unknown, there was no proof of global convergence of the implemented algorithms. In an extremely interesting recent paper, Kojima, Mizuno, and Yoshise[29] examine an infeasible primal-dual method by defining a critical neighborhood $\mathcal{N}$ consisting of points satisfying the three conditions

$$x_j z_j \geqslant \gamma x^T z/n, \qquad 1 \leqslant j \leqslant n, \tag{50a}$$

$$x^T z \geqslant \gamma_P \|Ax - b\| \quad \text{or} \quad \|Ax - b\| \leqslant \epsilon_P, \tag{50b}$$

$$x^T z \geqslant \gamma_D \|A^T y + z - c\| \quad \text{or} \quad \|A^T y + z - c\| \leqslant \epsilon_D, \tag{50c}$$

where γ, γ_P, and γ_D are specified constants, and ϵ_P and ϵ_D are the desired infeasibility tolerances. The values β_1, β_2, and β_3 are chosen as constants satisfying $0 < \beta_1 < \beta_2 < \beta_3 < 1$. Let $\bar{\alpha}^k$ be the maximum value of $\bar{\alpha}$, $0 \leqslant \bar{\alpha} \leqslant 1$, such that

$$(x^k, y^k, z^k) + \alpha(\Delta x, \Delta y, \Delta z) \in \mathcal{N} \quad \text{and} \tag{51a}$$

$$(x^k + \alpha \Delta x)^T (z^k + \alpha \Delta z) \leqslant (1 - \alpha(1 - \beta_2))(x^k)^T z^k \tag{51b}$$

hold for every $\alpha \in [0, \bar{\alpha}]$. They then choose $\alpha_P^k \in (0, 1]$, $\alpha_D^k \in (0, 1]$, and $(x^{k+1}, y^{k+1}, z^{k+1})$ such that

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k + \alpha_P^k \Delta x, y^k + \alpha_D^k \Delta y, z^k + \alpha_D^k \Delta z) \in \mathcal{N} \quad \text{and} \tag{52a}$$

$$(x^{k+1})^T z^{k+1} \leqslant (1 - \bar{\alpha}^k(1 - \beta_3))(x^k)^T z^k. \tag{52b}$$

With the additional requirement that $\mu = \beta_1 (x^k)^T z^k/n$, they then show global convergence to an optimal solution or that $\|(x, z)\| \geqslant \omega^*$ for an arbitrarily large value ω^*. A polynomial time complexity bound for the infeasible primal-dual algorithm is given by Mizuno.[44]

In Lustig, Marsten, and Shanno,[36] the global convergence theory is extended to the predictor-corrector algorithm. It is then shown that the globally convergent algorithm is very efficient in practice, and helps greatly in implementing a warm-start interior point algorithm, i.e., an algorithm that can use an optimal solution to one linear program as the starting point for solving a second linear program. Although the polynomial time complexity of this exact algorithm is still under study, this recent work seems to be finally closing the gap to the point where implemented algorithms can be demonstrated to have provably good theoretical properties.

8. Implementation Issues

This section briefly describes the implementation issues that arise in constructing a logarithmic barrier interior point code. While everything discussed here has been implemented in the primal-dual predictor-corrector interior point code OB1, the issues are essentially generic in that they are important for a fast implementation of any logarithmic barrier or projective interior point method.

8.1. The Cholesky Factorization

The single largest amount of computation time in solving most linear programming problems using interior point methods is expended on computing the Cholesky factorization of $A\Theta A^T$, where the diagonal matrix Θ is determined by the specific choice of algorithm, as previously explained. The Cholesky factorization computes a lower triangular matrix L such that

$$A\Theta A^T = LDL^T. \tag{53}$$

Because the goal of implementing interior point methods is to solve large problems quickly, it is important that the factorization (53) be done as quickly as possible while minimizing the storage requirement for L. In order to accomplish the latter, OB1 offers the option of two matrix orderings, the multiple minimum degree ordering (see Liu[31]) and the minimum local fill ordering (see Duff, Erisman, and Reid[14]). Both are heuristics for permuting the rows of the matrix $A\Theta A^T$ in order to minimize the fill-in in L. Generally, the minimum local fill algorithm produces a sparser L, but at higher initial cost to obtain the ordering. The multiple minimum degree algorithm produces a denser L, but often at a significant savings in the cost of computing the ordering.

