POWER SERIES VARIANTS OF KARMARKAR-TYPE ALGORITHMS

Narendra K. Karmarkar, Jeffrey C. Lagarias, Lev Slutsman, and Pyng Wang

Narendra K. Karmarkar *is in the Mathematical Foundations of Computing Department and* ***Jeffrey C. Lagarias*** *is in the Mathematical Studies Department; they are with AT&T Bell Laboratories in Murray Hill, New Jersey.* ***Lev Slutsman*** *and* ***Pyng Wang*** *are in the Advanced Decision Support Systems Department with AT&T Bell Laboratories in Holmdel, New Jersey. Mr. Karmarkar is an AT&T Bell Laboratories Fellow and does research in theoretical computer science and mathematical programming. He joined the company in 1983 and holds a B.S. from the Indian Institute of Technology, Bombay; an M.S. in electrical engineering from the California Institute of Technology; and a Ph.D. in computer science from the University of California at Berkeley. Mr. Lagarias is a member of technical staff and does research in discrete*
(continued on page 36)

Many interior-point linear programming algorithms have been proposed since the Karmarkar algorithm for linear programming problems appeared in 1984. These algorithms follow tangent (first-order) approximations to families of continuous trajectories that underlie such algorithms. This paper describes power-series variants of such algorithms that follow higher-order, truncated, power-series approximations to such trajectories. The choice of the power-series parameter is important to the performance of such algorithms, and this paper describes an apparently good choice of parameter. We describe two power-series algorithms; one builds on the dual-affine scaling algorithm and the other on a primal-dual path-following algorithm. Empirical results indicate that, compared to first-order methods, these higher-order power-series algorithms accelerate convergence by reducing the number of iterations. Both of these power-series algorithms have been successfully implemented in the AT&T KORBX® system.

Introduction

In 1984, N. K. Karmarkar presented[1] a new polynomial-time algorithm for solving linear programs. This algorithm is an interior-point method that takes a series of steps through the (relative) interior of the polytope of feasible solutions of the linear program. The algorithm uses a special interior-feasible solution as a starting point; this point is called the *center* of the polytope and has an analytical characterization.[2] A general linear program can be converted into an equivalent linear program in which such a starting point is available.

Various interior-point algorithms had been proposed since the early 1950s. However, the expensive computational steps they require, the possibility of numerical instability in the calculations, and some discouraging experimental results had led to a consensus view that such algorithms would not be competitive with the simplex method. Karmarkar's paper[1] provided rigorous theoretical justification for the possible good performance of interior-point methods, and this has led to the recent intense study of such algorithms, both theoretically and in practical implementations. There is now good experimental evidence that implementations of some of these interior-point methods outperform the simplex method on a large variety of problems.[3–5]

A wide variety of "Karmarkar-type" algorithms have now been developed. These algorithms may be grouped in two major classes: vector-field algorithms and path-following algorithms.

Vector-field algorithms associate a search direction to each interior-feasible solution, and each iteration of the algorithm takes a step in the search direction from the current feasible point to a new feasible point. The Karmarkar algorithm,[1] affine scaling algorithms,[6–10] and potential function algorithms[11] are vector-field algorithms. To each of these algorithms, one associates a family of trajectories obtained by taking the step size infinitesimally small; these trajectories fill up the interior of the feasible-solution polytope. All these trajectories approach an optimal solution of the linear programming problem.[10,12,13] These algorithms all have one trajectory in common, called the *central trajectory* or *central path* (which is defined later in the section on primal-dual algorithms and in Reference 12).

Path-following algorithms of the Karmarkar-type are algorithms that follow the central trajectory closely by one method or another. Conceptually, one may think of them as approximating at the kth iteration a goal point $\mathbf{x}^{(k)}$ on the central path. From the current iterate $\mathbf{y}^{(k-1)}$, the algorithm attempts to approach the point $\mathbf{x}^{(k)}$. Such path-following algorithms usually have defined at each step an auxiliary vector field whose trajectories converge to the current goal point $\mathbf{x}^{(k)}$, and the algorithms follow tangent (first-order) approximations to these trajectories. The points $\{\mathbf{x}^{(k)}\}$ are indexed by a parameter $\bar{\mu}$, so that $\mathbf{x}^{(k)} = \mathbf{x}(\bar{\mu}^{(k)})$, and the $\mathbf{x}^{(k)}$ converge to an optimal solution as k increases. Various path-following algorithms are discussed in References 14 through 19. Several of these path-following algorithms have some relation to earlier work in nonlinear programming, but the identification of the central path as a particularly good path to follow arose from study of the Karmarkar algorithm.[12]

All these Karmarkar-type algorithms use first-order approximations to the underlying continuous trajectories. A natural idea is to consider algorithms that take higher-order (smooth curve) approximations to the trajectories.[20] This paper describes higher-order approximations to the continuous trajectories consisting of truncated power-series expansions.

The primary benefit that power-series methods offer is the possibility of reducing the number of iterations such algorithms take, compared to first-order methods. When applied to vector-field algorithms, such power-series algorithms may take more accurate steps and reduce the number of iterations. For path-following methods, power-series algorithms may permit the goal points $\mathbf{x}^{(k)}$ to be moved faster along the central path than first-order methods, again reducing the number of iterations.

Practical implementations of Karmarkar-type algorithms use a "large" step size, one that is larger than the "small" step size used in the theoretical analysis of such algorithms. Large-step algorithms are often empirically observed to take fewer iterations to converge than small-step algorithms, although theoretical proof of this is lacking.

For (first-order) vector-field algorithms, a large step means moving a fixed fraction of the way to the boundary of the feasible-solution polytope, independent of the problem's dimension. For (first-order) path-following algorithms, a large step means decreasing the path parameter $\bar{\mu}$ (described later) by at least a fixed multiplicative constant. This paper presents empirical evidence that shows that some decrease in the number of iterations occurs for suitably chosen higher-order, large-step algorithms, compared to similar first-order, large-step algorithms.

Of particular importance to the performance of power-series methods is the choice of power-series parameter. In first-order methods, the search direction is invariant under change of coordinates, but this is no longer true for higher-order methods. Higher-order algorithms depend on the power-series parameter used, and the performance of such methods depends on this choice of parameter. The particular choice of power-series parameter we propose is determined by a heuristic described later.

One can construct power-series analogues of any first-order Karmarkar-type algorithm. This paper describes two such power-series algorithms, one for a vector-field algorithm—the dual-affine scaling algorithm—and one for a primal-dual path-following algorithm. Both algorithms have been implemented in the AT&T KORBX system,[21] where the higher-order approximations reduce the number of iterations compared to first-order methods. It is found that the dual power series of order 5 and primal-dual power series of order 3 reduce the number of iterations by almost 40 percent, when compared with their order-1 algorithms. In terms of computational speed, the power-series algorithms give a CPU (central-processing unit) time for small or very sparse problems that is comparable to the order-1 algorithm. For large or dense problems, the power-series algorithms achieve a significant reduction in CPU time, compared to similar first-order algorithms.

The example of these two algorithms strongly suggests that properly parameterized power-series algorithms can provide some speedup for a wide class of first-order, interior-point, linear-programming methods. The algorithms described here should be viewed as illustrative of this speedup and not as an optimal choice of first-order (or higher-order) methods. In particular, primal-dual algorithms of the vector-field type, such as primal-dual versions of affine scaling and projective scaling algorithms, merit further study.

In the next section, we review the primal-affine scaling, dual-affine scaling, and primal-dual path-following algorithms. Then, we present continuous versions of these algorithms, defined by systems of ordinary differential equations. We also develop power-series expansions to approximate the solution of the differential equations for the dual and primal-dual algorithms. The last section describes the computational performance of these power-series algorithms.

