SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853

TECHNICAL REPORT NO. 958

February 1991

PLAYING WITH INTERIOR POINTS

by

Michael J. Todd*

*Research supported in part by NSF, AFORS and ONR through NSF grant DMS-8920550.

PLAYING WITH INTERIOR POINTS

by

Michael J. Todd
School of Operations Research
and Industrial Engineering
229 Engineering & Theory Center Bldg.
Cornell University
Ithaca, NY 14853

Abstract. We describe some insights that have been obtained from experimenting with various interior-point algorithms on random linear programming problems.

1. Introduction

Most published computational results for interior-point approaches to linear programming have been concerned with state-of-the-art implementations of various algorithms to suites of large sparse test problems, usually including but not limited to the NETLIB suite (Gay [8]). Among such studies are those of Adler, Karmarkar, Resende and Veiga [1] and Monma and Morton [17] for the dual affine-scaling method; and those of McShane, Monma and Shanno [13], Choi, Monma and Shanno [5], Lustig, Marsten and Shanno [11,12], and Mehrotra [15] for primal-dual methods. However, if one is interested primarily in how the algorithms deal with certain features of the problems or how various families of algorithms compare, it often suffices to apply simple implementations to relatively small dense problems. Examples include Anstreicher and Watteyne [3], Todd and Wang [22], and Todd [19,20].

Here I do not wish to present detailed comparisons of various algorithms. Instead I will give some qualitative conclusions, and also stress the insights that can be obtained by observing the performance of different methods.

A delightfully painless environment for making such observations is provided by MATLAB [16], a high-performance interactive software package for performing matrix computations and displaying results. I will illustrate the ease of use of this package by displaying the MATLAB code for computing the projections of the scaled cost vector and the vector of ones onto the null space of the scaled coefficient matrix:

```
function [cp, ep] = projc(Abar, cbar)
%       computes the projections of cbar and  e  into the null
%       space of Abar.
cpep = [cbar ones(cbar)] - Abar' * (Abar' \ [cbar ones(cbar)]);
cp   = cpep(:,1);
ep   = cpep(:,2);
return
```

Research supported in part by NSF, AFORS and ONR through NSF grant DMS-8920550.

Here cbar ($\bar{c}$) and Abar ($\bar{A}$) are the scaled cost vector and coefficient matrix, ones(cbar) is the vector (e) of ones of the same dimension as cbar, and Abar$'$ \ [cbar ones(cbar)] yields the matrix $[y_c\ y_e]$, where y_c is the least-squares solution to $\bar{A}^T y_c \approx \bar{c}$ and y_e that to $\bar{A}^T y_e \approx e$. MATLAB computations are carried out in a numerically stable way, are remarkably efficient for dense matrices, and require, as is evident from the example above, very little programming time.

In section 2 I will discuss using MATLAB to elucidate the relationship between the low-complexity algorithm in [20] and path-following methods. Section 3 compares a more efficient version of the low-complexity algorithm to a variant which bears a strong resemblance to the methods of Gonzaga [9], Ye [25] and Freund [7] and also to the affine-scaling algorithm. Finally, in section 4 I consider solving problems where an initial strictly feasible solution is not at hand.

2. Following the central path

The basic low-complexity algorithm (BLCA) in [20] solves the standard-form linear programming problem

$$\text{(P)} \qquad \begin{aligned} \min\ & c^T x \\ Ax &= b \\ x &\geq 0 \end{aligned}$$

as follows. Given a strictly positive feasible solution $\hat{x}$, let $\bar{A} = A\hat{X}$ and $\bar{c} = \hat{X}c$, where $\hat{X} = \text{diag}(\hat{x})$. These are the transformed data in the scaled problem

$$\text{($\bar{P}$)} \qquad \begin{aligned} \min\ & \bar{c}^T \bar{x} \\ \bar{A}\bar{x} &= b \\ \bar{x} &\geq 0 \end{aligned}$$

in terms of the scaled variables $\bar{x} = \hat{X}^{-1}x$. Note that the current iterate $\hat{x}$ corresponds to $\bar{x} = e$, the vector of ones.

