Applied and
Computational
Mathematics
Division

NISTIR 89–4225

Center for Computing and Applied Mathematics

Optimal 3-Dimensional Methods for Linear Programming

Paul D. Domich, Paul T. Boggs,
Janet R. Donaldson and Christoph Witzgall

December, 1989

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Gaithersburg, MD 20899

Optimal 3-Dimensional Methods for Linear Programming [*]

Paul D. Domich [†‡] Paul T. Boggs [†§] Janet R. Donaldson [†¶]
Christoph Witzgall [†‖]

December 21, 1989

Abstract

Interior point algorithms for solving linear programming problems are considered. An *optimal 3-dimensional method* is developed that, at each iteration, solves a subproblem based on minimizing the cost function on low-dimensional cross sections of the feasible region. This idea was used by the authors in [Boggs *et al.*, Algorithmic Enhancements to the Method of Centers for Linear Programming Problems, *ORSA Journal of Computing*, 1(3):159-171, 1989.] The generators for the original 2-dimensional subproblems are derived from either a discrete or a continuous version of Huard's method of centers. The generators for the optimal 3-dimensional subproblem include the dual affine search direction, and two higher-order search directions. One of the higher order directions is a third order correction to the Newton recentering direction, and the other is a correction to the dual affine direction that is motivated by the use of rank-one updates of the second derivative information. Numerical results are presented for the optimal 3-dimensional method that indicate almost a 18.9% reduction in CPU time compared to our best dual affine implementation.

Keywords: Method of centers, dual affine direction, ordinary differential equations, low-dimensional subproblem, optimal multi-directional methods, factorable functions, third-order correction terms, recentering, center trajectory.

[*]Contribution of the National Institute of Standards and Technology (N.I.S.T.) and not subject to copyright in the United States. This research was supported in part by ONR Contract N-0014-87-F0053.

[†]Applied and Computational Mathematics Division, National Institute of Standards and Technology

[‡]Boulder, CO 80303-3328 (INTERNET: domich@cam.nist.gov)

[§]Gaithersburg, MD 20899 (INTERNET: boggs@cam.nist.gov)

[¶]Boulder, CO 80303-3328 (INTERNET: jrd@cam.nist.gov)

[‖]Gaithersburg, MD 20899 (INTERNET: witzgall@cam.nist.gov)

1. Introduction

In our previous work on solving linear programming (LP) problems [BogDDW89], we suggested an interior point strategy that consists of exactly solving a sequence of 2-dimensional subproblems. These 2-dimensional subproblems are generated by considering the restriction of the original LP to two search directions. In our best 2-dimensional algorithm, one of the directions is the dual affine direction and the second direction is an approximate recentering direction. In this paper, we extend this idea to include a third, higher-order direction derived from Huard's method of centers [Hua67]. Our numerical results demonstrate the effectiveness of using such higher-order information: the number of iterations required to solve a subset of the Netlib problems [Gay85] (those problems without explicit bounds) is reduced by 33.4% and the time by 18.9% over our new version of the dual affine method. Our results are competitive with the dual affine procedures reported in [Meh89], [MonM87] and [AdlRV86], and with the primal-dual interior point methods reported in [Sha85], [McSMS88] and [LusMS89].

Our techniques, which we call *optimal multi-dimensional methods*, effectively solve the problem of how to combine the various search directions that arise from interior point methods. The theoretical and algorithmic details of our optimal 3-dimensional methods are contained in §2, and the implementation details are given in §3. The computational results for the subset of the Netlib test problems [Gay85] are presented in §4.

In the remainder of this section, we motivate our optimal 2-dimensional method using Huard's original method of centers and a continuous version of it. The various solution "trajectories" discussed below are covered in greater detail in [BayL86] and [WitBD88b]. The corresponding numerical procedures evolving from them are discussed in [BogDDW89].

The LP problem that we solve is of the form

$$\begin{aligned} &\min_u c^{\mathrm{T}} u \\ &\text{s.t. } Au \le b \end{aligned} \tag{1.1}$$

where $c, u \in \mathbf{R}^n$, $A \in \mathbf{R}^{m\times n}$, and $b \in \mathbf{R}^m$. We define the set of residuals related to the constraints of (1.1) as

$$r_k(u) = b_k - A_k u, \qquad k = 1, \dots, m,$$

where A_k denotes the kth row of A, and let $r_0(u,t) = t - c^{\mathrm{T}}u$ correspond to the residual of the objective row assuming an upper bound t on the objective function value. In particular, let u_0 be a feasible point. Then for $t_0 = c^{\mathrm{T}}u_0 + \epsilon,\ \epsilon > 0$,

$$r_0(u_0, t_0) > 0 \qquad \text{and} \qquad r_k(u_0) > 0,\ k = 1, \dots, m.$$

The *center*, defined by the constraints of (1.1) and the objective constraint for $t = t_0$, is the feasible point u that solves

$$F(t_0) = \max_u [L(u, t_0)] \tag{1.2}$$

where

$$L(u, t_0) = \log r_0(u, t_0) + \sum_{k=1}^{m} \log r_k(u). \tag{1.3}$$

The optimization problem (1.2) can be solved using Newton's method. The search direction is then given by

$$\begin{aligned} s_n &= [\nabla_{uu} L(u,t)]^{-1} \nabla_u L(u,t) \\ &= \left[A^{\mathrm{T}} D^2 A + \tfrac{cc^{\mathrm{T}}}{\epsilon^2}\right]^{-1} \left(A^{\mathrm{T}} R + \tfrac{c}{\epsilon}\right), \end{aligned} \tag{1.4}$$

where $D = \operatorname{diag}\left[\frac{1}{r_1}, \frac{1}{r_2}, \ldots, \frac{1}{r_m}\right]$, $R = (\frac{1}{r_1}, \frac{1}{r_2}, \ldots, \frac{1}{r_m})^{\mathrm{T}}$, and $\epsilon = t - c^{\mathrm{T}} u > 0$. Applying the Sherman-Morrison formula to (1.4) results in

$$s_n = \beta_1(\epsilon) \left[A^{\mathrm{T}} D^2 A\right]^{-1} c + \beta_2(\epsilon) \left[A^{\mathrm{T}} D^2 A\right]^{-1} A^{\mathrm{T}} R \tag{1.5}$$

for real-valued functions $\beta_1(\epsilon)$ and $\beta_2(\epsilon)$. The first term in (1.5) is the dual affine search direction [AdlRV86] and the second is the Newton search direction for locating the center of the original polytope defined by $Au \leq b$ without the objective constraint.

Note that one could systematically reduce the value of t and solve a sequence of centering problems (1.2). This yields a sequence of iterates u_i with successively lower objective function values. With a reasonable selection of t_i, it can be shown that $\{u_i\}$ converges to an optimal solution as $i \longrightarrow \infty$. This procedure is Huard's original method of centers [Hua67] applied to the linear programming problem. The implementation of Huard's method proposed by Renegar [Ren86] was shown to possess an equivalent polynomial complexity bound to that of Karmarkar's original method [Kar84].