Once the ordering has been completed, OB1 computes the Cholesky factors (53) using a sparse column Cholesky factorization (see George and Liu[19]). This again can be implemented in two ways. The first way is called a leftward looking or "pulling" Cholesky, where all previously computed columns that contribute to the factorization are

addressed as each new column is computed. The second way is called a rightward looking or "pushing" Cholesky, where, after each column is computed, its contribution to all future columns is computed and stored. The latter is more complex to implement in terms of manipulating the data structures of the factorization, but has the advantage of better exploitation of cache memory. OB1 again offers the choice of Cholesky algorithms.

Modern computer architectures have different features that can enhance the performance of an interior point implementation. These features include cache memory, pipelining, vectorization, and superscalar capabilities. A discussion of these specific issues is beyond the scope of this paper. One must evaluate the relative expense of calculations involving integers versus calculations involving floating point numbers. Experience in solving problems on various computers indicates that each of the four possible combinations of ordering algorithms and Cholesky algorithms is best for a different architecture. Denote the two Cholesky algorithms by L (for left) and R (for right), and the two orderings D (for multiple minimum degree) and F (for minimum local fill). Table II gives the most efficient pairing for four standard architectures. The different combinations are needed because the rightward looking Cholesky exploits the cache memories effectively on the DECstation 5100 and RS/6000, whereas the extra overhead of the rightward algorithm cannot be recovered by the cache on the Sparcstation or on the CRAY, which has no cache, but does have vectorization. In addition, the superscalar architecture of the RS/6000 and the vector capabilities of the CRAY make it less expensive to do more vectorized floating point calculations in the factorization in order to save the logical, and hence not vectorizable, overhead of the more expensive ordering.

An important issue in the Cholesky factorization is the dense window. All ordering schemes force most of the nonzeros of L to the lower right-hand corner, causing the last set of columns in L to be dense. When the columns are sufficiently dense, indirect addressing may be dispensed with, significantly speeding up the algorithm at the expense of a little extra storage.

Finally, OB1 uses the concepts of supernodes and loop unrolling to further speed up calculations. These are not discussed here because they are properly the subject of a survey on numerical linear algebra rather than interior point methods. The interested reader is referred to Lustig, Marsten, and Shanno.[37]

Table II. Best Combination of Cholesky Algorithm and Ordering on Various Architectures

Platform	Cholesky	Ordering
Sun Sparcstation 1 +	L	F
DECstation 5100	R	F
IBM RS/6000	R	D
CRAY Y/MP	L	D

8.2. Problem Reduction

The most important issue in creating a fast implementation of an interior point method is clearly a fast implementation of the numerical linear algebra. For large linear programs, the second most important issue is preprocessing to reduce problem size. Many large models are generated by matrix generators and modeling programs that create linear programs where preprocessing is very effective. The preprocessor implemented in OB1 uses simple inferences either to reduce problem size or to determine that the problem is infeasible. The types of reductions used by OB1 are now briefly described.

1. Empty rows are either removed because they are redundant or the empty row is used to determine that the problem is infeasible, depending on the row type and the right-hand side.
2. Empty columns are used to set variables at either their upper or lower bounds or zero. Otherwise, it is determined that the problem is dual infeasible depending upon the bounds and the cost coefficient.
3. Fixed columns are removed and the right-hand side altered.
4. Infeasible bounds ($l_j > u_j$ for some j) are used to determine that the problem is infeasible.
5. Dual redundancies are used to determine that the dual is infeasible or to fix primal variables at bounds and remove them from the model.
6. By substituting appropriate upper and lower bounds for variables in rows, redundant rows are identified, or a set of variables that must be set to bounds is determined.
7. Rows reduced to simple bounds are either removed as redundant or used to improve variable bounds.
8. Columns with only a single nonzero coefficient are used to change the row type of the corresponding row or to eliminate the variable from the problem.
9. Nonnegative elements of rows can be used to estimate an upper bound on a variable. The minimum of all such implied upper bounds can be computed, and this can be used to determine redundant rows.
10. Equality rows with one positive element and all nonnegative variables can be used to pivot the corresponding variable out of the model.
11. Equality rows with exactly two variables can be used to pivot one of the variables out of the model.