The algorithm first computes the projections $\bar{c}_p$ and e_p of $\bar{c}$ and e onto the null space of $\bar{A}$, and sets $\bar{d}_\alpha := -\alpha\bar{c}_p + e_p$ where $\alpha := \bar{c}_p^T e / \bar{c}_p^T \bar{c}_p$. Then the search direction is $\bar{d} = \bar{d}_\alpha / \|\bar{d}_\alpha\|$ if $\|\bar{d}_\alpha\| \geq .3$ and $\bar{d} = -\bar{c}_p / \|\bar{c}_p\|$ otherwise. The new point is $\bar{x}_+ = e + .2\bar{d}$ in the scaled space, and $x_+ = \hat{X}\bar{x}_+$ in the original space. Perhaps surprisingly, this algorithm achieves the best known complexity in terms of the number of iterations: given a suitable starting point, it gets within 2^{-t} of the optimal value of (P) within $O(\sqrt{n}\ t)$ iterations, where n is the number of variables in (P).

The BLCA takes a step in the direction $\bar{d}_\alpha / \|\bar{d}_\alpha\|$ (a constant-cost centering step) if $\|\bar{d}_\alpha\|$ is large, and a step in the direction $-\bar{c}_p / \|\bar{c}_p\|$ (the affine-scaling direction of Dikin) if $\|\bar{d}_\alpha\|$ is small. In the latter case, $\bar{c}_p$ and e_p are in some sense close to being collinear, which characterizes the

central path (e.g. [18]). Indeed, Todd and Vial [21] recently showed that $\|\bar{d}_\alpha\|$ is exactly the measure of centrality of Roos and Vial [18]. Further, they showed that after some initial centering steps, each affine-scaling step is followed by at most three centering steps before another affine-scaling step. Hence the BLCA generates iterates close to the central path.

What only emerged from computational testing is how astonishingly close the iterates remain. Indeed, after the initial centering phase, no further centering steps were made in several runs on random standard-form problems of dimensions 50×100 up to 300×600, and in fact $\|\bar{d}_\alpha\|$ continued to decrease. This encouraged us to try modifications. Suppose we perform a line search on the logarithmic barrier function along the constant-cost centering direction if $\|\bar{d}_\alpha\| > 9/10$. We take a Newton step in this direction if $\|\bar{d}_\alpha\|$ lies between 1/16 and 9/10. Finally, if $\|\bar{d}_\alpha\|$ is smaller than 1/16, we take a step of length .4 (rather than .2) in the affine-scaling direction. It can be shown [21] that, after the initial centering phase, each affine-scaling step is followed by at most two centering steps; but in practice, a single centering step is only needed every 10-20 iterations.

In Figure 1, we show the trajectories of these algorithms on a 50×100 problem. Figure 1(a) plots the 31st component of x against the 30th, while 1(b) plots the 34th against the 33th. The dashed line corresponds to the BLCA, which requires 347 iterations; every tenth iterate is marked with a "+". The dotted line is the modification described above, needing only 170 iterations; every tenth iterate is marked with a "x." Finally, the solid line gives the progress of the most efficient variant of the BLCA, called variant 2 in [20], with $q = 2n$; this needs only 11 iterations. All trajectories start at (1,1).

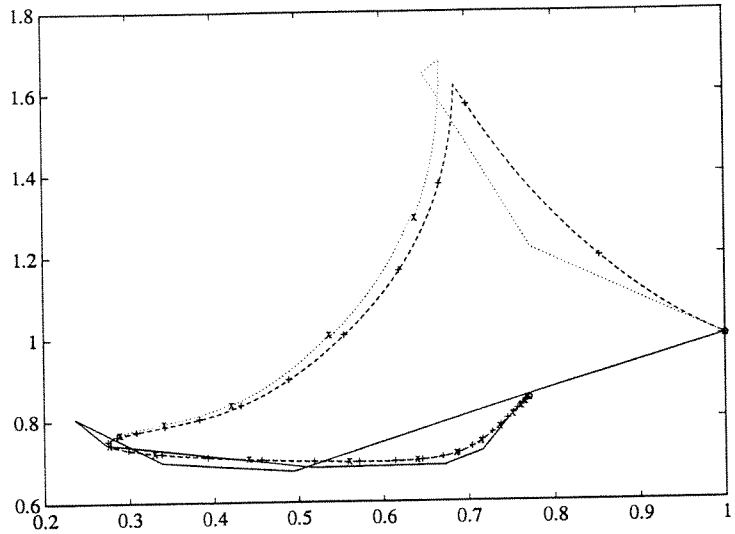

Figure 1(a)