It is also easily seen that by continuously moving the constraint corresponding to the objective function, one can find a continuous trajectory, rather than a discrete set of points u_k, that converges to an optimal solution of (1.1). In particular, differentiating (1.4) with respect to t results in the ODE

$$u'(t) = -\nabla_{uu} L(u,t)^{-1} \nabla_{ut} L(u,t). \tag{1.6}$$

This ODE can be supplied with initial conditions consisting of an arbitrary feasible point and an appropriate value of t with the result that for every feasible point there is a trajectory that connects it to an optimal solution of (1.1). The direction field at each feasible point can be easily shown to be proportional to the dual affine direction described earlier.

Since the solutions of (1.6) can get arbitrarily close to an exponential number of vertices (see, e.g., [MegS86]), it is of interest to consider modifications that "correct" the trajectories towards the center of the polytope. Thus a second ODE can be formulated [BogDDW89] that includes the Newton recentering component (1.5), i.e.,

$$u'(t) = -\nabla_{uu}L(u,t)^{-1}\left[\nabla_{ut}L(u,t) - \phi\nabla_u L(u,t)\right] \tag{1.7}$$

for $\phi > 0$. If $\nabla_u L(u,t) = 0$, then the recentering component has no effect, and the resulting solution is referred to as the *center trajectory*.

Both (1.5) and (1.7) indicate the motivation for combining the two directions

$$\begin{aligned} s_1 &= \left(A^{\mathrm{T}}D^2A\right)^{-1}c & (1.8)\\ s_2 &= \left(A^{\mathrm{T}}D^2A\right)^{-1}A^{\mathrm{T}}R & (1.9)\end{aligned}$$

into a single search direction

$$s = \varsigma_1 s_1 + \varsigma_2 s_2.$$

The values of the *weights* ς_1 and ς_2 dramatically affect the performance of an interior point procedure. One successful method for determining how to combine such multiple search directions when using large steplengths is the subproblem approach. This approach was suggested by [Kar85] and [Gon87a], and explored more fully in [BogDDW89].

Initially, we considered the 2-dimensional subproblem defined by the span of the two search directions (1.8) and (1.9). Assuming these directions are not colinear, i.e., the current point u is feasible and not on the center trajectory, s_1 and s_2 define a 2-dimensional plane that intersects the full LP polytope. The full LP objective $c^{\mathrm{T}}u$ can then be minimized on this plane using the linear program:

$$\begin{gathered}\min_{\varsigma_1,\varsigma_2} \varsigma_1 c^{\mathrm{T}}s_1 + \varsigma_2 c^{\mathrm{T}}s_2\\ \text{subject to}\\ \varsigma_1 As_1 + \varsigma_2 As_2 \le b - Au\\ \varsigma_1 \ge 0\\ \varsigma_2 \text{ unrestricted.}\end{gathered} \tag{1.10}$$

Note that since u is assumed to be feasible, $\varsigma_1 = 0$ and $\varsigma_2 = 0$ is also feasible. Further, since $A^{\mathrm{T}}D^2A$ is positive definite, the dual affine direction s_1 will always be a descent direction for the objective function; hence, the sign restriction on ς_1. These two conditions imply that a cost-improving feasible solution always exists for the subproblem. Finally, observe that the resulting weights ς_1 and ς_2 are not dependent on the arbitrary values of ϵ and ϕ defined earlier.

2. New Algorithms

2.1. Motivation

The optimal multi-dimensional approach is motivated by noting that the cost function is easily minimized on a low-dimensional polytope, and that the solution to this reduced problem determines a search direction that optimally weights the component search directions over the low-dimensional cross section of the full LP polytope. The new algorithm presented in this section uses a 3-dimensional subproblem. The new search direction included in this subproblem can be derived either from a third-order correction to the Newton recentering direction (1.5), or from a higher-order technique for solving (1.7).

2.2. The Subproblem Generators

The sole restriction on the subproblem generators is that they be linearly independent. In (1.10), the two generators

$$\begin{aligned} s_1 &= \left(A^{\mathrm{T}}D^2A\right)^{-1}c & (2.1)\\ s_2 &= \left(A^{\mathrm{T}}D^2A\right)^{-1}A^{\mathrm{T}}R. & (2.2)\end{aligned}$$

are linearly independent unless the current estimate u lies on the center trajectory, in which case, the dual affine direction (2.1) is the obvious search direction. In [BogDDW89], we discuss other possibilities for s_2, including the first-order (steepest descent) recentering direction

$$s_2 = A^{\mathrm{T}}R.$$

We also derive a correction to the dual affine direction,

$$s_2 = \left(A^{\mathrm{T}}D^2A\right)^{-1}A_k^{\mathrm{T}}, \qquad (2.3)$$

where k is the index of the first constraint encountered in the dual affine direction. This direction is motivated by considering the search direction $s_1 = \left(A^{\mathrm{T}}D^2A\right)^{-1}c$, and the rank-one update of $\left(A^{\mathrm{T}}D^2A\right)^{-1}$ reflecting the change in residual k which produces the "new" search direction $s_1^{new} = \left(A^{\mathrm{T}}D^2A\right)_{new}^{-1}c$. Assuming c is not orthogonal to A_k, it is easily shown that

$$s_1^{new} = \alpha_1 s_1 + \alpha_2\left(A^{\mathrm{T}}D^2A\right)^{-1}A_k^{\mathrm{T}}$$

for scalars α_1 and α_2. Thus, the direction $\left(A^{\mathrm{T}}D^2A\right)^{-1}A_k^{\mathrm{T}}$ represents a rank-one correction to the dual affine direction.

There are several ways to derive the third direction, s_3, for our optimal 3-dimensional method. By recalling the differential equation (1.7), one can expand the solution $u(t)$ in a Taylor series. The directions for the optimal 2-dimensional method can then be viewed as the independent directions that arise from considering the expansion through two terms. By considering the expansion through three terms, the new direction $s_3 = s_h$, described in detail in §2.3 below, is obtained. It is easily shown that this new direction can be found by considering either the expansion based on the dual affine direction or the Newton recentering component. Since the latter is easier to develop, we present that derivation in §2.3. We also show in §2.4 that this direction can be economically calculated by using the "factorable" form of the terms involved.

The set of generators for the subproblem can be varied to create a globally effective algorithm. See §3 for the details of our implementation of this algorithm, and [BogDDW89] for a detailed description and earlier computational results using the 2-dimensional subproblem generators.