In OB1, these reductions are applied in two phases. First, Rules 10 and 11 are repetitively applied, followed by repeated applications of Rules 1 through 9. After a solution is found to the reduced problem, a solution is found to the original problem by using an "uncrushing" technique. Because of the complexity of this uncrushing algorithm, Rules 10 and 11 are not applied again after the second phase. Overall, these rules have proven very effective in producing significant reductions in large models. The computational results of Section 9 will document experience with a number of large models.

Published in ORSA Journal on Computing on February 01, 1994 as DOI: 10.1287/ijoc.6.1.1.
This article has not been copyedited or formatted. The final version may differ from this version.

8.3. Starting Point

All interior point methods are sensitive to the initial estimate x^0 to the solution. The starting point currently used by OB1 is documented in Lustig, Marsten, and Shanno.[38] Briefly, it computes an estimate $\bar{x}^0 = A^T(AA^T)^{-1}b$, and adjusts all x_j that are below a threshold value to a new value at or above the threshold value. The slack variables for bounds are also set above a threshold value, so that initial bound infeasibility in the constraint $x + s = u$ is allowed. For the dual variables, y^0 is set to 0 and the vectors z^0 and w^0 are set in order to attempt to satisfy the dual feasibility constraints while again staying above a certain threshold. In OB1, the default primal and dual threshold values may be altered by changing them in a control file supplied by the user, called the SPEC file.

Although the rather complex starting point used as a default by OB1 was determined after extensive computational experimentation, almost any model can be solved in a smaller number of iterations by experimenting to find a problem-specific good starting point. The best choice of a default starting point is still very much an open question, not unlike determining the best initial starting basis for a simplex algorithm.

9. Computational Results

Initial computational studies comparing interior point codes to simplex codes concentrated on the NETLIB test set and generally compared performance against MINOS (for example, see Lustig, Marsten, and Shanno[34]). One difficulty with such a comparison is that the NETLIB test set, although initially extremely valuable, has been made uninteresting by the advances in both simplex and interior point methods. Most of the NETLIB problems can be solved in under 1 minute on a scientific workstation. Furthermore, both CPLEX and the OSL Release 2 simplex codes are anywhere from 2 to 10 times faster than MINOS, which is basically a nonlinear programming code.

One result of the early comparisons is irrefutable. Namely, for small problems, simplex codes are far more efficient than interior point codes. An immediate question arises as to what is small. In our experience, any problem where the sum of the number of rows plus the number of columns is less than 2000 is considered small and much more likely to be solved faster by simplex codes. For moderate problems, when the number of rows plus the number of columns is less than 10,000, results get more interesting. Here, good simplex codes and interior point codes compete more evenly in that, overall, no one method shows a clear advantage. On specific models, however, the differences can be very dramatic depending on the structure of the nonzeros in the matrix A.

As work progressed on both simplex and interior point codes, larger models became available for testing. An initial set of comparative results of OB1 versus OSL Release 1 simplex is documented in Lustig, Marsten, and Shanno.[35] For this paper, we have collected eight recently formulated real-world (as opposed to randomly generated) models of a size that were previously considered impossibly large. Here we have tested OB1 versus the OSL Release 2 simplex code. Other barrier codes, such as OSL barrier or LOQO,[52] would show somewhat different results, as would the CPLEX simplex code. However, the results shown here do accurately indicate differences between interior and simplex methods. It is indicative of the rapid pace of recent advances that these models are now solved by both methods on scientific workstations. As will be seen, however, the interior point method has a pronounced advantage on all but one model. While this can be interpreted as a demonstration that the predicted advantage of interior point codes on large models is simply becoming evident, we hasten to add one note of caution. As stated, these models are real and proprietary and were made available to us for testing because they were difficult or impossible to solve with the technology available to the modeler. Thus, they may represent a somewhat biased sample. Nonetheless, they clearly demonstrate that large real models exist for which interior point methods are vastly superior to the best contemporary simplex codes.

In addition, results are included for two extremely large models constructed for Delta Air Lines that were successfully solved by OB1 on a CRAY Y/MP. To the best of our knowledge, these problems have not been solved by any simplex code. Again, these are real-world models and present further strong evidence that interior point methods represent a significant advance in the state of the art.