Variants of the Karmarkar Algorithm

Consider a linear programming problem:

$$\begin{aligned} &\text{Minimize} && \mathbf{c}^T\mathbf{x} \\ &\text{subject to} && A\mathbf{x} = \mathbf{b} \\ &\text{for} && \mathbf{x} \geq \mathbf{0} \end{aligned} \qquad \text{(P)}$$

where $\mathbf{c}$ and $\mathbf{x}$ are n-vectors ($\mathbf{c}^T$ is the transpose of $\mathbf{c}$), $\mathbf{b}$ is an m-vector, and A is an $m \times n$ matrix. We assume that the matrix A has full row rank, and the feasible region of the polytope defined by $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ has an interior point, say $\mathbf{x}^0 > \mathbf{0}$.

Starting at $\mathbf{x}^0 > \mathbf{0}$, an interior-point algorithm seeks a search direction $\Delta\mathbf{x}$ such that:

$$\mathbf{x} = \mathbf{x}^0 + \xi\,\Delta\mathbf{x}$$

is strictly interior feasible and $f(\mathbf{x}) < f(\mathbf{x}^0)$ for some real parameter $\xi > 0$, where f is a goal function that measures progress of the algorithm, such as the Karmarkar potential function.[1]

One of the main ideas in Karmarkar's work[1] is the use of a rescaling of the linear programming problem(P) for selecting a search direction. Through the use of coordinate transformations, the rescaling moves the starting point $\mathbf{x}^0$ closer to the center of the transformed polytope to allow a larger step size and, hence, increased reduction of the objective function during each step. In the rest of this section, we review the primal-affine scaling algorithm, the dual-affine scaling algorithm, and then a path-following algorithm.

Primal-Affine Scaling Algorithm. This algorithm applies a series of affine transformations to the problem (P). Consider the diagonal matrix $D = diag(x_1^0, x_2^0, \cdots, x_n^0)$ and the affine transformation defined by $\mathbf{y} = D^{-1}\mathbf{x}$. With respect to the $\mathbf{y}$-coordinate system, the problem (P) has the matrix AD and the cost vector $D\mathbf{c}$. In this transformed space, the search direction $\Delta\mathbf{y}$ is chosen to be the orthogonal projection of $-D\mathbf{c}$ onto the null space of AD ($-D\mathbf{c}$ is the negative gradient of the transformed objective function). That is,

$$\Delta\mathbf{y} = -P_{AD} D\mathbf{c}$$

where P_{AD} is the projection to the null space of AD and is given by:

$$P_{AD} = I - (AD)^T (AD^2 A^T)^{-1} (AD)$$

where I denotes the identify matrix. By transforming $\Delta\mathbf{y}$ back to the original space, we have:

$$\Delta\mathbf{x} = -DP_{AD} D\mathbf{c} = -D^2(\mathbf{c} - A^T\mathbf{w})$$

with

$$\mathbf{w} = (AD^2 A^T)^{-1} AD^2\mathbf{c}$$

For a more detailed description of this algorithm, see Reference 9.

Dual-Affine Scaling Algorithm. The dual of the primal problem (P) can be expressed in the following form:

$$\begin{aligned} &\text{Maximize} && \mathbf{b}^T\mathbf{w} \\ &\text{subject to} && A^T\mathbf{w} + \mathbf{s} = \mathbf{c} \\ &\text{for} && \mathbf{s} \geq \mathbf{0} \end{aligned} \tag{D}$$

We assume that (D) has a feasible point $(\mathbf{w}^0, \mathbf{s}^0)$ with $\mathbf{s}^0 > \mathbf{0}$. The dual-affine scaling algorithm seeks a direction $(\Delta\mathbf{w}, \Delta\mathbf{s})$ such that:

$$\begin{aligned} \mathbf{w} &= \mathbf{w}^0 + \xi\Delta\mathbf{w} \\ \mathbf{s} &= \mathbf{s}^0 + \xi\Delta\mathbf{s} \end{aligned} \tag{1}$$

is strictly interior feasible (i.e., $A^T\mathbf{w} + \mathbf{s} = \mathbf{c}$ for $\mathbf{s} > \mathbf{0}$) and $\mathbf{b}^T\mathbf{w} > \mathbf{b}^T\mathbf{w}^0$ for some $\xi > 0$.

By applying the affine transformation $\mathbf{u} = D\mathbf{s}$ with $D = diag(1/s_1^0, \cdots, 1/s_n^0)$, it was shown in References 3 and 4 that the dual-affine direction $(\Delta\mathbf{w}, \Delta\mathbf{s})$ is given by:

$$\begin{aligned} \Delta\mathbf{w} &= (AD^2 A^T)^{-1}\mathbf{b} \\ \Delta\mathbf{s} &= -A^T\Delta\mathbf{w} \end{aligned} \tag{2}$$

It was also shown in Reference 22 that the dual-affine scaling algorithm is exactly the primal-affine scaling algorithm applied to the dual problem.

Primal-Dual Path-Following Algorithm. The family of algorithms that we describe for following the central path of the primal-dual problem[15,16] includes the primal-dual affine scaling algorithm as a limiting case. These algorithms simultaneously solve the primal problem (P) and its dual (D) described in the previous sections. We follow the notation of Reference 15. Intuitively, the central path is a curve that goes through the "center of the polytope" that is far from the coordinate walls. Analytically, this path can be derived using the logarithmic barrier programs (P_μ) given by:

$$\begin{aligned} &\text{Minimize} && \mathbf{c}^T\mathbf{x} - \mu\sum_{i=1}^{n}\log x_i \\ &\text{subject to} && A\mathbf{x} = \mathbf{b} \\ &\text{for} && \mathbf{x} > \mathbf{0} \end{aligned} \tag{P_μ}$$

Here, $\mu > 0$ is the barrier parameter.

Given the assumptions stated earlier for the primal (P) and dual (D) problems, the problem (P_μ) has a unique optimal solution[16] that is defined by the Kuhn-Tucker stationary conditions:

$$\begin{aligned} &A\mathbf{x} = \mathbf{b} \\ &A^T\mathbf{w} + \mathbf{s} = \mathbf{c} \\ &x_i s_i = \mu, \quad \text{for } i = 1, 2, \cdots, n \\ &\mathbf{x} > \mathbf{0} \end{aligned} \tag{3}$$

Because $\mathbf{x}$ and $(\mathbf{w}, \mathbf{s})$ defined by system (3) are feasible for problems (P) and (D), it is easy to verify that the duality gap $g = g(\mathbf{x}, \mathbf{w})$ is given by:

$$g = \mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{w} = \mathbf{x}^T\mathbf{s} = n\mu \tag{4}$$

Let $\mathbf{y}(\mu) = (\mathbf{x}(\mu), \mathbf{w}(\mu), \mathbf{s}(\mu))$ denote the solution of equation (3). As μ varies, $\mathbf{y}(\mu)$ traces a curve in $\mathbf{R}^{2n+m}$ that consists of the solutions of system (3). We will call this curve the *central path.* It follows from equation (4) that, when $\mu \to 0$, the central path leads to an optimal solution $(\mathbf{x}*, (\mathbf{w}*, \mathbf{s}*))$ of the programs (P) and (D), and $\mathbf{y}(0)$ is this optimal solution. The primal-dual algorithm that we now describe is based on the idea of almost following the central path $\mathbf{y}(\mu)$.

Let us assume that an interior primal-dual feasible solution $\mathbf{y}^0 = (\mathbf{x}^0, \mathbf{w}^0, \mathbf{s}^0)$ of the programs (P) and (D) is known. This means:

$$\begin{aligned} &A\mathbf{x}^0 = \mathbf{b} \\ &A^T\mathbf{w}^0 + \mathbf{s}^0 = \mathbf{c} \\ &x_i^0 s_i^0 = \mu(i), \quad \text{for } i = 1,2, \cdots, n \end{aligned}$$

To follow the central path, we would like to move from the starting point $\mathbf{y}^0$ in the direction of some point $\mathbf{y}(\bar{\mu})$ on the central path. To reduce the current duality gap $g^0 = \mathbf{c}^T\mathbf{x}^0 - \mathbf{b}^T\mathbf{w}^0$, we choose:

$$\bar{\mu} \equiv \sigma \frac{\mathbf{c}^T\mathbf{x}^0 - \mathbf{b}^T\mathbf{w}^0}{n} = \sigma \frac{g^0}{n} \tag{5}$$

where σ is a parameter, with $0 < \sigma < 1$, to be chosen as described below. From equation (4) and the fact that $\mathbf{y}(\bar{\mu})$ satisfies equation (3) with $\mu = \bar{\mu}$, it follows that the duality gap at $\mathbf{y}(\bar{\mu})$ is equal to σg^0, which is less than g^0. We will use one step of Newton's method to solve system (3) approximately.