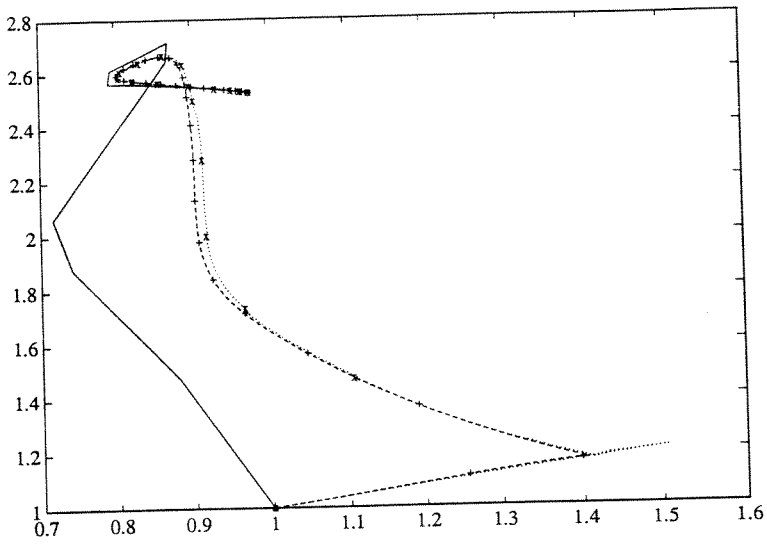

Figure 1(b)

Notice how the modified BLCA reaches the vicinity of the central path in only 3 iterations, and after 10 is further along the path than the BLCA after 30 iterations. Both these methods follow very smooth paths. (Experiments with modifications taking step sizes of .8 or .95 along the affine-scaling direction lead to similar results, with fewer iterations and still a very small number of centering steps.) We must conclude that interior-point methods whose step is of fixed (or uniformly bounded by one) Euclidean length in the scaled space seem to be very short-step methods with a strong resemblance to path-following methods. Such methods include the original projective-scaling algorithm of Karmarkar [10] (although a line search is usually employed) and the affine-scaling method with the step size analyzed by Dikin [6], Barnes [4], and Tsuchiya [23] (in practice, a step a certain proportion of the way to the boundary is typically used). See also Xiao and Goldfarb [24] and Anstreicher [2].

We also remark that the efficient method (the solid line) also appears to follow the central path quite closely after its initial few steps.

3. Problems with e feasible

Here we briefly discuss experiments on problems generated so that the vector of ones, e, yields an initial feasible solution. (Each entry of A was generated as an independent standard Gaussian random variable; b was set to Ae; and c was set to $A^T y + s$, where y and s were generated as was A, and then s replaced by |s|, the vector of absolute values of the components of s. This was also the way the problems of the previous section were generated.) We tested variant 2 of the BLCA (with $q = 2n$) against the affine-scaling algorithm, with step size .95 of the way to the boundary of the feasible region, and against another variant of the BLCA. We call this variant 3; it only differs from variant 2 in its choice of direction, which is $\bar{d}_\zeta$ (see section 4 of [20]). This is very close to the potential reduction methods of Gonzaga [9], Ye [25] and Freund [7]. The direction chosen by variant 2 is usually half-way between $\bar{d}_\zeta$ and the affine-scaling direction. With $q = 2n$, both variants require O(nt) iterations to obtain t digits of accuracy.