Before presenting the results, it is important to state how the comparative runs were done. All comparisons are against the OSL Release 2 simplex code. This code gives the user the option of either a primal simplex or a dual simplex algorithm and gives four choices for a crash basis that can be used with either algorithm. All eight options were run on each model. Although someone experienced with both the model and the OSL simplex code could undoubtedly improve upon these particular times by manipulating the model or the solution technique, it is our contention that the average user will probably run a particular model with default settings using the crash option that the user finds best for that model, if indeed the user is willing to experiment with all possible starts. Against this, OB1 was always run on pure defaults, exactly as the average user would try to solve a problem. In one case, it is then shown how OB1 can be improved by altering the starting point in the control file supplied by the user (the SPEC file).

Tables III–V depict the original sizes of the problems and the reductions in model size after the problem is preprocessed, as discussed in Section 8. The results of running each of the eight models on an IBM RS/6000 Powerstation 530 using OSL Release 2 simplex, and OB1 are shown in Tables VI through XIII. For OSL, iteration counts and CPU times (in seconds) are given for each pairing of crash option and simplex algorithm (primal or dual). For OB1, the CPU time and number of interior point iterations are given, as well as the number of nonzeros in the Cholesky factorization. For both programs, preprocessing time is included and is usually less than 5% of the total time. Finally, as Tables IV and V demonstrate, OSL and OB1 do not produce identical reductions. Thus, the tests can be considered

Published in ORSA Journal on Computing on February 01, 1994 as DOI: 10.1287/ijoc.6.1.1.
This article has not been copyedited or formatted. The final version may differ from this version.

Table III. Original Problem Size

Model	Rows	Columns	Non-Zeros
energy1	16,223	28,568	88,340
energy2	8,335	21,200	161,160
fuel	18,401	33,905	205,789
schedule	23,259	29,342	75,520
continent	10,377	57,253	198,214
car	43,387	107,164	189,864
energy3	27,145	31,053	268,153
initial	27,441	15,128	95,971

Table IV. Problem Size, after Pre-processing by OB1

Model	Rows	Columns	Non-Zeros
energy1	12,237	25,964	81,788
energy2	6,665	20,656	142,641
fuel	13,330	31,511	186,284
schedule	5,156	12,506	37,174
continent	6,869	57,158	183,649
car	43,387	107,164	189,864
energy3	9,519	31,053	192,138
initial	19,100	11,173	79,249

Table V. Problem Size, after Pre-processing by OSL

Model	Rows	Columns	Non-Zeros
energy1	12,907	24,245	77,253
energy2	6,940	18,497	128,132
fuel	14,918	30,762	172,960
schedule	7,923	14,689	43,709
continent	6,924	45,777	184,035
car	43,387	107,164	189,864
energy3	9,640	28,649	195,516
initial	19,193	10,956	79,380

Table VI. Results for energy1

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	24,153	58,802	3,559.45	11,748.93
2	28,098	44,953	3,961.85	8,556.77
3	29,731	48,281	4,199.95	8,845.63
4	28,491	32,439	4,233.47	6,333.40

OB1 Iterations	OB1 Time	Cholesky Nonzeros
35	215.61	143,258

Table VII. Results for energy2

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	61,116	130,653	11,082.32	21,638.78
2	73,228	109,994	14,194.86	18,295.98
3	68,398	106,532	11,160.70	17,695.70
4	74,764	106,022	14,548.50	17,418.24

OB1 Iterations	OB1 Time	Cholesky Nonzeros
44	403.49	466,844

Table VIII. Results for fuel

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	78,216	200,306	20,623.28	53,817.91
2	89,028	194,044	23,895.31	51,828.05
3	82,425	145,288	19,855.74	39,607.27
4	77,744	149,102	20,211.66	40,315.80

OB1 Iterations	OB1 Time	Cholesky Nonzeros
66	2,362.95	1,260,697

Table IX. Results for schedule

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	64,376	67,785	8,324.10	12,271.12
2	74,432	47,847	9,327.44	7,891.47
3	75,476	52,026	9,659.25	8,476.78
4	72,887	59,039	9,508.50	9,533.72