The *Newton search direction* $(\Delta\mathbf{x}, \Delta\mathbf{w}, \Delta\mathbf{s})$ for equation (3) at $(\mathbf{x}^0, \mathbf{w}^0, \mathbf{s}^0)$ may be written as:

$$\begin{aligned} &A\Delta\mathbf{x} = 0 \\ &A^T\Delta\mathbf{w} + \Delta\mathbf{s} = 0 \\ &S^0\Delta\mathbf{x} + X^0\Delta\mathbf{s} = \bar{\mu}\mathbf{e} - X^0S^0\mathbf{e} \end{aligned} \tag{6}$$

Here, $\mathbf{e} = (1, 1, \cdots, 1)$, and $S^0 = diag(s_i^0)$ and $X^0 = diag(x_i^0)$ are diagonal matrices. Define:

$$\mathbf{v}(\bar{\mu}) = \bar{\mu}\mathbf{e} - X^0S^0\mathbf{e}$$

By simple calculations, the linear system (6) may be explicitly solved as follows:

$$\begin{aligned} &\Delta\mathbf{w} = -(AD^2A^T)^{-1}AS^{-1}\mathbf{v}(\mu) \\ &\Delta\mathbf{s} = -A^T\Delta\mathbf{w} \\ &\Delta\mathbf{x} = S^{-1}v(\mu) - D^2\Delta\mathbf{s} \end{aligned} \tag{7}$$

where $D^2 = S^{-1}X$, on choosing $X = X^0$, $S = S^0$, and $\mu = \bar{\mu}$. For later use, we write equation (6) for a general feasible point $(\mathbf{x}, \mathbf{w}, \mathbf{s})$ for problems (P) and (D) and with an arbitrary barrier-parameter value $\mu > 0$. We remark that the *primal-dual affine scaling direction* arises from taking

$\mu = 0$ in equation (7).

The new primal-dual feasible solution $\mathbf{y}^1 = (\mathbf{x}^1, \mathbf{w}^1, \mathbf{s}^1)$ may be written as:

$$\mathbf{y}^1 = \mathbf{y}^0 + \xi \Delta \mathbf{y} \tag{8}$$

where $\Delta\mathbf{y} = (\Delta\mathbf{x}, \Delta\mathbf{w}, \Delta\mathbf{s})$ and ξ is to be chosen to guarantee the positivity of $\mathbf{x}^1$ and $\mathbf{s}^1$, while maintaining reasonable closeness to the central path.

The idea behind this algorithm is that $\mathbf{y}^1$ will be a reasonable approximation to the point $\mathbf{y}(\bar{\mu})$, with $\bar{\mu}$ defined by equation (5). At the next iteration, the value of $\bar{\mu}$ decreases, using equation (5), and the iterative process is repeated until the duality gap becomes sufficiently small. The step size of the path-following algorithm is controlled by the choice of the parameter σ, which determines how fast the parameter $\bar{\mu}$ is driven to zero. Gonzaga's analysis[19] allows one to show that a choice like $\sigma = 1 - 1/(50\sqrt{n})$ guarantees that the algorithm converges in $O(\sqrt{n}L)$ iterations, where L measures the input size in bits. In practical implementations, one would like to take a bigger step, e.g., to take $\sigma = 1/2$; but in this discussion, these theoretical analyses no longer apply.

The *primal-dual affine scaling algorithm* arises in the limiting case that one takes $\bar{\mu} = 0$ at every step of the algorithm. Monteiro, Adler, and Resende showed[16] that, if the starting point is chosen close to the central path and the step size is chosen properly, then this algorithm can be guaranteed to converge in $O(nL^2)$ iterations.

Power-Series Algorithms

We extend the algorithms previously discussed to continuous versions defined by systems of ordinary differential equations. These equations specify a vector field on the interior of the feasible-solution polytope, and the integral curves of the vector field are called the continuous trajectories of the algorithm. Approximation of the trajectory by the method of power-series expansion was presented in References 20 and 12. Also, in Reference 3, the power-series approach was applied to the dual algorithm, and computational results for the order-2 approximation were reported. That paper used the choice of power-series parameter described below, without motivation.

In the rest of this section, we analyze the system of differential equations with parameterizations and derive the power-series expansion with a particular choice of parameter to approximate the solution of this system of equations. This derivation leads to recursive formulas for both the dual and primal-dual power-series algorithms. Each recursion requires a computational time quadratic in the number of the constraints. Furthermore, the computational complexity of one iteration is proportional to the order of the power-series approximation.

Dual Power-Series Algorithm. In equation (1), as the step size ξ is taken infinitesimally small, it is seen that the iteration points trace a smooth curve and that the curve is governed by a system of differential equations derived from system (2). More precisely, it follows from equation (1) that $\Delta\mathbf{w} \to d\mathbf{w}/d\xi$ and $\Delta\mathbf{s} \to d\mathbf{s}/d\xi$ as $\xi \to 0$. Then, system (2) becomes the following system of ordinary differential equations:

$$\begin{aligned} \frac{d\mathbf{w}}{d\xi} &= [AD^2(\xi)A^T]^{-1}\mathbf{b} \\ \frac{d\mathbf{s}}{d\xi} &= -A^T\frac{d\mathbf{w}}{d\xi} \\ \text{for}\quad \mathbf{w}(0) &= \mathbf{w}^0, \quad \mathbf{s}(0) = \mathbf{s}^0 \end{aligned} \tag{9}$$

where

$$D(\xi) = diag\left(\frac{1}{s_1(\xi)}, \ldots, \frac{1}{s_n(\xi)}\right)$$

The solution $(\mathbf{w}(\xi), \mathbf{s}(\xi))$ of system (9) traces out a continuous trajectory of the dual-affine scaling algorithm; this trajectory has the property that the dual-affine direction $(\Delta\mathbf{w}, \Delta\mathbf{s})$ is the tangent approximation of this family of trajectories.

In practice, it is useful to choose a different parameterization of this trajectory for reasons explained in the next section. Consider a new parameter t defined by $\xi = \phi(t)$, where we suppose $\phi(0) = 0$ and $\xi = \sum_{j\geq 1} \phi^{(j)} t^j$, with $\phi^{(1)} > 0$. For any function $f(\xi)$, the corresponding function in the t-coordinate system is given by the composition of $f(\xi)$ and $\phi(t)$, i.e., $\hat{f}(t) = f(\phi(t))$. To simplify notation, we will write $f(t)$ for $\hat{f}(t)$, and adopt this convention throughout the rest of this paper. Using $d\mathbf{w}/dt = d\mathbf{w}(\phi(t))/dt = (d\mathbf{w}/d\xi)\,(d\xi/dt)$, it is easy to see from equation (9) that the trajectory described by $(\mathbf{w}(t), \mathbf{s}(t))$ in the t-coordinate space satisfies the system of differential equations:

$$\frac{d\mathbf{w}}{dt} = \rho(t)\;[AD^2(t)A^T]^{-1}\mathbf{b}$$

$$\frac{d\mathbf{w}}{dt} = -A^T\,\frac{d\mathbf{w}}{dt}$$

$$\text{for}\quad \mathbf{w}(0) = \mathbf{w}^0,\;\; \mathbf{s}(0) = \mathbf{s}^0$$

$$D(t) = diag\left[\frac{1}{s_1(t)}, \cdots, \frac{1}{s_n(t)}\right] \tag{10}$$

where $\rho(t) = d\xi/dt = d\phi/dt$. Note that knowledge of the function $\rho(t)$ completely determines the change of parameter from ξ to t, because we know that $\phi(0) = 0$.