The termination criterion in all cases was that the relative error in the objective function be no more than 10^{-4}. (In the line search, we checked whether a step all the way to the boundary would satisfy this criterion, and, if so, took such a step.) Here the relative error is with respect to the lower bound. Such a lower bound is generated in variants 2 and 3, using lemma 5 of [20]. We also found that the same procedure generated lower bounds for the affine-scaling algorithm. The latter is normally terminated when the relative improvement in the objective function falls below a certain level. Especially if a dual solution is required, it might be worth adding a further test at this stage, by trying to compute a lower bound using lemma 5 of [20]. If successful, this provides a guarantee of the quality of the solution as well as a near-optimal dual solution, and our results indicate that success is likely for well-scaled problems with near-central initial solutions. The additional cost required is a further least-squares solution, to obtain e_p.

The table below gives the average number of iterations required for five random problems and the three algrithms discussed above. For the affine-scaling algorithm, the average proportion of the way to the boundary was (of course) .95. For variant 2, it ranged from .96 to .99, and for variant 3, from .88 to .99. The advantage of the latter algorithms is that their use of a potential function allows further flexibility: short steps will be taken if necessary, longer steps if safe.

	100×200	200×400	300×600	400×800
affine-scaling	11.8	12.8	13.8	14.4
variant 2	12.2	13.6	13.8	14.4
variant 3	13.6	16.0	18.2	19.2

We see that in an algorithm with a polynomial-time guarantee it generally pays to choose a direction between that of the pure potential reduction algorithm and that of the affine-scaling algorithm, and that the penalty of using variant 2 instead of the latter is very slight in terms of number of iterations. (Unfortunately, variant 2 needs an extra projection at each iteration.)

Finally, all the runs with variant 2, and to a lesser extent, the other methods, show a strong symmetry between the objective function values and the lower bounds generated -- see Figure 2 for a typical example, where the solid line shows the progress of the objective and the dotted line that of the bound; at each iteration, both have roughly the same accuracy.

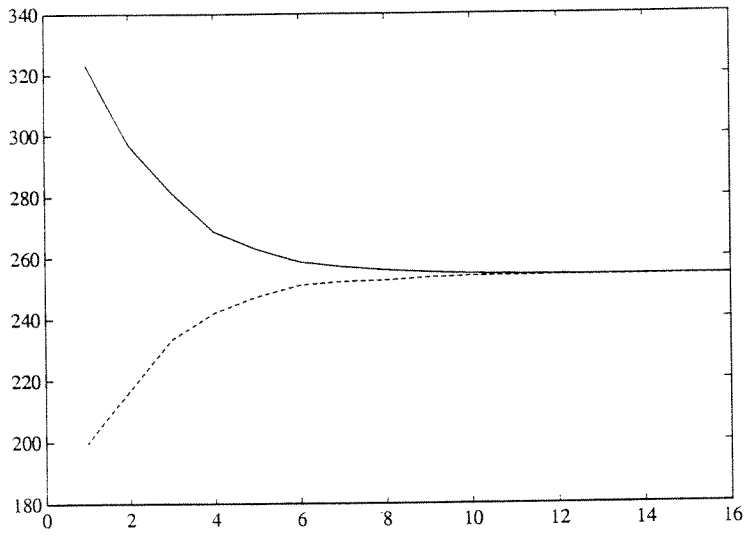

Figure 2.

4. Problems without initial feasible point

We conclude by making some brief comments about our results where an initial feasible point is not available. We generate such problems as in section 3, but with $b = A\tilde{x}$ where each component of $\tilde{x}$ is generated as the absolute value of an independent standard Gaussian random variable. Now we add an artificial variable with column b - Ae and cost 10^{10} to get a new problem with an extra variable. We modified the line search to go all the way to the boundary if this eliminated the artificial variable.

Both variants 2 and 3 had difficulty with such problems if they were initialized with lower bounds of $-\infty$. The iterates grew very large (usually to the order of 10^8) before returning to the neighborhood of the optimal solution, as in the experiments in [19]. We therefore initialized the lower bound to -10^5, as in [19]. In this case, variant 2 was usually comparable to the affine-scaling algorithm and two to six iterations faster than variant 3. Average iteration counts varied from 13.4 to 28.0. Typical trajectories for a 300×599 problem are shown in figure 3(a). Here the dashed line, marked with a "+" every ten iterations, shows variant 2; the solid line, with "o" 's, shows the affine-scaling method and the dotted line, with "x" 's, illustrates variant 3. The algorithms require 19, 18 and 22 iterations respectively. If we modified variant 3 to make it monotonic ($\bar{d} = \bar{d}_\zeta$ if $\zeta > \alpha$, else $\bar{d} = \bar{d}_\alpha$), its behavior becomes almost identical to that of variant 2 -- see figure 3(b); 19 iterations are now required.