OB1 Iterations	OB1 Time	Cholesky Nonzeros
36	1,037.33	860,248

Table X. Results for continent

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	15,797	103,588	1,954.03	30,276.37
2	26,572	72,734	3,072.82	13,915.88
3	17,486	92,490	2,066.85	25,730.60
4	23,783	55,630	2,832.91	11,091.39

OB1 Iterations	OB1 Time	Cholesky Nonzeros
64	771.48	313,078

Table XI. Results for car

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	21,194	26,232	8,588.33	13,084.82
2	21,065	25,328	8,563.74	13,972.07
3	31,006	35,354	10,027.83	18,593.98
4	31,006	35,354	10,018.56	18,537.19

OB1 Iterations	OB1 Time	Cholesky Nonzeros
53	645.21	188,153

Table XII. Results for energy3

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	26,313	131,312	5,283.86	44,468.94
2	57,212	71,043	10,851.67	16,459.54
3	26,435	157,730	5,270.91	52,763.17
4	75,886	76,018	14,271.20	19,741.72

OB1 Iterations	OB1 Time	Cholesky Nonzeros
86	865.31	341,084
64	691.95[a]	

[a]With altered starting point.

Table XIII. Results for initial

OSL Crash	OSL Iterations		OSL Time	
	Primal	Dual	Primal	Dual
1	33,534	7,134	4,942.16	1,646.88
2	24,725	4,716	3,151.83	831.81
3	> 19,671[a]	5,412		1,085.50
4	26,474	5,046	6,256.45	1,026.25

OB1 Iterations	OB1 Time	Cholesky Nonzeros
58	9,252.81[b]	6,753,621

[a]Primal with crash 3: OSL returned with the message "Numerical difficulties: the iterations are not yielding better solutions." The objective value was 69,765,303 while the optimum is at 56,007,256.

[b]OB1 was run on a Model 550 to solve *initial*. The Model 550 is about 40% faster than the Model 530 that was used for the other runs.

comparative tests of complete systems rather than algorithms, although it is our belief that if both codes used the same problem reductions, the results would change very little.

As can be seen from the first seven models, OB1 always outperformed the OSL simplex code by a significant amount, ranging from at least 2.5 to 20 times faster. Furthermore, the iteration counts for all models using OB1 were very low, as predicted by theory. The eighth model, however, shows precisely how interior point methods can sometimes require large computation times to solve a problem. For the problem *initial*, the number of nonzeros in A is 79,249, yet the number in L is 6,753,621. When this sort of catastrophic fill-in occurs, interior point methods often cannot compete with simplex codes. For the most part, however, the performance is excellent.

Two large models from Delta Air Lines were run on the CRAY Y/MP M-90. The results are included in Tables XIV and XV.

It is our contention that these last two models represent the state of the art of the type of model that can currently be solved by interior point methods. To the best of our knowledge, neither model has been solved by any simplex code, due to these models' massive size and degeneracy. Furthermore, the second model illustrates how parallelism can be applied to interior point methods. The CRAY basic linear algebra routines to parallelize a dense Cholesky factorization were used. As the dense window contained more than half the nonzeros of L and computations were perfectly balanced, overall computation time was reduced by a factor of four by using eight processors. Sparse parallel factorizations that are equally efficient should further improve this performance.

Table XIV. Results for Delta Model 1

rows = 99,533 columns = 117,117 $|A|$ = 407,068

After pre-processing:

rows = 32,229 columns = 68,539 $|A|$ = 240,371

Number of Cholesky nonzeros = 32,974,664

Solution: 54 iterations to 8 digits

Task	Time (CRAY CPU secs.)
pre-processing	55.55
ordering	458.75
solution	28,940.39
post-processing	62.95

Table XV. Results for Delta Model 2

rows = 270,796 columns = 303,986 $|A|$ = 849,216

After pre-processing:

rows = 45,116 columns = 101,790 $|A|$ = 342,180

Number of Cholesky nonzeros = 105,394,294

Solution: 40 iterations to 6 digits

Task	Time (Cray CPU secs.)
pre-processing	128 secs
ordering	1094 secs
solution	82 hours[a]

[a]22 hours wall-clock time

10. Conclusions and Future Directions

This paper summarizes the extremely rapid development of interior point methods for linear programming and discusses issues important for their implementation. Computational results clearly demonstrate that great progress has been made in the capability of routinely solving larger models by either simplex or interior point methods. Furthermore, the results show that for many large models, interior point codes are significantly faster.