Now, we derive higher-order power-series expansion formulas to approximate the continuous trajectory $(\mathbf{w}(t), \mathbf{s}(t))$. To simplify notation, we define:

$$z_i(t) = s_i^2(t),\quad \text{for } i = 1,2,\cdots,n$$

$$y_i(t) = s_i^{-2}(t) = \frac{1}{z_i(t)},\quad \text{for } i = 1,2,\cdots,n$$

$$Z(t) = diag(z_1(t), \cdots, z_n(t))$$

$$Y(t) = diag(y_1(t), \cdots, y_n(t)) = D^2(t)$$

Let:

$$M(t) = AD^2(t)A^T = AY(t)A^T$$

System (10) can be rewritten as:

$$M(t)\,\frac{d\mathbf{w}}{dt} = \rho(t)\,\mathbf{b}$$

$$\frac{d\mathbf{s}}{dt} = -A^T\,\frac{d\mathbf{w}}{dt}$$

$$\text{for}\quad \mathbf{w}(0) = \mathbf{w}^0,\;\; \mathbf{s}(0) = \mathbf{s}^0 \tag{11}$$

We are interested in finding a power-series expansion of the solution $(\mathbf{w}(t), \mathbf{s}(t))$, where:

$$\mathbf{w}(t) = \sum_{j=0}^{\infty} \mathbf{w}^{(j)} t^j$$

$$\mathbf{s}(t) = \sum_{j=0}^{\infty} \mathbf{s}^{(j)} t^j$$

Similarly, we introduce expansions for the following diagonal matrix functions:

$$Z(t) = \sum_{j=0}^{\infty} Z^{(j)} t^j$$

$$Y(t) = \sum_{j=0}^{\infty} Y^{(j)} t^j$$

$$M(t) = \sum_{j=0}^{\infty} M^{(j)} t^j$$

and a scalar function:

$$\rho(t) = \sum_{j=0}^{\infty} \rho^{(j)} t^j \tag{12}$$

Here, for any function $f(t)$, we let $f^{(j)} = (1/j!) \times (d^j f/dt^j)\,|_{t=0}$, the jth derivative of f at 0 divided by $j!$.

To obtain approximations of the solution curve, we need to compute $\mathbf{w}^{(k)}$ and $s^{(k)}$ for $k \geq 1$, and to determine $\rho^{(k)}$ for $k \geq 0$. Note that the first derivatives $\mathbf{w}^{(1)}$ and $\mathbf{s}^{(1)}$ are given by equation (10) with $t = 0$, and that they are components of the dual-affine scaling direction of equation (2) with a proper choice of $\rho(0)$. In terms of the notation of system (11), this means:

$$\begin{aligned} \mathbf{w}^{(1)} &= M(0)^{-1}\rho(0)\mathbf{b} \\ \mathbf{s}^{(1)} &= -A^T\mathbf{w}^{(1)} \\ &\text{for} \quad \mathbf{w}(0) = \mathbf{w}^0, \ \mathbf{s}(0) = \mathbf{s}^0 \end{aligned} \tag{13}$$

where $M(0) = AD^2(0)A^T$ and $D(0) = diag(1/s_i(0)) = diag(1/s_i^0)$.

After differentiating equation (11) k times and letting $t = 0$, we have:

$$\mathbf{s}^{(k+1)} = -A^T\mathbf{w}^{(k+1)} \tag{14}$$

and

$$\sum_{j=0}^{k} (j+1)\, M^{(k-j)}\mathbf{w}^{(j+1)} = \rho^{(k)}\mathbf{b}$$

That is,

$$M^{(0)}\mathbf{w}^{(k+1)} = \frac{1}{k+1}\left[\rho^{(k)}\mathbf{b} - \sum_{j=1}^{k} jM^{(k-j+1)}\mathbf{w}^{(j)}\right]$$

Because $M^{(0)} = M(0)$, we can solve this for $\mathbf{w}^{(k+1)}$ to obtain:

$$\mathbf{w}^{(k+1)} = \frac{1}{k+1}M(0)^{-1}\left[\rho^{(k)}\mathbf{b} - \sum_{j=1}^{k} jM^{(k-j+1)}\mathbf{w}^{(j)}\right] \tag{15}$$

This shows that the $(k+1)$th derivatives at $t = 0$ can be obtained from equation (15) with derivatives of order up to k.

It remains to determine $M^{(j)}$ and $\rho^{(j)}$ recursively. Let $S(t) = diag(s_1(t), \cdots, s_n(t))$. Because $Z(t) = S^2(t)$, we have:

$$Z^{(k)} = \sum_{j=0}^{k} S^{(k-j)}S^{(j)}$$

Using the identity $Z(t)\,Y(t) = I$, we obtain:

$$\sum_{j=0}^{k} Z^{(k-j)}Y^{(j)} = \mathbf{0}, \quad \text{for } k \geq 1$$

Thus,

$$Y^{(k)} = -Z(0)^{-1}\left[\sum_{j=0}^{k-1} Z^{(k-j)}Y^{(j)}\right], \quad \text{for } k \geq 1 \tag{16}$$

Because:

$$\begin{aligned} M^{(k-j+1)}\mathbf{w}^{(j)} &= AY^{(k-j+1)}A^T\mathbf{w}^{(j)} \\ &= -AY^{(k-j+1)}\mathbf{s}^{(j)} \end{aligned}$$

equation (15) then becomes:

$$w^{(k+1)} = \frac{1}{k+1}M(0)^{-1} \times \left[\rho^{(k)}\,\mathbf{b} + A\left(\sum_{j=1}^{k} jY^{(k-j+1)}\mathbf{s}^{(j)}\right)\right], \quad \text{for } k \geq 1 \tag{17}$$

where $Y^{(j)}$ is given by equation (16), $M(0) = AD^2(0)A^T$ with $D(0) = diag(1/s_i^0)$, and $\rho^{(k)}$ remains to be determined by the choice of power-series parameter specified implicitly by equation (12).

We will introduce some notation to simplify equation (17) even more. Let:

$$\mathbf{u}^k = \begin{cases} M(0)^{-1}\mathbf{b} & \text{if } k = 0 \\ M(0)^{-1}\left[A\left(\sum_{j=1}^{k} jY^{(k-j+1)}\mathbf{s}^{(j)}\right)\right] & \text{if } k > 0 \end{cases} \tag{18}$$

By equations (13) and (17), we now obtain:

$$\mathbf{w}^{(k+1)} = \begin{cases} \rho^{(0)}\mathbf{u}^0 & \text{if } k=0 \\ \frac{1}{k+1}\left[\rho^{(k)}\mathbf{u}^0 + \mathbf{u}^k\right] & \text{if } k>0 \end{cases} \tag{19}$$

Let:

$$\mathbf{p}^k = A^T\mathbf{u}^k, \text{ for } k \geq 0 \tag{20}$$

We obtain from equation (14) that:

$$\mathbf{s}^{(k+1)} = \begin{cases} -\rho^{(0)}\mathbf{p}^0 & \text{if } k=0 \\ \frac{-1}{k+1}\left[\rho^{(k)}\mathbf{p}^0 + \mathbf{p}^k\right] & \text{if } k>0 \end{cases} \tag{21}$$

What remains is to select the power-series parameter t or, equivalently, to choose $\rho(t)$ by specifying the coefficients $\rho^{(k)}$ in equation (12). This choice of $\rho(t)$ will depend on the current starting point $(\mathbf{w}^0, \mathbf{s}^0)$.