2.3. Third Order Correction Terms

The derivation of the term s_h as a correction to the recentering direction follows that described in [JacM86]. To simplify the presentation, we give an informal derivation in one dimension; the extension to higher dimensions is straightforward. Thus, assume that we fix t and that we use a Taylor's series expansion of the scalar function $L(u) \equiv L(u,t)$,

$$L(u+\delta u) = L(u) + L'(u)\delta u + L''(u)\frac{(\delta u)^2}{2!} + L'''(u)\frac{(\delta u)^3}{3!} + \ldots. \tag{2.4}$$

The standard second-order Newton method for maximizing the function $L(u)$ assumes that

$$L'''(u)\frac{(\delta u)^3}{3!} \approx 0,$$

resulting in

$$s_n = [L''(u)]^{-1} L'(u).$$

This direction s_n thus maximizes (2.4) through three terms. Instead, suppose that a significant third-order correction term exists, say s_h. Then for $\delta u = s_n + s_h$, the recentering optimality condition requires

$$L'(u+\delta u) = 0,$$

as before. Ignoring terms of order $O(\|\delta u\|^4)$ in (2.4), we have

$$\begin{aligned} 0 &= L'(u) + L''(u)(s_n + s_h) + L'''(u)\frac{(s_n+s_h)^2}{2!} + O(\|\delta u\|^3) \\ &= L'(u) + L''(u)(s_n + s_h) + L'''(u)\frac{(s_n^2 + 2s_n s_h + s_h^2)}{2!} + O(\|\delta u\|^3) \end{aligned}$$

Note that s_n is $O(\|L'(u)\|)$. Assuming s_h is $O(\|L'(u)\|^2)$, then $s_n s_h$ is $O(\|L'(u)\|^3)$ and s_h^2 is $O(\|L'(u)\|^4)$. Ignoring $O(\|L'(u)\|^3)$ terms, we have

$$\begin{aligned} 0 &= L'(u) + L''(u)(s_n + s_h) + L'''(u)\frac{(s_n^2)}{2!} \\ &= L'(u) - L'(u) + L''(u)s_h + L'''(u)\frac{(s_n^2)}{2!}. \end{aligned}$$

Hence, the third-order correction term is

$$s_h = -\frac{1}{2}[L''(u)]^{-1}L'''(u)s_n^2. \tag{2.5}$$

2.4. The Fractorable Form of $L'''(u)s_n^2$

The actual computation of s_h is considerably simplified using the outer product form of L'''. With the notation from [McCS88], we consider

$$g(x) = U(\alpha(x)),$$

for $U(\alpha)$ possessing as many continuous derivatives as required and $\alpha = a^{\mathrm{T}}x$. Then

$$\begin{aligned} \nabla g(x)^{\mathrm{T}} s &= [\delta U(\alpha(x))/\delta\alpha]a^{\mathrm{T}}s \\ \nabla^2 g(x) s &= a[\delta^2 U(\alpha(x))/\delta\alpha^2](a^{\mathrm{T}}s) \\ \nabla^3 g(x) \otimes s^2 &= a[\delta^3 U(\alpha(x))/\delta\alpha^3](a^{\mathrm{T}}s)^2. \end{aligned}$$

Let $U(\alpha_k(u)) = \log(b_k - A_k u)$. Then the third-order correction term (2.5) can be written as

$$s_h = \frac{1}{2}[L''(u)]^{-1}\sum_{k=1}^{m} A_k^{\mathrm{T}}\left[\frac{1}{(b_k - A_k u)^3}\right](A_k s_n)^2. \tag{2.6}$$

The third-order correction term, s_h, thus requires no increase in the order of work per iteration since the Cholesky factorization of $L''(u)$ has been previously computed and since the summation in (2.6) is $O(nm)$ operations.

2.5. Solving the Subproblem.

The subproblem solution is found using a specialized revised dual simplex approach applied to the dual formulation of the subproblem. This procedure is used to minimize the dimension of the basis matrix and to provide feasible primal solutions to the original subproblem. The procedure also allows the user to terminate the subproblem solution process after a fixed number of iterations with a feasible set of weights for the search direction.

The actual primal subproblem consists of five columns and m rows, ($m \gg 5$). The five columns correspond to the positive dual affine direction (2.1), the third-order correction direction (2.5), and either the recentering direction (2.2) or the rank-one update direction (2.3) described earlier. The second and third columns are replicated and negated in the subproblem to allow for positive and negative multipliers. Since the zero vector is feasible to the subproblem, the primal problem is always initially feasible.

The revised dual simplex approach is applied to the dual formulation of the subproblem. Let S denote the $n \times 5$ matrix composed of search directions $s_1, \ldots, s_5$. Then dual subproblem formulation is

$$\begin{gathered} \min_y r^{\mathrm{T}} y \\ \text{subject to} \\ -(AS)^{\mathrm{T}} y \leq S^{\mathrm{T}} c \\ y \geq 0, \end{gathered} \tag{2.7}$$

where the ith constraint of (2.7) is $(-As_i)^{\mathrm{T}} y \leq s_i^{\mathrm{T}} c$, for $i = 1, \ldots, 5$. This dual LP is dual feasible since $r_j > 0$, for all j.

The procedure begins by forming the LU factorization of the 5×5 basis matrix B (initially $B = I$) and finding the updated solution vector $B^{-1} S^{\mathrm{T}} c$. Next, the dual variables, $\underline{\beta}^{\mathrm{T}} = r_B B^{-1}$, are computed where r_B is the row vector of objective coefficients corresponding to columns in the current basis. The pivot row selection rule finds the row with the largest index k such that $B_k^{-1} S^{\mathrm{T}} c < 0$, corresponding to an infeasible primal variable, where the ith row of B^{-1} is denoted as B_i^{-1}. Because $m \gg 5$, the kth row of B^{-1} is computed explicitly and is used to update the m entries in row k. Computing B_k^{-1} requires only three backsolves using the LU factorization of B since two columns of B^{-1} are known to be unit vectors.

The updated kth row of $(-AS)^{\mathrm{T}}$, $B_k^{-1}(-AS)^{\mathrm{T}}$, is computed and column j is selected that minimizes

$$\frac{\underline{\beta}^{\mathrm{T}}(-AS)_i^{\mathrm{T}} - r_i}{B_k^{-1}(-AS)_i^{\mathrm{T}}}$$

over i such that $B_k^{-1}(-AS)_i^{\mathrm{T}} < 0$, where $(-AS)_i^{\mathrm{T}}$ is the ith column of $(-AS)^{\mathrm{T}}$. This column selection rule guarantees dual feasibility of $\underline{\beta}$ and hence provides feasible search direction multipliers at each step. The pivot is performed and the basis is updated to include column j. This process is repeated until $B^{-1} S^{\mathrm{T}} c \geq 0$. For complete details of the dual simplex approach described above see Bazaraa and Jarvis [BazJ77].

Empirical evidence suggests the subproblem polytopes often have a large number of redundant constraints. Using the specialized procedure described above, the optimal vertex is usually found after a very small number of simplex pivots. (For the test problems reported in §4, all subproblem solutions required fewer than 50 pivots, and more than 95% of the total number of subproblem solutions required fewer than 10 pivots.) An inordinate number of pivots could occur, however, due to the redundant constraints at a near-optimal vertex. Suboptimal solutions, e.g., those obtained after a fixed number of simplex pivots, may be substituted for the optimal solution to save computation time. For the test problems reported in §4, however, the subproblem solution time is only 10% of the total optimal 3-dimensional solution time; terminating the subproblem prematurely causes an increase in the number of major iterations resulting in a overall increase in CPU time.

3. Implementational Details.

Optimal 3-Dimensional Method. In the results reported here, we use the following basic algorithm.