Finally, we examined the behavior of these methods when started at the feasible solution $\tilde{x}$, so that an artificial variable is not needed. The behavior of variant 2 (now with initial bound $-\infty$) was comparable to its behavior on the artificial problem (with initial bound -10^5). The affine-scaling method performed very poorly, partly because of its difficulty in generating lower bounds, but also because some variables became prematurely very small. Variant 3 needed either an initial lower bound of -10^5 or monotonic directions to perform adequately. In the latter case it was again comparable to variant 2. These results are illustrated by the trajectories for the same problem as before. (Here the trajectories start near (1,.3).) Figure 3(c) shows variant 2 (dashed line, 20 iterations), the affine-scaling algorithm (solid line, 85 iterations), and variant 3 with initial bound -10^5 (dotted line, 36 iterations). Notice how the latter initially has variables increasing substantially -- in fact, some reach the order of 10^3, while the initial and final solutions are of order 1. Figure 3(d) shows the first two algorithms as above, and variant 3 with monotonic directions (dotted line, 20 iterations). Here we see the poor behavior of the affine-scaling method -- one variable becomes much too small and impedes progress.

Overall, the results suggest that the affine-scaling algorithm can perform very badly when the initial solution is far from centered (compare with the analytic results of Megiddo and Shub [14]), and that it is perhaps preferable to introduce an artificial variable even if a feasible solution is known, or to take some initial centering steps.

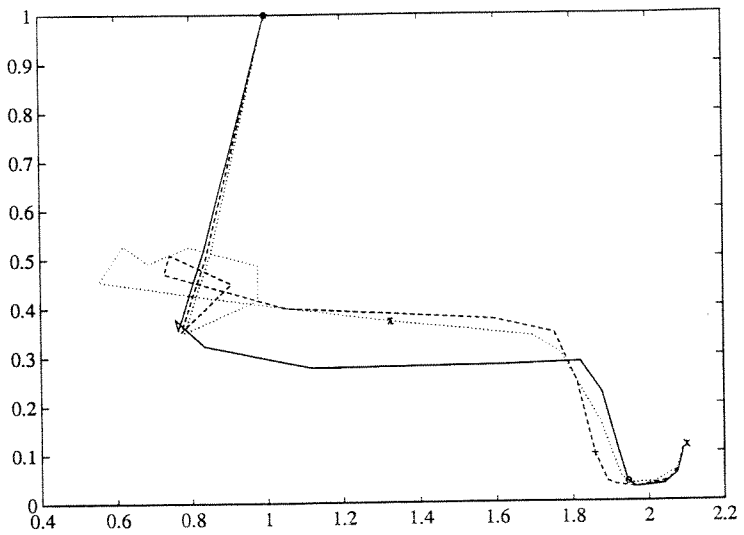

Figure 3(a)

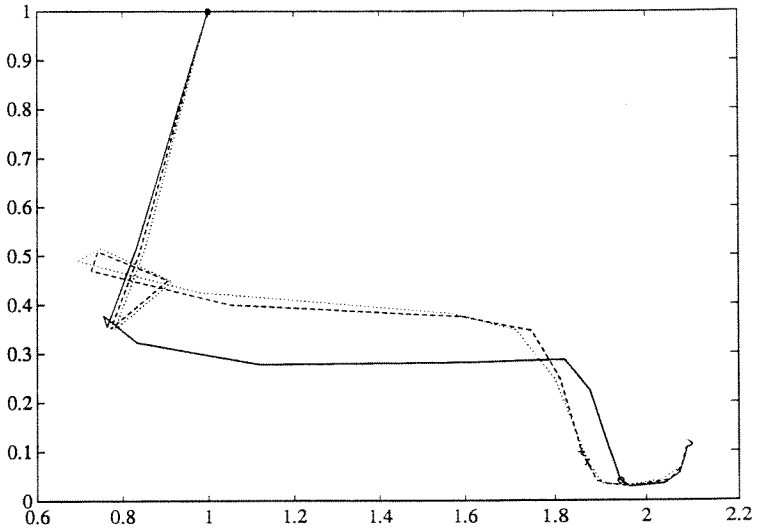

Figure 3(b)