Choice of Power-Series Parameterization $\rho(t)$. The point set determined by the solution curve $(\mathbf{w}(t), \mathbf{s}(t))$ of system (10) is independent of the choice of the parameter t. Intuitively, the choice of t determines only the rate at which one travels along the trajectory. However, when this trajectory is approximated by a finite truncation of the power series, then different parameterizations of the trajectory lead to different approximation curves, for approximations of second order or higher. A good choice of parameterization is crucial to get an approximation curve that remains close to the central trajectory for a large-size step. Making a good choice depends on taking into account "global" properties of the dual-affine scaling trajectory in some region about the current point $(\mathbf{w}^0, \mathbf{s}^0)$.

Our choice of parameterization, $\rho(t)$, is motivated by the following simple example. Consider the linear program:

$$\begin{aligned} &\text{Maximize} && \mathbf{b}^T\mathbf{w} \\ &\text{subject to} && -\mathbf{w} \geq \mathbf{0} \end{aligned} \tag{22}$$

The interior of the polytope is $-\mathbf{R}^n_+ = \{\mathbf{w} : \text{all } w_i < 0\}$. This becomes a linear program in the dual-affine scaling form by adding slack variables $\mathbf{s} = (s_1, \cdots, s_n)$ with:

$$\begin{aligned} &\text{Maximize} && \mathbf{b}^T\mathbf{w} \\ &\text{subject to} && \mathbf{w} + \mathbf{s} = \mathbf{0} \\ &\text{for} && \mathbf{s} \geq \mathbf{0} \end{aligned} \tag{23}$$

Assume that $\mathbf{b} = (b_1, \cdots, b_n)$ has all $b_j > 0$. Then, this linear program has $\mathbf{w} = \mathbf{s} = \mathbf{0}$ as the unique optimal solution. We can explicitly compute the dual-affine scaling trajectories for this example.

Because $A = I$ is the identity matrix, the differential equation (9) for the continuous trajectories becomes:

$$\frac{d\mathbf{w}}{d\xi} = [D(\xi)]^{-2}\,\mathbf{b}$$

$$\frac{d\mathbf{s}}{d\xi} = -\frac{d\mathbf{w}}{d\xi}$$

$$\mathbf{w}(0) = -\mathbf{s}^0, \quad \mathbf{s}(0) = \mathbf{s}^0$$

where $D(\xi)^{-2} = diag(s_1(\xi)^2, \cdots, s_n(\xi)^2)$, and $\mathbf{s}^0 = (s_1^0, \cdots, s_n^0)$ from $\mathbf{R}^n_+$ is a set of slack variables that correspond to the initial feasible point $\mathbf{w}^0$ in $-\mathbf{R}^n_+$. These equations are exactly the set of Riccati equations:

$$\frac{ds_i}{d\xi} = -s_i^2 b_i, \text{ for } i = 1,2, \cdots, n$$

that have the explicit solutions:

$$s_i(\xi) = \frac{s_i^0}{1 + b_i s_i^0 \xi}, \text{ for } i = 1,2, \cdots, n$$

and $w_i(\xi) = -s_i(\xi)$. Clearly, the trajectory $\mathbf{w}(\xi)$ approaches the optimal point $\mathbf{0} = (0, 0, \cdots, 0)$ as $\xi \to \infty$.

The dual-affine scaling search direction $(\Delta\mathbf{w}, \Delta\mathbf{s})$ is given by the tangent vector to the curve $w_i(\xi)$ at $\mathbf{w}^0$, which is:

$$\Delta\mathbf{w}_i = -\Delta\mathbf{s}_i = b_i(s_i^0)^2, \text{ for } i = 1,2, \cdots, n$$

Let s_l denote the constraint that is violated first in taking a step in this direction, i.e., the blocking wall of the dual-affine scaling search direction. We claim that the choice:

$$t \equiv -[s_l(\xi) - s_l(0)]$$

is a good choice for the power-series parameter for this reason: *In this coordinate system, the optimal solution is strictly inside the region of convergence of the power-series expansion of the dual-affine scaling trajectory.* To prove this claim, we may assume—without loss of generality—that $b_1 \geq b_2 \geq \cdots \geq b_n > 0$ by permuting the coordinates, if necessary.

First, we consider the special case where $\mathbf{s}^0 = (1, 1, \cdots, 1)$. Then, each $s_i(\xi) = 1/(1 + b_i\xi)$, and $\Delta\mathbf{s}_i = -b_i$. Because $b_1 \geq b_i$ for all i, the constraint $s_1 = 0$ is the blocking wall of the dual-affine direction. Thus, the new parameter becomes:

$$t = 1 - s_1(\xi) = \frac{b_1\xi}{1 + b_1\xi} \tag{24}$$

This is a fractional linear transformation of ξ, which is easily inverted to yield:

$$\xi = \frac{(1/b_1)\, t}{1 - t} \equiv \phi(t)$$

Consequently,

$$s_i(t) \equiv s_i(\phi(t))$$

$$= \frac{1 - t}{\left[\frac{b_i}{b_1} - 1\right] t + 1}, \text{ for } i = 1,2, \cdots, n \tag{25}$$

The radius of convergence of the t-power series for $s_i(\phi(t))$ is $1/[1 - (b_i/b_1)]$, because $s_i(\phi(t))$ has a finite singularity only at $t = -1/[(b_i/b_1) - 1]$. Hence, the radius of convergence of $\mathbf{s}(\phi(t)) = (s_1(\phi(t)), \cdots, s_n(\phi(t)))$ is exactly $1/[1 - (b_n/b_1)]$ and this is *strictly greater than* 1. Formula (25) shows that, in the t-coordinate system, the optimal solution $\mathbf{0}$ is reached at $t = 1$, and this proves the claim in this special case.

The proof of the claim for a general $\mathbf{s}^0$ in $\mathbf{R}_+^n$ reduces to that of the special case by an affine transformation. Denote $p_i^0 = -\Delta\mathbf{s}_i = b_i(s_i^0)^2$. The blocking wall in the general case is determined by:

$$l = argmax \left\{ \frac{p_i^0}{s_i^0} : 1 \leq i \leq n \right\}$$

In this t-coordinate system, the constraint associated with the first blocking wall l is:

$$t \equiv s_l(0) - s_l(\xi) = s_l(0) - s_l(\phi(t)) \equiv s_l(0) - s_l(t)$$

For convenience, we will linearly rescale the new power-series parameter t to make a step of length 1 correspond to hitting the boundary if a dual-affine scaling step were taken. Thus, for a general linear program, we set:

$$s_l(0)t = -s_l(t) + s_l(0) \tag{26}$$

where l is the blocking wall. This is our *heuristic choice of power-series parameter.*

We have seen that this is a good choice of parameter for the "trivial" linear program (23). Moreover, for a general linear program that is dual nondegenerate:

- When one gets close to an optimal solution, the constraints that are close to being binding have equation (23)'s form (up to an affine transformation).
- The other constraints are "far away" and have little influence on the shape of the affine scaling trajectories.

Hence, one expects that this heuristic choice of parameter will perform well, at least at the later stages of the algorithm.

In equation (26), we made the choice of parameter. We now show how to compute recursively the expansion coefficients $\rho^{(k)}$ of $\rho(t) = \sum_{k=0}^{\infty} \rho^{(k)} t^k$, where $\rho(t) = d\xi/dt$. Define:

$$l = argmax \left\{ \frac{p_i^0}{s_i^0} : p_i^0 > 0 \right\}$$

This specifies the first constraint that will be violated in taking a step in the dual-affine scaling search direction. By definition, the heuristic choice of parameter t is:

$$s_l(0)\, t \equiv s_l^0 - s_l(t)$$

Hence, the coefficients:

$$s_l(t) \equiv \sum_{k=0}^{\infty} s_l^{(k)} t^k$$

satisfy:

$$s_l^{(0)} = s_l^0$$

$$s_l^{(1)} = -s_l^{(0)}$$

$$s_l^{(k)} = 0, \text{ for } k > 1 \qquad (27)$$

Now, we can insert formula (27) into (21) for the lth coordinate of $\mathbf{s}(t)$, and this gives a recursion for computing the desired $\rho^{(k)}$:

$$\rho^{(0)} = \frac{s_l^{(0)}}{p_l^0} = \frac{s_l^0}{p_l^0}$$

$$\rho^{(k)} = -\frac{p_l^k}{p_l^0}, \text{ for } k > 0 \qquad (28)$$

Recursions (19), (21), and (28) now permit the recursive calculation of the dual power-series coefficients in the t-coordinate system, as desired.