Algorithm 3.1. *Optimal 3-Dimensional Method*

```
1: Compute $s_1 = \left(A^T D^2 A\right)^{-1} c$ at $u_i$.
2: Compute either
    2.1: the rank-one update direction by
        a: finding the index $k$ of the first constraint encountered in the direction $s_1$ and
        b: computing $s_2 = \left(A^T D^2 A\right)^{-1} A_k$, or
    2.2: the recentering direction $s_2 = \left(A^T D^2 A\right)^{-1} A^T R$.
3: Compute the third-order correction direction $s_3 = s_h$ as in (2.6).
4: Solve for $\varsigma_1$, $\varsigma_2$ and $\varsigma_3$ in the 3-dimensional subproblem defined by $s_1$, $s_2$ and $s_3$. i.e.,
```

$$\begin{gathered}\min_{\varsigma_1,\varsigma_2,\varsigma_3} \{\varsigma_1 c^T s_1 + \varsigma_2 c^T s_2 + \varsigma_3 c^T s_3\} \\ \text{subject to} \\ \varsigma_1 A s_1 + \varsigma_2 A s_2 + \varsigma_3 A s_3 \le b - Au \cdot \\ \varsigma_1 \ge 0 \\ \varsigma_2, \varsigma_3 \text{ unrestricted.}\end{gathered} \tag{3.1}$$

```
5: Compute $u_{i+1} = u_i + (0.99)[\varsigma_1 s_1 + \varsigma_2 s_2 + \varsigma_3 s_3]$.
```

Problem Scaling. Our scaling algorithm for A uses techniques discussed in detail in [GilMW81, p.353]. First, each row of A is scaled by the geometric mean of the minimum and maximum absolute values of the nonzero elements of the row. The columns of A are next scaled in the analogous manner. This scaling of the rows and columns is then repeated until the greatest ratio of two nonzero elements in the same column does not change by more than 10%. Finally, each row is scaled by its maximum magnitude, followed by a scaling of each column by its maximum magnitude. Although this scaling of A requires multiple passes through the data structures, the actual cost is minimal as is seen in §4. We find this scaling improves the robustness both of our optimal 3-dimensional method and of our dual affine procedure over that observed using scaling with a fixed number of passes as recently suggested in [MarSSPB88].

The subproblem constraint matrix defined in (3.1) is also scaled. First, each column of the subproblem constraint matrix is scaled by the square of the two-norm of the corresponding subproblem generator s_k, $k = 1, 2, 3$, (see §3). Then each row is scaled by its two-norm. This scaling of the subproblem constraint matrix was also found to improve the numerical stability of the method.

Starting Values and Phase 1 Procedure. Initially, $u_0 = \underline{0}$. We observe that, for the problems examined, this choice of starting value outperforms starting values derived as a function of the problem data. (See, e.g., [AdlRV86], [MarSSPB88], and [Meh89].) Choi *et al.* [ChoMS88] reports similar experience with $u_0 = \underline{0}$. Lustig *et al.* [LusMS89], however, notes that this choice may not be satisfactory for the full expanded Netlib test set.

When necessary, an initial feasible solution is obtained using a big-M Phase 1 procedure (see, e.g., [BazJ77]). We implement this by appending a dense column to the constraint matrix A to produce the Phase 1 problem

$$\min_{u,u_a} c^{\mathrm{T}} u + M u_a$$
$$\text{s.t. } Au - u_a \underline{e} \leq b$$

for $\underline{e} = (1, \ldots, 1)^{\mathrm{T}}$. (The initial choice for u_a is discussed below.) As was true in our earlier work [BogDDW89], the value of M is not dynamically updated at each iteration. Unlike our earlier work, however, the coefficient M of the artificial variable is a function of the scaled values of c, rather than always the value 10^8.

It has been widely reported that interior point methods are sensitive to the selection of M. Numerical difficulities are encountered when M is either too large or too small. For the primal-dual procedures, Choi *et al.* [ChoMS88] uses $\mathrm{M} = \rho n^2 \max\{\|c\|_\infty, \|b\|_\infty\}$, where ρ is either 3 or 30. Similarly, Mehrotra [Meh89] uses $\mathrm{M} = 10^4 \|c\|$. For our work, we set

$$\mathrm{M} = 10 \times \min\{10^7, \|c\|_\infty \times \max\{10^3, \|c\|_\infty\}\}.$$

We found this heuristic more robust than those previously reported. It is arbitrary, but when coupled with our scaling of A, its use significantly improves both our optimal 3-dimensional results, and to a lesser degree, our dual affine results (see §4).

In addition to the selection of M, the big-M Phase 1 procedure requires that we initialize the artificial variable u_a. In our earlier work, we set $u_a = 2\|r\|_\infty$, which, for our choice of u_0 is the same as $u_a = 2\|b\|_\infty$. This is closely related to the value used by McShane *et al.* [McSMS88] of $u_a = 1 + 2\|r\|_\infty$. Mehrotra [Meh89], on the other hand, sets $u_a = 2 \mid \min_{1 \le i \le n}\{r_i\} \mid$. Both values guarantee initial feasibility. To obtain the results reported in §4, we set u_a using both the range of the residuals, and the geometric mean of the maximum and minimum absolute values of the residuals. The results reported in §4 were obtained by conditionally specifying u_a based on the size of the ratio $\mid r_{\min}/r_{\max} \mid$. Specifically, we used

$$u_a = \begin{cases} \mid r_{\min} \mid + \lambda_1 \varphi_1(r) & \text{if } 10 \times \mid r_{\min} \mid < \mid r_{\max} \mid \\ \mid r_{\min} \mid + \lambda_2 \varphi_2(r) & \text{otherwise,} \end{cases}$$

where

$$\begin{aligned} \varphi_1(r) &= (1 + \mid r_{\min} \mid + \mid r_{\max} \mid)^{1/2} \\ \varphi_2(r) &= \max\{1, (r_{\max} - r_{\min})\}, \end{aligned}$$

and where $\lambda_1 = 1$ and $\lambda_2 = .3$. This value of u_a ensures that the starting values u_0 and u_a are safely interior without excessively inflating the original polytope. Note that $\varphi_1(r)$ is the geometric mean of the minimum and maximum residual after adding the minimum amount required to make all residuals greater than or equal to one, and $\varphi_2(r)$ is the range of the residuals.

Problem Preprocessing. The Netlib test problem set [Gay85] has test problems that may contain empty rows or columns, and variables that are explicitly fixed. A small amount of preprocessing is performed on the test problems to remove these extraneous items. The process first identifies fixed variables and removes them from the problem. This is a multiple pass process, since removal of a variable in an earlier iteration may result in identifying additional fixed variables. Once all identified fixed variables are removed, the procedure removes all corresponding empty rows from the problem. (See also [BreMW75].)

Data Structures. The A matrix is stored in sparse format using the XMP experimental mathematical programming data structures described in [Mar81]. The Hessian matrix A^TD^2A is stored using the data structures from the Yale Sparse Matrix Package SMPAK [SCA85].

Hessian GMW Modification. The Hessian is encoded and factored using the Yale Sparse Matrix Package SMPAK [SCA85]. The minimum degree ordering subroutine is invoked at the beginning of Phase 1 and again at the beginning of Phase 2 to find a permutation of the rows and columns that reduces fill-in. The symbolic factorization routine is also invoked only once at the beginning of each Phase. As discussed below, constraints are implicitly dropped as our algorithm progresses. We do not repeat the symbolic factorization as constraints are dropped however.