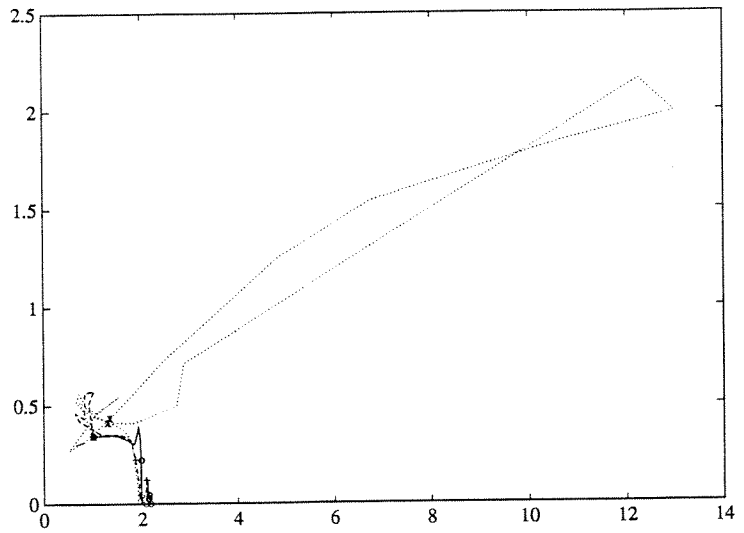

Figure 3(c)

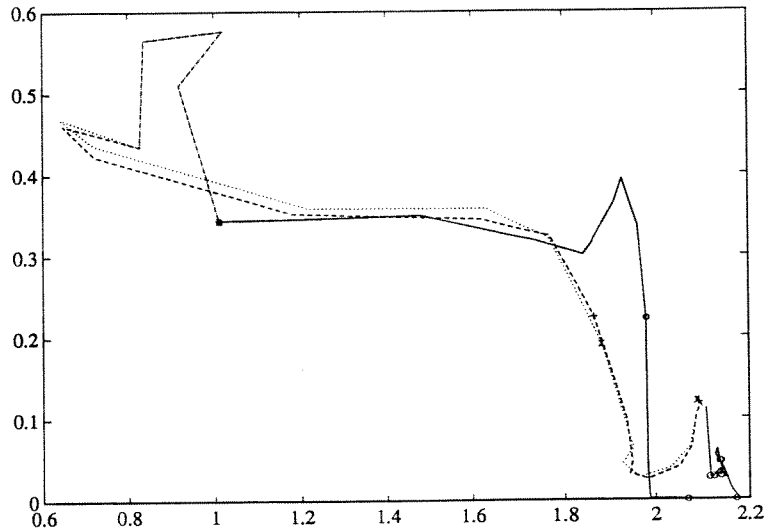

Figure 3(d)

References

[1] I. Adler, M.G.C. Resende, G. Veiga and N. Karmarkar, "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming* 44 (1989), 297-335.

[2] K. Anstreicher, "Strict monotonicity and improved complexity in the standard form projective algorithm for linear programming," CORE Discussion Paper 9035, CORE, Université Catholique de Louvain (Louvain-la-Neuve, Belgium), 1990.

[3] K.M. Anstreicher and P. Watteyne, "A family of search directions for Karmarkar's algorithm," CORE Discussion Paper 9030, CORE, Université Catholique de Louvain (Louvain-la-Neuve, Belgium), 1990.

[4] E.R. Barnes, "A variation on Karmarkar's algorithm for solving linear programming problems," *Mathematical Programming* 36 (1986), 174-182.