Primal-Dual Power-Series Algorithm. First, we describe the continuous version of the primal-dual algorithm. Because $\Delta \mathbf{y} \to d\mathbf{y}/d\xi$ as $\xi \to 0$, system (6) may be written as a system of ordinary differential equations:

$$A \frac{d\mathbf{x}}{d\xi} = \mathbf{0}$$

$$A^T \frac{d\mathbf{w}}{d\xi} + \frac{d\mathbf{s}}{d\xi} = \mathbf{0}$$

$$S \frac{d\mathbf{x}}{d\xi} + X \frac{d\mathbf{s}}{d\xi} = \mu\mathbf{e} - XS\mathbf{e}$$

$$\text{for } \mathbf{x}(0) = \mathbf{x}^0,\ \mathbf{w}(0) = \mathbf{w}^0,\ \mathbf{s}(0) = \mathbf{s}^0 \qquad (29)$$

Here, $\mu \geq 0$ is a fixed value of the penalty parameter. The solution $\mathbf{y}(\xi) = (\mathbf{x}(\xi), \mathbf{w}(\xi), \mathbf{s}(\xi))$ of system (29) is a continuous trajectory of the primal-dual algorithm.

The trajectory that arises from the differential equation (29) for any feasible initial condition $(\mathbf{x}(0), \mathbf{w}(0), \mathbf{s}(0))$ converges, as $\xi \to \infty$, to the point $\mathbf{y}(\mu)$ on the central path. In particular, the choice $\mu = 0$ gives the *primal-dual affine scaling trajectories.* However, in implementations of the algorithm, we shall take a positive value of μ at each step and decrease the value of μ as the algorithm progresses. As Renegar's analysis shows,[17] if the parameter σ in equation (5) is chosen sufficiently close to 1, then a first-order (Newton) step is already very good, and higher-order power-series methods will yield little improvement. But if bigger steps are taken [e.g., $\sigma = 1/2$, in which case the point $(\mathbf{x}^0, \mathbf{w}^0, \mathbf{s}^0)$ is usually outside the region where a Newton step is known to be good], then higher-order power-series approximations that follow these trajectories to $\mathbf{y}(\mu)$ may yield an improved step that gets much closer to $\mathbf{y}(\mu)$ than a first-order method can.

As presented in the dual power-series algorithm discussion earlier, it is important in practice to consider a reparameterized version of system (29), which is:

$$A \frac{d\mathbf{x}}{dt} = \mathbf{0}$$

$$A^T \frac{d\mathbf{w}}{dt} + \frac{d\mathbf{s}}{dt} = \mathbf{0}$$

$$S \frac{d\mathbf{x}}{dt} + X \frac{d\mathbf{s}}{dt} = \rho(t)\,\mathbf{v}(t)$$

$$\mathbf{x}(0) = \mathbf{x}^0,\ \ \mathbf{w}(0) = \mathbf{w}^0,\ \ \mathbf{s}(0) = \mathbf{s}^0 \qquad (30)$$

where $\rho(t) = d\xi/dt$ is a scalar parameterization function used to accelerate the algorithm and $\mathbf{v}(t) = \mu\mathbf{e} - X(t)\,S(t)\mathbf{e}$. We will find the solution of system (30) in the form of a power series. In addition to power series for $\mathbf{w}(t)$ and $\mathbf{s}(t)$, we will need similar expansions for $\mathbf{x}(t)$ and $\mathbf{v}(t)$:

$$\mathbf{x}(t) = \sum_{j=0}^{\infty} \mathbf{x}^{(j)} t^j$$

$$\mathbf{v}(t) = \sum_{j=0}^{\infty} v^{(j)} t^j.$$

To evaluate the power series for $(\mathbf{x}(t), \mathbf{w}(t), \mathbf{s}(t))$, we must compute $\mathbf{x}^{(k)}$, $\mathbf{w}^{(k)}$, and $\mathbf{s}^{(k)}$. Let:

$$\mathbf{f}^{(k)} = \sum_{j=0}^{k} X^{(j)} S^{(k-j)} \mathbf{e}$$

$$\bar{\mathbf{f}}^{(k)} = \sum_{j=1}^{k-1} X^{(j)} S^{(k-j)} \mathbf{e}$$

$$\bar{\mathbf{f}}^{(0)} \equiv \mathbf{0}$$

Then,

$$\mathbf{v}^{(k)} = \mu\,\delta_{0k}\mathbf{e} - \mathbf{f}^{(k)}$$

Here, δ_{0k} is the Kronecker delta symbol and $k \geq 0$.

When we differentiate equation (30) k times and let $t = 0$, we obtain the following linear system:

$$A\mathbf{x}^{(k+1)} = \mathbf{0}$$

$$A^T\mathbf{w}^{(k+1)} + \mathbf{s}^{(k+1)} = \mathbf{0}$$

$$S\mathbf{x}^{(k+1)} + X\mathbf{s}^{(k+1)} = \frac{1}{k+1} \sum_{j=0}^{k} \rho^{(j)} v^{(k-j)} - \bar{\mathbf{f}}^{(k+1)} \qquad (31)$$

We observe that systems (31) and (30) differ only in their right-hand sides. We also see that the right-hand side of equation (31) depends only on the derivatives of order less than or equal to k. System (31) can be solved for $\mathbf{x}^{(k+1)}$, $\mathbf{w}^{(k+1)}$, and $\mathbf{s}^{(k+1)}$ as follows:

$$\mathbf{w}^{(k+1)} = \frac{1}{k+1} \sum_{j=0}^{k} \rho^{(j)} \hat{\mathbf{w}}^{(k-j)} - \bar{\mathbf{w}}^{(k+1)}$$

$$\mathbf{s}^{(k+1)} = \frac{1}{k+1} \sum_{j=0}^{k} \rho^{(j)} \hat{\mathbf{s}}^{(k-j)} - \bar{\mathbf{s}}^{(k+1)}$$

$$\mathbf{x}^{(k+1)} = \frac{1}{k+1} \sum_{j=0}^{k} \rho^{(j)} \hat{\mathbf{x}}^{(k-j)} - \bar{\mathbf{x}}^{(k+1)} \qquad (32)$$

where

$$\hat{\mathbf{w}}^{(k)} = -[AD^2(0)A^T]^{-1}AS^{-1}(0)\mathbf{v}^{(k)}$$

$$\bar{\mathbf{w}}^{(k)} = -[AD^2(0)A^T]^{-1}AS^{-1}(0)\bar{\mathbf{f}}^{(k)}$$

$$\hat{\mathbf{s}}^{(k)} = -A^T\hat{\mathbf{w}}^{(k)}$$

$$\bar{\mathbf{s}}^{(k)} = -A^T\bar{\mathbf{w}}^{(k)}$$

$$\hat{\mathbf{x}}^{(k)} = S^{-1}(0)\mathbf{v}^{(k)} - D^2(0)\hat{\mathbf{s}}^{(k)}$$

$$\bar{\mathbf{x}}^{(k)} = S^{-1}(0)\bar{\mathbf{f}}^{(k)} - D^2(0)\bar{\mathbf{s}}^{(k)} \quad (33)$$

Here, $k \geq 0$. It remains to determine the coefficients $\rho^{(k)}$ of $\rho(t)$, which will depend on $(\mathbf{x}^0, \mathbf{w}^0, \mathbf{s}^0)$.