At each iteration, the Cholesky factorization $U^{\mathrm{T}}EU$ is formed using SMPAK routine SNF. We have altered this code to detect near singularity of the Hessian using the modified Cholesky algorithm specified on page 111 of Gill *et al.* [GilMW81]. In this algorithm, the Cholesky factors U and E are computed subject to two requirements. First, all elements E_{ii} must be strictly positive. Second, the elements $(U^{\mathrm{T}}E^{1/2})$ must satisfy the uniform bound specified in [GilMW81]. As the algorithm is presented in [GilMW81], when these conditions are not met for a row i, the diagonal elements of the original matrix are implicitly increased until the conditions are satisfied. This results in an increase in the value for E_{ii}, while the entries of the corresponding row i of U are left unchanged.

In our implementation, we use the criteria of the modified Cholesky algorithm to identify and remove rows of U that are effectively dependent. Thus, the algorithm was changed so that when a nonzero quantity would be added to the diagonal entry E_{ii}, the corresponding off-diagonal entries in U are zeroed as well. Note that this procedure is slightly more expensive than that of simply changing nonpositive diagonal entries E_{ii} to some small positive number during the factorization. It has the advantage, however, of isolating dependent columns of $A^{\mathrm{T}}DA$ and, thus, obtaining solutions unaffected by the dependent columns.

Constraint Dropping. Constraints that are sufficiently far from the current point u, i.e., those having residuals $r_j(u)$ that satisfy

$$r_j(u) > 10^{12} \times \min\{r_k(u),\ k = 1, \ldots, m\}, \tag{3.2}$$

are explicitly removed from the computations. Constraints j that satisfy (3.2) are "dropped" by setting R_j and D_{jj} to zero prior to computing $A^{\mathrm{T}}D^2A$ and $A^{\mathrm{T}}R$. This can improve the sparsity in $A^{\mathrm{T}}D^2A$ and the numerical accuracy of the resulting search directions, and therefore can lead to improved performance.

Steplength Selection. All procedures reported in §4 use a steplength equal to 99% of the maximum feasible steplength in the designated search direction. This steplength shifts the current estimate u to within 1% of the distance to the boundary of the polytope.

Stopping Criteria. Three standard convergence tests are used to terminate the iterations. These tests are based on (a) the relative change in the objective function, (b) the relative difference between the primal and dual objective values (see, e.g., [AdlRV86]), and (c) the steplength.

Objective function convergence is obtained when

$$\frac{|\, z_i - z_{i-1} \,|}{|\, z_i \,|} < 10^{-8} \tag{3.3}$$

where z_i is the objective function value at iteration i. The convergence criterion based on the relative difference between the primal and dual objective values is

$$\frac{|\, z_i^d - z_i \,|}{\max\{|\, z_i^d \,|, |\, z_i \,|\}} < 10^{-8} \tag{3.4}$$

where z_i^d is the dual objective function value at the current iteration. This, of course, can only be tested when the dual multiplier estimate $D^2A\left(A^{\mathrm{T}}D^2A\right)^{-1}c$ is nonnegative (see, e.g., [MonM87]). The third convergence criterion is based on steplength, where convergence is observed when the steplength is less than or equal to 10^{-16}. All problems in our test set met either (3.3) or (3.4).

Computing Environment. The methods reported here are implemented in Fortran 77 and executed in double precision on a Sun 3/60 using compiler version F77-Sun 1.1 and options -O and -f68881.

4. Computational Results

During this study, several variants of the methods described in §2 were analyzed. In this section, computational results are presented for the optimal 3-dimensional method and for two variants of the dual affine method. Previously, it has been shown in [AdlRV86], [MonM87], and [McSMS88] that the dual affine method compares favorably to MINOS 5.0 [MurS83], a well-known and widely-available implementation of the simplex method. No direct comparison with MINOS is made, however, in this paper.

The methods analyzed in this study were tested on 31 of the 54 publicly available linear programming problems available through Netlib [Gay85]. The problems omitted from our study are those with explicit bounds, which our implementations do not currently handle. The size and optimal objective value for each of these 31 problems are shown in Table 4.1. The number of rows and columns specified for the *Original Problem*, as well as the optimal objective value, are supplied by [Gay85]. These problem dimensions exclude the slack column, the right-hand side vector, and the cost row. The *Reformulated Problem* dimensions indicate the size of the dual problems that we actually solved after preprocessing the data as described in §3.

All but 3 of the 31 problems analyzed require a Phase 1 procedure to obtain an initial feasible point when using the starting values specified in §3. Another 8 problems do not have a full dimensional interior and therefore only require Phase 1 to find the optimal solution. The remaining 20 problems require both Phase 1 and Phase 2.

4.1. Results

The computational results provided here compare our *Optimal 3-D* algorithm with our *Dual Affine* results. Both methods use the same Fortran "kernel" routines. Although only the essential computations are executed for each method, we expect that a further reduction in execution time for both methods could be obtained by customizing each. It may not be meaningful, therefore, to compare directly these times with times obtained using other codes. In addition, we recognize that our timing results are influenced by the order of computations, the compiler and the computing environment. Even the relative timing results reported here may not stay constant if run on a different computer. For a detailed discussion of the difficulties inherent in comparing numerical methods see [JacBNP89].

Table 4.2 summarizes our *Original Dual Affine* results, computed as discussed in [BogDDW89], and our *New Dual Affine* results and our *Optimal 3-D* results, computed as described in §3. For each of these methods, Table 4.2 reports:

Iter.: the number of iterations required for Phase 1 and the total number of iterations, with the two values listed as p/q;

Time: the CPU time, in seconds, excluding data input, preprocessing and scaling as described in §2, and formulation of the dual problem; and

Error: the relative error of the solution, $|(\hat{z} - z^*)/z^*|$, where $\hat{z}$ is the estimated optimal objective value and z^* is the *true* optimal value listed in Table 4.1.

The times for the *Original Dual Affine* results are labeled by Θ, the *New Dual Affine* results by Φ, and the *Optimal 3-D* results by Ψ, for consistancy with Table 4.3. The total number of iterations and the total times required to run all of the problems are listed at the bottom of Table 4.2.

The iteration counts for our *Original Dual Affine* results are nearly identical to the dual affine results reported in [AdlRV86], [MonM87] and [McSMS88]. We have provided the times for this work to allow our new results to be compared to our earlier work and that reported by others.

Table 4.3 provides a comparison of our *Optimal 3-D* approach with our *Dual Affine* results. For convenience, the CPU times for the each of the methods are repeated from Table 4.2. In addition, Table 4.3 reports the change in time between the *Optimal 3-D* approach and the *Dual Affine* approaches, listed both in seconds and as a relative percentage of the *Dual Affine* times, and the ratio of the *Dual Affine* times to the *Optimal 3-D* times.