Three current efforts are underway to further improve interior point methods. First, work is underway to design and implement a full warm-start capability in OB1, based on the work of Lustig, Marsten, and Shanno,[36] where warm-starting an interior point code from a previously obtained solution to a perturbation of the problem is demonstrated. In some cases, this should make interior point methods competitive with simplex methods using an advanced basis.

A second area of current interest is in determining an analytically valid way of detecting infeasibility and unboundedness in interior point codes. Often, these conditions are detected by the preprocessor. Otherwise, the condition is generally manifested by either the primal or dual objective function becoming unbounded. However, the determination of an analytically valid and numerically computable set of conditions without reintroducing artificial variables remains an interesting topic.

Finally, anyone actively engaged in using contemporary linear programming codes is quickly convinced that an ideal code contains both interior point and simplex methods, with an efficient basis recovery method included to easily bridge between the methods. Bixby and Saltzman[6] present an initial study of this issue. Recent work by Bixby and Lustig indicates that such a crossover, based on Megiddo's strongly polynomial algorithm,[42] will prove very efficient in practice. When an efficient crossover exists, a combination of the two algorithms can be very efficient (see Bixby et al.[5]). The future practice of solving linear programs should be able to take advantage of both algorithms to solve extremely difficult, if not presently unsolvable, problems.

Acknowledgments

The research of David Shanno is sponsored by the Air Force Office of Scientific Research, Air Force System Command under grant AFOSR-92-J0046. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notations thereon.

The authors would like to thank three anonymous referees for their comments on an initial draft of this paper.

References

1. I. Adler, N.K. Karmarkar, M.G.C. Resende and G. Veiga, 1989. An Implementation of Karmarkar's Algorithm for Linear Programming, *Mathematical Programming 44*, 297–335.
2. K.M. Anstreicher, 1990. On Long Step Path Following and SUMT for Linear and Quadratic Programming, Technical Report, Yale School of Management, Yale University, New Haven, CT.
3. O. Bahn, J.L. Goffin, J.P. Vial and O.D. Merle, 1991. Implementation and Behavior of an Interior Point Cutting Plane Algorithm for Convex Programming: An Application to Geometric Programming, Working Paper, University of Geneva, Geneva, Switzerland.
4. E.R. Barnes, 1986. A Variation on Karmarkar's Algorithm for Solving Linear Programming Problems, *Mathematical Programming 36*, 174–182.
5. R.E. Bixby, J.W. Gregory, I.J. Lustig, R.E. Marsten and D.F. Shanno, 1992. Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods, *Operations Research 40*, 885–897.
6. R.E. Bixby and M.J. Saltzman, 1992. Recovering an Optimal LP Basis from an Interior Point Solution, Technical Report 607, Department of Mathematical Sciences, Clemson University, Clemson, SC.
7. T.J. Carpenter, I.J. Lustig, J.M. Mulvey and D.F. Shanno, 1993. Higher Order Predictor-Corrector Interior Point Methods with Application to Quadratic Objectives, *SIAM Journal on Optimization 3*, 696–725.
8. I.C. Choi, C.L. Monma and D.F. Shanno, 1990. Further Development of a Primal-Dual Interior Point Method, *ORSA Journal on Computing 2*, 304–311.
9. CPLEX Optimization, Inc., 1993. *Using the CPLEX™ Callable Library and CPLEX™ Mixed Integer Library*, Incline Village, Nevada.
10. D. den Hertog, J. Kaliski, C. Roos and T. Terlaky, 1992. A Logarithmic Barrier Cutting Plane Method for Convex Programming, Technical Report, TUDelft, The Netherlands.
11. D. den Hertog and C. Roos, 1991. A Survey of Search Directions in Interior Point Methods for Linear Programming, *Mathematical Programming 52*, 481–509.
12. I.I. Dikin, 1967. Iterative Solution of Problems of Linear and Quadratic Programming, *Doklady Akademii Nauk SSSR 174*, 747–748. Translated in: *Soviet Mathematics Doklady 8*, 674–675, 1967.
13. I.I. Dikin, 1974. On the Convergence of an Iterative Process, *Upravlyaemye Sistemi 12*, 54–60. (in Russian).
14. I.S. Duff, A. Erisman and J. Reid, 1986. *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, England.
15. A.V. Fiacco and G.P. McCormick, 1968. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley & Sons, New York.
16. K.R. Frisch, 1955. The Logarithmic Potential Method for Convex Programming, Institute of Economics, University of Oslo, Oslo, Norway (unpublished manuscript).
17. D.M. Gay, 1987. A Variant of Karmarkar's Linear Programming Algorithm for Problems in Standard Form, *Mathematical Programming 37*, 81–90.
18. D.M. Gay, 1988. Electronic Mail Distribution of Linear Programming Test Problems, *Mathematical Programming Society COAL Newsletter*.
19. A. George and J. Liu, 1981. *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ.
20. G. de Ghellinck and J.P. Vial, 1986. A Polynomial Newton Method for Linear Programming, *Algorithmica 1:4*, 425–453.
21. P.E. Gill, W. Murray, M.A. Saunders, J.A. Tomlin and M.H. Wright, 1986. On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method, *Mathematical Programming 36*, 183–209.
22. J.L. Goffin and J.P. Vial, 1990. Cutting Planes and Column Generation Techniques with the Projective Algorithm, *Journal of Optimization Theory and Applications 65*, 409–429.
23. C.C. Gonzaga, 1989. Conical Projection Algorithms for Linear Programming, *Mathematical Programming 43*, 151–173.