Choice of Primal-Dual Power-Series Parameter. We use the same heuristic here as we did to choose the power-series parameter $\rho(t)$. That is, we set the parameter t essentially equal to the variable that corresponds to a step's blocking wall in the (first-order) primal-dual Newton search direction for the given value $\bar{\mu}$. This variable may be either some primal variable [i.e., set $x_l(0)t = -x_l(t) + x_l(0)$] or a dual slack variable [i.e., set $s_l(0)t = -s_l(t) + s_l(0)$].

By equation (8), the Newton search direction step of length ξ is:

$$x_i^1 = x_i^0 + \xi\Delta\mathbf{x}_i$$

$$s_i^1 = s_i^0 + \xi\Delta\mathbf{s}_i$$

where $\Delta\mathbf{x} = \hat{\mathbf{x}}^{(0)}$ and $\Delta\mathbf{s} = \hat{\mathbf{s}}^{(0)}$ are given by equations (7) and (33). The smallest value of ξ, such that some $x_l^1 = 0$ or some $s_l^1 = 0$, determines the *blocking wall*; and the corresponding variable is the *blocking variable*. Now, we give the formulas for $\rho^{(k)}$ for the case where x_l^1 is the blocking variable. (The formulas for the case where s_l^1 is a blocking variable are analogous.) Again, as when choosing $\rho(t)$, we have:

$$x_l(0)t = -x_l(t) + x_l(0)$$

Hence,

$$x_l(t) = \sum_{k=0}^{\infty} x_l^{(k)}t^k \equiv x_l^0 - x_l^0 t$$

and we obtain:

$$x_l^{(1)} = -x_l^0$$

$$x_l^{(k)} = 0, \quad \text{for } k > 1 \quad (34)$$

This uniquely determines $\rho^{(k)}$. Indeed, from equations (32) and (34), we have:

$$x_l^{(1)} = \rho^{(0)}\hat{x}_l^{(0)} = -x_l^0$$

so that:

$$\rho(0) = -\frac{x_l^0}{\hat{x}_l^{(0)}} \quad (35)$$

For $k > 1$, we obtain:

$$x_l^{(k+1)} = 0 = \frac{1}{k+1}\left[\rho^{(k)}\hat{x}_l^{(0)} + \sum_{j=0}^{k-1}\rho^{(j)}\hat{x}_l^{(k-j)}\right] - \bar{x}^{(k+1)}$$

So,

$$\rho^{(k)} = \frac{(k+1)\,\bar{x}_l^{(k+1)} - \sum_{j=0}^{k-1}\rho^{(j)}\hat{x}_l^{(k-j)}}{\hat{x}_l^{(0)}} \quad (36)$$

This concludes the derivation of the primal-dual power-series formulas.

Computational Results

Both power-series algorithms discussed above have been implemented in the AT&T KORBX system.[21]

Table I. Number of iterations for small and/or sparse problems

	Problem size			Number of iterations			
Problem	m	n	nz	Dual	Dual-PS(5)	PD	PD-PS(3)
bandm	305	472	2,494	23	13	27	18
brandy	220	249	2,148	21	12	25	20
capri	271	353	1,767	25	13	28	19
ken10	10,037	14,837	33,337	28	14	28	19
ken14	38,496	47,316	110,232	57	27	34	24

NOTE: m = number of columns in the matrix A; n = number of rows in the matrix A; nz = number of nonzero elements in the matrix A; dual = dual power series of order 1; dual-PS(5) = dual power series of order 5; PD = primal-dual power series of order 1; PD-PS(3) = dual power series of order 3.

Table II. Number of iterations for large and/or dense problems

	Problem size			Number of iterations			
Problem	m	n	nz	Dual	Dual-PS(5)	PD	PD-PS(3)
flight	1,441	3,652	43,167	38	21	39	28
p01p01	3,605	8,194	90,452	50	25	57	40
day5	7,291	22,801	48,926	63	25	42	26
t01p06	4,420	6,711	101,377	46	24	39	23
t01p09	6,642	10,707	203,597	52	28	49	29

NOTE: m = number of columns in the matrix A; n = number of rows in the matrix A; nz = number of nonzero elements in the matrix A; dual = dual power series of order 1; dual-PS(5) = dual power series of order 5; PD = primal-dual power series of order 1; PD-PS(3) = dual power series of order 3.

All experiments reported here were performed on the KORBX system processor with eight ***advanced computational elements***, using Release 2.8.0 of KLP.

The step sizes for the two algorithms used in the tests described below were computed as follows. The first-order dual-affine scaling algorithm takes a step $t = 0.995$. (This is 0.995 of the distance to the blocking wall.) The higher-order dual-affine power-series algorithm takes the smallest value of $j = 1, 2, \cdots$ for which $t = (0.995)^j$ is strictly feasible. For the primal-dual power-series algorithm, the largest value t_0 of t that is feasible with $0 \leq t \leq 2$ is determined by binary search, and the step size taken is $0.999\, t_0$. The path-following parameter $\bar{\mu}$ is defined as $\bar{\mu}^{(k+1)} = \sigma^{(k+1)}\, \bar{\mu}^{(k)}$. The parameter $\sigma^{(k+1)}$ in equation (5) is determined by $\sigma^{(k+1)} = (1/2)\, [g^{(k)}/g^{(k-1)}]$, where $g^{(k)}$ is the current duality gap and $g^{(k-1)}$ is the duality gap at the previous iteration, with $\sigma = 1/2$ being used at the first iteration. These choices of step size were used for initial experiments; improved step-size heuristics are under development.

Two sets of test problems, each consisting of five problems, are presented. The first set consists of small-size problems and some relatively sparse problems, while the second set consists of medium- and large-size

Table III. CPU time for small and/or sparse problems

Problem	Problem size			CPU time (seconds)			
	m	n	nz	Dual	Dual-PS(5)	PD	PD-PS(3)
`bandm`	305	472	2494	5.1	5.2	5.3	6.2
`brandy`	220	249	2148	3.6	3.3	3.8	4.8
`capri`	271	353	1767	7.0	6.1	6.0	7.1
`ken10`	10,037	14,837	33,337	147.0	178.0	172.0	191.0
`ken14`	38,496	47,316	110,232	1225.0	1460.0	928.0	1148.0

NOTE: m = number of columns in the matrix A; n = number of rows in the matrix A; nz = number of nonzero elements in the matrix A; dual = dual power series of order 1; dual-PS(5) = dual power series of order 5; PD = primal-dual power series of order 1; PD-PS(3) = dual power series of order 3.

problems and some less sparse problems. In both problem sets, we found that the dual power series of order 5 and the primal-dual power series of order 3 reduce the number of iterations by almost 40 percent in comparison with their order-1 algorithms (as shown in Tables I and II). The decrease in the number of iterations observed here is typical of what we observed in about 300 test problems. In each table, m and n are the dimensions of the matrix A, and nz is the number of nonzero elements in A.

From the previous sections, the major computational effort involves solving a system of linear equations—namely, $AD^2A^T\mathbf{y} = \mathbf{x}$. For a direct method, the above system is usually solved by using Cholesky factorization together with forward-and-backward substitutions. Specifically,

$$AD^2A^T = \hat{L}\,D'\,\hat{L}^T$$

where D' is a diagonal matrix, and $\hat{L}$ is the lower triangular Cholesky factor. Using the factor $\hat{L}$, the linear equation can then be solved by a forward-and-backward substitution.