4.2. Observations

Convergence. Table 4.2 shows that our *Optimal 3-D* results agree well with the accepted optimal values provided in [Gay85]. Each problem in the test set is solved "correctly", converging to the *true* value with at least 6 digits, and in all but 3 cases with 8 or more digits, of agreement.

Iteration Counts. Table 4.2 shows that our *Optimal 3-D* approach results in a significant reduction in the number of iterations when compared to either of our *Dual Affine* results. The number of iterations required by the *Optimal 3-D* method is less than that required by the *Dual Affine* methods for all problems. The relative reductions range from 17.6% to 52.9% for the individual problems, and over the set of 31 problems the total reduction in iterations is 33.4%. Since our dual affine results compare favorably with those reported in [AdlRV86], [MonM87], [McSMS88] and [LusMS89], we believe that our *Optimal 3-D* results are also competative with most current interior point work.

Timings. The *Optimal 3-D* approach results in an overall reduction in CPU time of 18.9% when compared with our *New Dual Affine* results, and 34.9% when compared with our *Original Dual Affine* results. When compared with our *New Dual Affine* results, 22 of the 31 problems show a reduction in CPU time using the *Optimal 3-D* approach. Individual times decrease by as much as 39.2%, and 15 of the problems decrease in time by 10% or more.

Of the problems experiencing an increase in time, the *Optimal 3-D* approach causes a relative increase of more than 10% for problems Scagr7, Share1b, Scsd6 and Scsd8. The largest absolute increase is 11.7 seconds (problem Scsd8), which is 16.5% of its individual execution time (82.8 seconds) and is only 0.2% of the total execution time for all problems (6562 seconds).

It is easily shown that the solution of the subproblem does not increase the *order* of work per iteration. Our timings show that the additional time required to set up and solve the subproblem is only 13.7% (896 seconds) of the total CPU time (6562 seconds) for the *Optimal 3-D* method. Of this, 640 seconds are spent in the simplex solver, and the remaining time is primarily that required to solve for $s_2 = \left(A^{\mathrm{T}} D^2 A\right)^{-1} A_k^{\mathrm{T}}$ or $\left(A^{\mathrm{T}} D^2 A\right)^{-1} A^{\mathrm{T}} R$ and $s_3 = s_h$.

Data input and processing requires a total of about 498 seconds for the 31 problems. Of this, 123 seconds are required for the scaling and preprocessing described in §3. Since the data input and processing time is constant for each of our methods, it is relatively larger for our *Optimal 3-D* method than for our *New Dual Affine* method. Our results would change slightly in favor of our *New Dual Affine* results, therefore, if the data input and processing times were included. The change is not significant, however, and does not alter our conclusions.

4.3. Conclusions

This study demonstrates the computational advantages of using optimal third-order methods as more sophisticated adaptations to the traditional method of centers. In particular, our *Optimal 3-D* results are a significant improvement over our *New Dual Affine* results. The total number of iterations is reduced by 33.4%, and the total CPU time by 18.9%. The iteration counts for our *Optimal 3-D* results are also very competitive with the dual affine and primal-dual results reported in [AdlRV86], [MonM87], [McSMS88], and [LusMS89].

Table 4.1: Test Problem Characteristics

Problem Name	Original Rows	Original Col.	Reformulated Rows	Reformulated Col.	Nonzeros Phase 1 Const.	Nonzeros Phase 1 Hess.	Nonzeros Phase 2 Const.	Nonzeros Phase 2 Hess.	Optimal Value
Afiro	27	32	51	27	153	118	102	90	-.46475314286e3
ADLittle	56	97	137	55	554	433	417	377	.22549496316e6
Scagr7	129	140	184	128	641	753	457	606	-.23313892547e7
Sc205	205	203	315	203	978	858	663	654	-.52202061211e2
Share2b	96	79	162	96	939	968	777	871	-.41573224074e3
Share1b	117	225	248	112	1396	1080	1148	967	-.76589318579e5
Scorpion	388	358	453	375	1913	2367	1460	1991	.18781248227e4
Scagr25	471	500	670	470	2387	2841	1717	2370	-.14753433060e8
ScTap1	300	480	660	300	2532	1987	1872	1686	.14122500000e4
† BrandY	220	249	247	137	2227	2364	—	—	.15185098965e4
‡ Scsd1	77	760	760	77	—	—	2388	1133	.86666666743e1
Israel	174	142	316	174	2759	11402	2443	11227	-.89664482186e6
BandM	305	472	425	258	2459	3370	2034	3111	-.15862801845e3
† Scfxm1	330	457	592	322	3299	3526	—	—	.18416759028e5
† E226	223	282	469	220	3206	3012	—	—	-.18751929066e2
† Scrs8	490	1169	1250	465	4428	2450	—	—	.90429695380e3
† Beaconfd	173	262	262	140	3357	2504	—	—	.33592485807e5
‡ Scsd6	147	1350	1350	147	—	—	4316	2099	.50500000078e2
Ship04s	402	1458	1414	268	5538	3265	4124	2996	.17987147004e7
† Scfxm2	660	914	1184	644	6603	7071	—	—	.36660261564e5
Ship04l	402	2118	2162	356	8530	4933	6368	4576	.17933245379e7
Ship08s	778	2387	2171	416	8477	4969	6306	4552	.19200982105e7
ScTap2	1090	1880	2500	1090	9834	7686	7334	6595	.17248071428e4
† Scfxm3	990	1371	1776	966	9907	10616	—	—	.54901254549e5
Ship12s	1151	2763	2293	466	8849	5126	6556	4659	.14892361344e7
‡ Scsd8	397	2750	2570	397	—	—	8584	4280	.90499999992e3
ScTap3	1480	2480	3340	1480	13074	10347	9734	8866	.14240000000e4
CzProb	929	3523	3141	737	12595	7715	9454	6977	.21851966989e7
† 25FV47	821	1571	1849	793	12415	12509	—	—	.55018458883e4
Ship08l	778	4283	4339	688	17149	9841	12810	9152	.19090552113e7
Ship12l	1151	5427	5329	838	20993	11942	15664	11103	.14701879193e7