24. C.C. GONZAGA, 1992. Path Following Methods for Linear Programming, *SIAM Review 34:2*, 167–227.
25. P. HUARD, 1970. A Method of Centers by Upper-Bounding Functions with Applications, pp. 1–30 in *Nonlinear Programming: Proceedings of a Symposium held at the University of Wisconsin, Madison, Wisconsin, May 1970*, J.B. Rosen, O.L. Mangasarian and K. Ritter (eds.), Academic Press, New York.
26. INTERNATIONAL BUSINESS MACHINES CORPORATION, 1991. *Optimization Subroutine Library Guide and Reference, Release 2.*
27. N.K. KARMARKAR, 1984. A New Polynomial-Time Algorithm for Linear Programming, *Combinatorica 4*, 373–395.
28. M. KOJIMA, N. MEGIDDO and S. MIZUNO, 1991. A Primal-Dual Infeasible Interior Point Algorithm for Linear Programming, Research Report RJ 8500, IBM Almaden Research Center, San Jose, CA.
29. M. KOJIMA, S. MIZUNO and A. YOSHISE, 1989. A Primal-Dual Interior Point Algorithm for Linear Programming, in *Progress in Mathematical Programming: Interior Point and Related Methods*, N. Megiddo (ed.), Springer Verlag, New York, pp. 29–47.
30. E. KRANICH, 1991. Interior Point Methods for Mathematical Programming: A Bibliography, Discussion Paper 171, Institute of Economy and Operations Research, Fern Universität Hagen, P.O. Box 940, D-5800 Hagen 1, West Germany.
31. J. LIU, 1985. Modification of the Minimum-Degree Algorithm by Multiple Elimination, *ACM Transactions on Mathematical Software 11*, 141–153.
32. I.J. LUSTIG, 1990/91. Feasibility Issues in a Primal-Dual Interior Point Method for Linear Programming, *Mathematical Programming 49*, 145–162.
33. I.J. LUSTIG, R.E. MARSTEN and D.F. SHANNO, 1990. The Primal-Dual Interior Point Method on the Cray Supercomputer, pp. 70–80 in *Large-Scale Numerical Optimization, Papers from the Workshop held at Cornell University, Ithaca, NY, October 1989*, T.F. Coleman and Y. Li (eds.), vol. 46 of *SIAM Proceedings in Applied Mathematics*, Society of Industrial and Applied Mathematics, (SIAM), Philadelphia, PA.
34. I.J. LUSTIG, R.E. MARSTEN and D.F. SHANNO, 1991. Computational Experience with a Primal-Dual Interior Point Method for Linear Programming, *Linear Algebra and its Applications 152*, 191–222.
35. I.J. LUSTIG, R.E. MARSTEN and D.F. SHANNO, 1991. Interior Method vs. Simplex Method: Beyond NETLIB, *COAL Newsletter 19*, 41–44.
36. I.J. LUSTIG, R.E. MARSTEN and D.F. SHANNO, 1992. Computational Experience with a Globally Convergent Primal-Dual Predictor-Corrector Algorithm for Linear Programming, Technical Report SOR 92-10, School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ.
37. I.J. LUSTIG, R.E. MARSTEN and D.F. SHANNO, 1992. The Interaction of Algorithms and Architectures for Interior Point Methods, pp. 190–205 in *Advances in Optimization and Parallel Computing*, P.M. Pardolos (ed.), North-Holland, Amsterdam, The Netherlands.
38. I.J. LUSTIG, R.E. MARSTEN and D.F. SHANNO, 1992. On Implementing Mehrotra's Predictor-Corrector Interior Point Method for Linear Programming, *SIAM Journal on Optimization 2*, 435–449.