The running time of one iteration of these power-series algorithms has three components: a Cholesky factorization step, a forward-and-backward solve for each term in the power series, and recursion steps given by equations (17) and (32), respectively. For the power-series algorithms, equations (18) and (33) show that the coefficient matrices of the linear equations to be solved are the same as those of order-1 algorithms, but their right-hand sides are different. Consequently, the Cholesky factor obtained for the order-1 term can be reused for higher-order terms. For the dual power-series method, only one forward-and-backward solve is needed for each additional order, while in the primal-dual method, two forward-and-backward solves are needed. Thus, the Cholesky factorization step is the same for first-order and higher-order methods, there are $O(k)$ forward-and-backward solves for a kth-order power-series method [taking $O(km^2)$ arithmetic operations in all], and all the recursion steps for a kth-order power-series method take $O(k^2)$ vector operations [for $O(k^2m)$ arithmetic operations in all]. For small k, the forward-and-backward solves computationally dominate the recursion steps, although the recursion steps could become important for large enough values of k.

In terms of computational speed, use of the power series of higher orders will be advantageous if the time saved in reducing the number of iterations (mainly Cholesky factorization) offsets the extra time in the forward-and-backward solve. For small and/or very sparse problems, factorization complexity is comparable

Table IV. CPU time for large and/or dense problems

	Problem size			CPU time (seconds)			
Problem	*m*	*n*	*nz*	Dual	Dual-PS(5)	PD	PD-PS(3)
`flight`	1,441	3,652	43,167	267	166	291	251
`p01p01`	3,605	8,194	90,452	328	206	383	311
`day5`	7,291	22,801	48,926	733	485	669	522
`t01p06`	4,420	6,711	101,377	2153	1149	1992	923
`t01p09`	6,642	10,707	203,597	8303	4155	8413	4623

NOTE: *m* = number of columns in the matrix *A*; *n* = number of rows in the matrix *A*; *nz* = number of nonzero elements in the matrix *A*; dual = dual power series of order 1; dual-PS(5) = dual power series of order 5; PD = primal-dual power series of order 1; PD-PS(3) = dual power series of order 3.

to the amount of computations required by the evaluation of higher-order terms. This phenomenon is shown in Table III where we see that the power-series methods are roughly comparable to (but slightly worse than) the corresponding order-1 methods. However, for dense and/or large problems, the computational complexity of the factorization dominates the complexity of the power-series evaluation, and a significant reduction in CPU time is achieved. Table IV shows the reductions for the dual power series of order 5 and the primal-dual power series of order 3.

References

1. N. Karmarkar, "A New Polynomial Time Algorithm for Linear Programming," *Combinatorica*, Vol. 4, No. 4, 1984, pp. 373-395.
2. G. Sonnevend, "An 'Analytical Center' for Polyhedrons and New Classes of Global Algorithms for Linear (Smooth, Convex) Programming," *Proceedings of the 12th IFIP Conference on System Modeling, Budapest 1985*, Lecture Notes in Control and Information Science, No. 84, Springer-Verlag, New York, 1986, pp. 866-876.
3. I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga, "An Implementation of Karmarkar's Algorithm for Linear Programming," Technical Report ORC 86-8, Operations Research Center, University of California, Berkeley, California, 1986. (To appear in *Mathematical Programming*.)
4. N. Karmarkar and K. G. Ramakrishnan, "Implementation and Computational Results of the Karmarkar Algorithm for Linear Programming, Using an Iterative Method for Computing Projections," presented at the 13th International Mathematical Programming Symposium, Tokyo, Japan, August 1988.
5. C. L. Monma and A. J. Morton, "Computational Experiences with a Dual Affine Variant of Karmarkar's Method for Linear Programming," *Operations Research Letters*, Vol. 6, 1987, pp. 261-267.
6. I. I. Dikin, "Iterative Solution of Problems of Linear and Quadratic Programming," *Soviet Mathematics Doklady*, Vol. 8, No. 3, 1967, pp. 674-675.
7. I. I. Dikin, "On the Speed of an Iterative Process," *Upravlyaemye Sistemi*, 1974, pp. 1250-1260.
8. E. Barnes, "A Variation on Karmarkar's Algorithm for Solving Linear Programming Problems," *Mathematical Programming*, Vol. 36, 1986, pp. 174-182.
9. R. J. Vanderbei, M. S. Meketon, and B. A. Freedman, "A Modification of Karmarkar's Algorithm," *Algorithmica*, Vol. 1, 1986, pp. 395-407.
10. D. A. Bayer and J. C. Lagarias, "The Nonlinear Geometry of Linear Programming, I. Affine and Projective Scaling Trajectories," *Transactions of the American Mathematical Society*, to be published, 1989.
11. Y. Ye, "A Class of Potential Functions for Linear Programming," Management Science Working Paper, University of Iowa, No. 88-13, 1988.
12. D. A. Bayer and J. C. Lagarias, "The Nonlinear Geometry of Linear Programming, II. Legendre Transform coordinates and Central Trajectories," *Transactions of the American Mathematical Society*, to be published, 1989.
13. J. C. Lagarias, "The Nonlinear Geometry of Linear Programming, III. Projective Legendre Transform coordinates and Hilbert Geometry," *Transactions of the American Mathematical Society*, to be published, 1989.
14. N. Megiddo, "Pathways to the Optimal Set in Linear Programming," *Proceedings of the 12th International Mathematical Programming Symposium*, Tokyo, Japan, 1986, pp. 1-34.

15. M. Kojima, S. Mizuno, and A. Yoshise, "A Primal-Dual Interior Point Algorithm for Linear Programming," *Progress in Mathematical Programming, Interior-Point and Related Methods,* N. Megiddo (ed.), Springer-Verlag, New York, 1989, pp. 29-48.
16. R. D. C. Monteiro, I. Adler, and M. G. C. Resende, "A Polynomial-time Primal-Dual Affine Scaling Algorithm for Linear and Convex Quadratic Programming and its Power Series Extension," preprint, Industrial Engineering and Operations Research Department, University of California, Berkeley, California, 1988.
17. J. Renegar, "A Polynomial-time Algorithm, Based on Newton's Method, for Linear Programming," *Mathematical Programming,* Vol. 40, 1988, pp. 59-94.
18. P. Vaidya, "An Algorithm for Linear Programming Which Requires $O((m+n)n^2 + (m+n)^{1.5}nL)$ Arithmetic Operations," *Proceedings of the 19th ACM Symposium on Theory of Computing,* Association for Computing Machinery, New York, New York, 1987, pp. 29-38.
19. C. Gonzaga, "An Algorithm for Solving Linear Programming in $O(n^3L)$ Operations," *Progress in Mathematical Programming, Interior-Point and Related Methods* N. Megiddo (ed.), Springer-Verlag, New York, 1989, pp. 1-28
20. D. A. Bayer, N. Karmarkar, and J. C. Lagarias, "Method and Apparatus for Optimizing System Operational Parameters," U.S. Patent 4,744,027, issued May 10, 1988.
21. Y. C. Cheng, D. J. Houck, J. M. Liu, M. S. Meketon, L. Slutsman, R. J. Vanderbei, and P. Wang, "The AT&T KORBX® System," *AT&T Technical Journal,* Vol. 68, No. 3, May/June 1989, pp. 7-19.
22. R. J. Vanderbei, "The Affine-scaling Algorithm for Linear Programs with Free Variables," *Mathematical Programming,* Vol. 43, 1989, pp. 31-44.

Biographies (continued)

mathematics, computational complexity theory, number theory, cryptography, and mathematical programming. He joined the company in 1974 and has an S.B., S.M., and Ph.D. in mathematics from the Massachusetts Institute of Technology. Mr. Slutsman is a distinguished member of technical staff and is responsible for design and development of primal-dual algorithms for solving linear and quadratic programming problems. He joined the company in 1981 and has an M.S. in mathematics from the University of Leningrad, U.S.S.R., and a Ph.D. in applied mathematics from Leningrad Mining Institute, U.S.S.R. Mr. Wang is a distinguished member of technical staff and works on airline-schedule planning problems. He joined the company in 1980 and has a B.S. in mathematics from the National Taiwan Normal University, Taiwan; an M.S. in operations research and systems analysis from the University of North Carolina at Chapel Hill; and a Ph.D. in mathematics from the University of Illinois at Champaign-Urbana.

(Manuscript received February 14, 1989)