† indicates Phase 1 problem

‡ indicates Phase 2 problem

Table 4.2: Results

	Original Dual Affine			New Dual Affine			Optimal 3-D		
Problem	Iter.	Time (sec)	Error	Iter.	Time (sec)	Error	Iter.	Time (sec)	Error
Afiro	1/ 20	1.3	5.E-09	2/ 19	1.3	9.E-09	1/ 10	1.3	4.E-09
ADLittle	1/ 22	5.3	4.E-09	2/ 25	6.1	9.E-10	1/ 15	6.0	3.E-11
Scagr7	3/ 24	8.7	2.E-07	3/ 25	8.9	2.E-07	2/ 18	10.6	2.E-07
Sc205	4/ 27	15.1	3.E-09	6/ 28	16.9	2.E-09	3/ 17	15.7	1.E-10
Share2b	4/ 30	13.6	3.E-09	3/ 27	12.4	3.E-09	2/ 14	9.9	5.E-09
Share1b	7/ 37	27.0	9.E-09	4/ 34	23.3	4.E-09	2/ 28	28.4	5.E-11
Scorpion	5/ 23	37.7	4.E-09	2/ 26	33.2	5.E-09	2/ 17	33.5	2.E-12
Scagr25	3/ 28	50.5	1.E-09	3/ 33	55.7	3.E-10	2/ 21	55.8	1.E-11
ScTap1	6/ 34	55.5	8.E-09	2/ 35	49.5	9.E-09	1/ 21	45.9	9.E-10
† BrandY	37/ 37	108.0	1.E-05	32/ 32	74.6	6.E-09	21/ 21	58.1	1.E-10
‡ Scsd1	0/ 17	19.5	5.E-09	0/ 17	19.2	5.E-09	0/ 8	14.5	2.E-09
Israel	9/ 40	479.9	8.E-09	4/ 31	375.5	1.E-08	3/ 24	307.0	1.E-10
BandM	7/ 30	87.5	8.E-09	7/ 30	74.2	9.E-09	4/ 21	65.3	2.E-10
† Scfxm1	33/ 33	138.6	1.E-09	38/ 38	154.9	1.E-09	23/ 23	113.3	2.E-10
† E226	37/ 37	130.3	3.E-08	32/ 32	109.7	9.E-09	24/ 24	101.0	5.E-07
† Scrs8	52/ 52	450.0	3.E-08	34/ 34	261.7	6.E-09	25/ 25	224.7	2.E-10
† BeaconFD	25/ 25	94.8	6.E-09	25/ 25	73.9	8.E-09	17/ 17	59.3	2.E-09
‡ Scsd6	0/ 19	38.5	5.E-09	0/ 19	38.5	5.E-09	0/ 13	43.0	1.E-09
Ship04s	5/ 29	129.0	3.E-09	2/ 24	83.8	7.E-10	1/ 15	75.1	6.E-09
† Scfxm2	37/ 37	449.8	7.E-09	39/ 39	463.9	6.E-09	29/ 29	387.9	8.E-09
Ship04l	4/ 29	201.8	2.E-09	2/ 22	153.5	2.E-08	1/ 17	152.3	3.E-10
Ship08s	5/ 31	271.0	2.E-09	1/ 24	122.5	8.E-10	1/ 16	132.3	4.E-09
ScTap2	6/ 32	498.4	2.E-09	2/ 27	362.1	5.E-09	1/ 16	285.0	3.E-09
† Scfxm3	38/ 38	906.2	1.E-09	40/ 40	896.3	8.E-09	31/ 31	783.5	3.E-10
Ship12s	5/ 30	363.3	2.E-08	1/ 24	114.0	3.E-09	1/ 16	122.3	2.E-09
‡ Scsd8	0/ 19	73.6	8.E-09	0/ 19	71.1	8.E-09	0/ 13	82.8	2.E-10
ScTap3	6/ 34	766.6	5.E-09	2/ 29	526.6	6.E-09	1/ 18	445.9	4.E-09
CzProb	3/ 44	970.7	1.E-09	2/ 53	933.1	9.E-10	1/ 41	852.0	3.E-11
† 25FV47	55/ 55	2443.2	4.E-08	51/ 51	2095.8	5.E-08	28/ 28	1275.2	4.E-06
Ship08l	4/ 28	494.8	1.E-09	1/ 27	375.0	6.E-10	1/ 17	353.1	3.E-10
Ship12l	5/ 28	755.3	2.E-08	2/ 29	506.5	6.E-10	1/ 17	421.4	1.E-08
All	407/969	10085.5		344/918	8093.7		230/611	6562.1	

† indicates Phase 1 problem

‡ indicates Phase 2 problem

Table 4.3: Comparison

	Optimal 3-D	New Dual Affine				Original Dual Affine			
	Ψ	Φ	$\Psi - \Phi$	$\frac{\Psi-\Phi}{\Phi}$	Φ/Ψ	Θ	$\Psi - \Theta$	$\frac{\Psi-\Theta}{\Theta}$	Θ/Ψ
Problem	(sec)	(sec)	(sec)	(%)		(sec)	(sec)	(%)	
Afiro	1.3	1.3	0.0	0.0	1.00	1.3	0.0	0.0	1.00
ADLittle	6.0	6.1	-0.1	-1.6	1.02	5.3	0.7	13.2	0.88
Scagr7	10.6	8.9	1.7	19.1	0.84	8.7	1.9	21.8	0.82
Sc205	15.7	16.9	-1.2	-7.1	1.08	15.1	0.6	4.0	0.96
Share2b	9.9	12.4	-2.5	-20.2	1.25	13.6	-3.7	-27.2	1.37
Share1b	28.4	23.3	5.1	21.9	0.82	27.0	1.4	5.2	0.95
Scorpion	33.5	33.2	0.3	0.9	0.99	37.7	-4.2	-11.1	1.13
Scagr25	55.8	55.7	0.1	0.2	1.00	50.5	5.3	10.5	0.91
ScTap1	45.9	49.5	-3.6	-7.3	1.08	55.5	-9.6	-17.3	1.21
† BrandY	58.1	74.6	-16.5	-22.1	1.28	108.0	-49.9	-46.2	1.86
‡ Scsd1	14.5	19.2	-4.7	-24.5	1.32	19.5	-5.0	-25.6	1.34
Israel	307.0	375.5	-68.5	-18.2	1.22	479.9	-172.9	-36.0	1.56
BandM	65.3	74.2	-8.9	-12.0	1.14	87.5	-22.2	-25.4	1.34
† Scfxm1	113.3	154.9	-41.6	-26.9	1.37	138.6	-25.3	-18.3	1.22
† E226	101.0	109.7	-8.7	-7.9	1.09	130.3	-29.3	-22.5	1.29
† Scrs8	224.7	261.7	-37.0	-14.1	1.16	450.0	-225.3	-50.1	2.00
† BeaconFD	59.3	73.9	-14.6	-19.8	1.25	94.8	-35.5	-37.4	1.60
‡ Scsd6	43.0	38.5	4.5	11.7	0.90	38.5	4.5	11.7	0.90
Ship04s	75.1	83.8	-8.7	-10.4	1.12	129.0	-53.9	-41.8	1.72
† Scfxm2	387.9	463.9	-76.0	-16.4	1.20	449.8	-61.9	-13.8	1.16
Ship04l	152.3	153.5	-1.2	-0.8	1.01	201.8	-49.5	-24.5	1.33
Ship08s	132.3	122.5	9.8	8.0	0.93	271.0	-138.7	-51.2	2.05
ScTap2	285.0	362.1	-77.1	-21.3	1.27	498.4	-213.4	-42.8	1.75
† Scfxm3	783.5	896.3	-112.8	-12.6	1.14	906.2	-122.7	-13.5	1.16
Ship12s	122.3	114.0	8.3	7.3	0.93	363.3	-241.0	-66.3	2.97
‡ Scsd8	82.8	71.1	11.7	16.5	0.86	73.6	9.2	12.5	0.89
ScTap3	445.9	526.6	-80.7	-15.3	1.18	766.6	-320.7	-41.8	1.72
CzProb	852.0	933.1	-81.1	-8.7	1.10	970.7	-118.7	-12.2	1.14
† 25FV47	1275.2	2095.8	-820.6	-39.2	1.64	2443.2	-1168.0	-47.8	1.92
Ship08l	353.1	375.0	-21.9	-5.8	1.06	494.8	-141.7	-28.6	1.40
Ship12l	421.4	506.5	-85.1	-16.8	1.20	755.3	-333.9	-44.2	1.79
All	6562.1	8093.7	-1531.6	-18.9	1.23	10085.5	-3523.4	-34.9	1.54

† indicates Phase 1 problem

‡ indicates Phase 2 problem

References

[AdlRV86] Ilan Adler, Mauricio G. C. Resende, and Geraldo Veiga. An implementation of Karmarkar's algorithm for linear programming. Manuscript ORC 86-8, Department of Industrial Engineering and Operations Research, University of California, May 1986.