39. R.E. MARSTEN, M.J. SALTZMAN, D.F. SHANNO, J.F. BALLINTIJN and G.S. PIERCE, 1989. Implementation of a Dual Affine Interior Point Algorithm for Linear Programming, *ORSA Journal on Computing 1*, 287–297.
40. K.A. MCSHANE, C.L. MONMA and D.F. SHANNO, 1989. An Implementation of a Primal-Dual Interior Point Method for Linear Programming, *ORSA Journal on Computing 1*, 70–83.
41. N. MEGIDDO, 1989. Pathways to the Optimal Set in Linear Programming, pp. 131–138 in *Progress in Mathematical Programming: Interior Point and Related Methods*, N. Megiddo (ed.), Springer Verlag, NY.
42. N. MEGIDDO, 1991. On Finding Primal- and Dual-Optimal Bases, *ORSA Journal on Computing 3*, 63–65.
43. S. MEHROTRA, 1992. On the Implementation of a Primal-Dual Interior Point Method, *SIAM Journal on Optimization 2:4*, 575–601.
44. S. MIZUNO, 1992. Polynomiality of the Kojima-Megiddo-Mizuno Infeasible Interior Point Algorithm for Linear Programming, Technical Report 1006, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY.
45. R.D.C. MONTEIRO and I. ADLER, 1989. Interior Path Following Primal-Dual Algorithms: Part I: Linear Programming, *Mathematical Programming 44*, 27–41.
46. M.J.D. POWELL, 1991. On the Number of Iterations of Karmarkar's Algorithm for Linear Programming, Technical Report DAMTP 1991/NA23, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK.
47. J. RENEGAR, 1988. A Polynomial-Time Algorithm, Based on Newton's Method, for Linear Programming, *Mathematical Programming 40*, 59–93.
48. D.F. SHANNO and A. BAGCHI, 1990. A Unified View of Interior Point Methods for Linear Programming, *Annals of Operations Research 22*, 55–70.
49. G. SONNEVEND, 1986. An "Analytic Center" for Polyhedrons and New Classes of Global Algorithms for Linear (smooth, convex) Programming, pp. 866–876, in *System Modelling and Optimization: Proceedings of the 12th IFIP Conference held in Budapest, Hungary, September 1985*, A. Prekopa, J. Szelezsan and B. Strazicky, (eds.), vol. 84 of *Lecture Notes in Control and Information Sciences*. Springer Verlag, Berlin, West Germany.
50. M.J. TODD and B.P. BURRELL, 1986. An Extension of Karmarkar's Algorithm for Linear Programming using Dual Variables, *Algorithmica 1:4*, 409–424.
51. J.A. TOMLIN, 1987. An Experimental Approach to Karmarkar's Projective Method for Linear Programming, *Mathematical Programming Study 31*, 175–191.
52. R.J. VANDERBEI, 1992. LOQO User's Manual, Technical Report SOR 92-5, School of Engineering and Applied Science, Department of Civil Engineering and Operations Research, Princeton University, Princeton, NJ.
53. R.J. VANDERBEI, M.S. MEKETON and B.A. FREEDMAN, 1986. A Modification of Karmarkar's Linear Programming Algorithm, *Algorithmica 1:4*, 395–407.
54. H. YAMASHITA, 1986. A Polynomially and Quadratically Convergent Method for Linear Programming, Working Paper, Mathematical Systems Institute, Inc., Tokyo, Japan.