[5] I.C. Choi, C.L. Monma and D.F. Shanno, "Further development of a primal-dual interior point method," *ORSA Journal on Computing* 2 (1990), 304-311.

[6] I.I. Dikin, "Iterative solution of problems of linear and quadratic programming," *Doklady Academiia Nauk SSSR* 174 (1967), 747-748.

[7] R.M. Freund, "Polynomial-time algorithms for linear programming based only on primal scaling and projected gradients of a potential function," manuscript, Sloan School of Management, M.I.T., Cambridge, Massachusetts, 1988.

[8] D.M. Gay, "Electronic mail distribution of linear programming test problems," *COAL Newsletter* 13 (1985), 10-12.

[9] C. Gonzaga, "Polynomial affine algorithms for linear programming," *Mathematical Programming* 49 (1990), 7-21.

[10] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica* 4 (1984), 373-395.

[11] I.J. Lustig, R.E. Marsten and D.F. Shanno, "Computational experience with a primal-dual interior-point method for linear programming," Technical Report SOR 89-17, Department of Civil Engineering and Operations Research, Princeton University, 1989, to appear in *Linear Algebra and its Applications.*

[12] I.J. Lustig, R.E. Marston and D.F. Shanno, "The primal-dual interior point method on the Cray supercomputer," in: Large-Scale Numerical Optimization, T.F. Coleman and Y. Li, eds., *SIAM*, Philadelphia, 1990, 70-80.

[13] K.A. McShane, C.L. Monma and D.F. Shanno, "An implementation of a primal-dual interior point method for linear programming," *ORSA Journal on Computing* 1 (1989), 70-83.

[14] N. Megiddo and M. Shub, "Boundary behavior of interior point algorithms in linear programming," *Mathematics of Operations Research* 14 (1989), 97-146.

[15] S. Mehrotra, "On the implementation of a (primal-dual) interior point method," Technical Report 90-03, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1990.

[16] C.B. Moler, J. Little, S. Bangert and S. Kleiman, Pro-Matlab User's Guide MathWorks, Sherborn, MA, 1987.

[17] C. Monma and A. Morton, "Computational experience with a dual affine variant of Karmarkar's method for linear programming," *Operations Research Letters* 6 (1987), 261-267.

[18] C. Roos and J.-P. Vial, "A polynomial method of approximate centers for linear programming," Report, Delft University of Technology (1988), to appear in *Mathematical Programming.*

[19] M.J. Todd, "The effects of degeneracy and null and unbounded variables on variants of Karmarkar's linear programming algorithm," in: Large-Scale Numerical Optimization, T.F. Coleman and Y. Li, eds., *SIAM*, Philadelphia, 1990, 81-91.

[20] M.J. Todd, "A low complexity interior point algorithm for linear programming," Technical Report 903, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1990.

[21] M.J. Todd and J.-P. Vial, "Todd's low-complexity algorithm is a predictor-corrector path-following method," Technical Report No. 952, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1990.

[22] M.J. Todd and Y. Wang, "On combined phase 1 - phase 2 projective methods for linear programming," Technical Report No. 877, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1989, to appear in Algorithmica.

[23] T. Tsuchiya, "Global convergence of the affine scaling methods for degenerate linear programming problems," Technical Report, The Institute of Statistical Mathematics, Tokyo, 1990.

[24] D. Xiao and D. Goldfarb, "A path-following projective interior point method for linear programming," Manuscript, Department of IE/OR, Columbia University, New York, 1990.

[25] Y. Ye, "An $O(n^3L)$ potential reduction algorithm for linear programming," manuscript, Department of Management Science, University of Iowa, Iowa City, IA, 1988, to appear in *Mathematical Programming.*

central path (e.g. [18]). Indeed, Todd and Vial [21] recently showed that $\|\bar{d}_\alpha\|$ is exactly the measure of centrality of Roos and Vial [18]. Further, they showed that after some initial centering steps, each affine-scaling step is followed by at most three centering steps before another affine-scaling step. Hence the BLCA generates iterates close to the central path.