[BayL86] D. A. Bayer and J. C. Lagarias. The nonlinear geometry of linear programming: I. affine and projective scaling trajectories, II. legendre transform coordinates, III. central trajectories, IV. Karmarkar's linear programming algorithm and Newton's method. Internal memoranda, AT&T Bell Laboratories, Murray Hill, NJ, 1986.

[BazJ77] Mokhtar S. Bazaraa and John J. Jarvis. *Linear Programming and Network Flows.* John Wiley and Sons, Inc., New York, 1977.

[BogDDW89] Paul T. Boggs, Paul D. Domich, Janet R. Donaldson, and Christoph Witzgall. Algorithmic enhancements to the method of centers for linear programming programming. *ORSA Journal on Computing*, 1(3):159–171, 1989.

[BreMW75] A. L. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, 8:54–83, 1975.

[ChoMS88] In Chan Choi, Clyde L. Monma, and David F. Shanno. Further development of a primal-dual interior point method. Research Report RRR 60-88, Rutgers University, December 1988.

[Gay85] David M. Gay. Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Newsletter 13, December 1985.

[GilMW81] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization.* Academic Press, Inc., 1981.

[Gon87a] Clovis C. Gonzaga. Search directions for interior linear programming methods. Preliminary version, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, May 1987.

[Hua67] Pierre Huard. Resolution of mathematical programming with nonlinear constraints by the method of centres. In J. Abadie, editor, *Nonlinear Programming*, pages 209–219. North Holland, Amsterdam, 1967.

[JacBNP89] Richard H. F. Jackson, Paul T. Boggs, Stephen G. Nash, and Susan Powell. Report of the ad hoc committee to revise the guidelines for reporting computational experiments in mathematical programming. Draft, January 1989.

[JacM86] Richard H. F. Jackson and Garth P. McCormick. The polyadic structure of factorable function tensors with applications to higher-order minimization techniques. *Journal of Optimization Theory and Applications*, 51(1):63–93, 1986.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[Kar85] Narendra Karmarkar, 1985. ORSA/TIMS Joint National Meeting, Boston MA.

[LusMS89] Irvin J. Lustig, Roy E. Marsten, and David F. Shanno. Computational experience with the primal-dual interior point method for linear programming. Technical Report SOR 89-17, Department of Civil Engineering and Operations Research, Princeton University, School of Industrial Engineering and Operations Research, Georgia Institute of Technology, and Rutgers Center for Operations Research, Rutgers University, October 1989.

[Mar81] Roy E. Marsten. The design of the XMP linear programming library. *ACM Transactions on Mathematical Software*, 7(4):481–497, December 1981.

[MarSSPB88] Roy E. Marsten, Matthew J. Saltzman, David F. Shanno, George S. Pierce, and J. F. Ballintijn. Implementation of a dual affine interior point algorithm for linear programming. Research Report RRR 44-88, Rutgers University, August 1988.

[McCS88] Garth P. McCormick and Ariela Sofer. Optimization with unary functions. The George Washington University and George Mason University, October 1988.

[McSMS88] Kevin A. McShane, Clyde L. Monma, and David F. Shanno. An implementation of a primal-dual interior point method for linear programming. Research Report RRR 24-88, Rutgers University, March 1988.

[MegS86] Nimrod Megiddo and Michael Shub. Boundary behavior of interior point algorithms in linear programming. Manuscript RJ 5319 (54679), IBM Almaden Research Center, Tel Aviv University and IBM T. J. Watson Research Center, September 1986.

[Meh89] Sanjay Mehrotra. Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods. Technical Report 89-04, Northwestern University, August 1989.

[MonM87] Clyde L. Monma and Andrew J. Morton. Computational experience with a dual affine variant of Karmarkar's method for linear programming. Internal memorandum, Bell Communications Research, March 1987.

[MurS83] Bruce A. Murtaugh and Mitchell A. Saunders. MINOS 5.0 user's guide. Technical Report SOL 83-20, Stanford Optimization Laboratory, December 1983.

[Ren86] J. Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. Report 07118-86, Mathematical Sciences Research Institute, University of California at Berkeley, June 1986.

[Sha85] David F. Shanno. Computing Karmarkar projections quickly. Working Paper 85-10, University of California, December 1985.

[SCA85] *SMPAK User's Guide Version 1.0*, 1985.

[WitBD88b] Christoph Witzgall, Paul T. Boggs, and Paul D. Domich. On the convergence behavior of trajectories for linear programming. Submitted to *Proceedings of the AMS-IME-SIAM Research Converence on "Mathematical Developments Arising from Linear Programming Algorithms"*, June 1988.

NBS-114A (REV. 2-80)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** *(See instructions)*	1. PUBLICATION OR REPORT NO. NISTIR 89-4225	2. Performing Organ. Report No.	3. Publication Date DECEMBER 1989

4. TITLE AND SUBTITLE

Optimal 3-Dimensional Methods for Linear Programming

5. AUTHOR(S)

Paul D. Domich, Paul T. Boggs, Janet R. Donaldson, and Christoph Witzgall

6. PERFORMING ORGANIZATION *(If joint or other than NBS, see instructions)*

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)*

Interior point algorithms for solving linear programming problems are considered. An optimal 3-dimensional method is developed that, at each iteration, solves a subproblem based on minimizing the cost function on low-dimensional cross sections of the feasible region. This idea was used by the authors in [Boggs et al., Algorithmic Enhancements to the Method of Centers for Linear Programming Problems, ORSA Journal of Computing, 1(3):159-171, 1989.] The generators for the original 2-dimensional subproblems are derived from either a discrete or a continuous version of Huard's method of centers. The generators for the optimal 3-dimensional subproblem include the dual affine search direction, and two higher-order search directions. One of the higher order directions is a third order correction to the Newton recentering direction, and the other is a correction to the dual affine direction that is motivated by the use of rank-one updates of the second derivative information. Numerical results are presented for the optimal 3-dimensional method that indicate almost a 23% reduction in CPU time compared to our best dual affine implementation.

12. KEY WORDS *(Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)*

method of centers; dual affine direction; ordinary differential equations; low-dimensional subproblem; optimal multi-directional methods; factorable functions; third-order correction terms, recentering, center trajectory

13. AVAILABILITY

[✓] Unlimited
☐ For Official Distribution. Do Not Release to NTIS
☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.
[✓] Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

24

15. Price

A02