What only emerged from computational testing is how astonishingly close the iterates remain. Indeed, after the initial centering phase, no further centering steps were made in several runs on random standard-form problems of dimensions 50×100 up to 300×600, and in fact $\|\bar{d}_\alpha\|$ continued to decrease. This encouraged us to try modifications. Suppose we perform a line search on the logarithmic barrier function along the constant-cost centering direction if $\|\bar{d}_\alpha\| > 9/10$. We take a Newton step in this direction if $\|\bar{d}_\alpha\|$ lies between 1/16 and 9/10. Finally, if $\|\bar{d}_\alpha\|$ is smaller than 1/16, we take a step of length .4 (rather than .2) in the affine-scaling direction. It can be shown [21] that, after the initial centering phase, each affine-scaling step is followed by at most two centering steps; but in practice, a single centering step is only needed every 10-20 iterations.

In Figure 1, we show the trajectories of these algorithms on a 50×100 problem. Figure 1(a) plots the 31st component of x against the 30th, while 1(b) plots the 34th against the 33th. The dashed line corresponds to the BLCA, which requires 347 iterations; every tenth iterate is marked with a "+". The dotted line is the modification described above, needing only 170 iterations; every tenth iterate is marked with a "x." Finally, the solid line gives the progress of the most efficient variant of the BLCA, called variant 2 in [20], with q = 2n; this needs only 11 iterations. All trajectories start at (1,1).

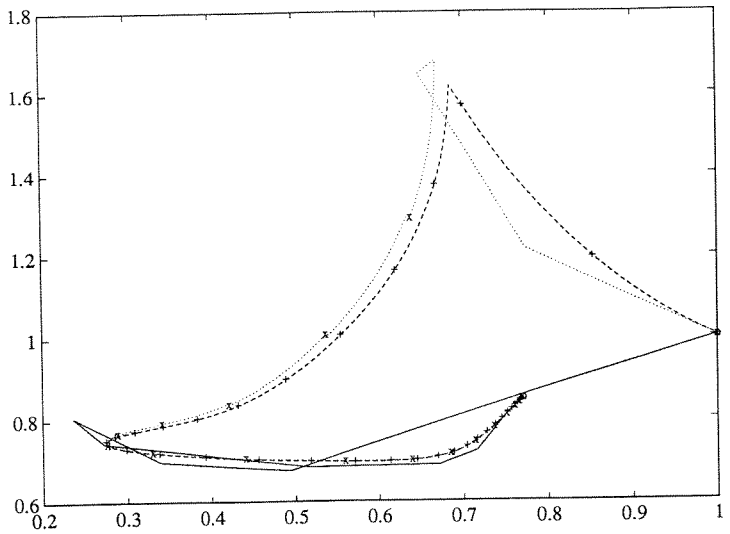

Figure 1(a)

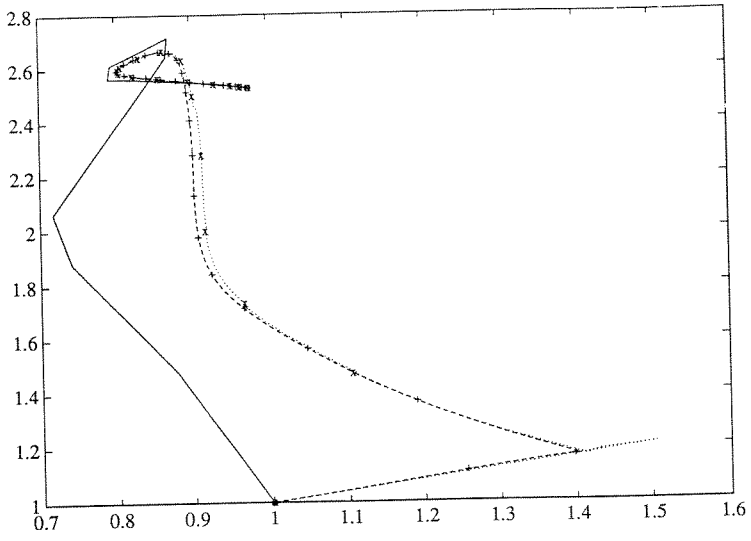

Figure 1(b)

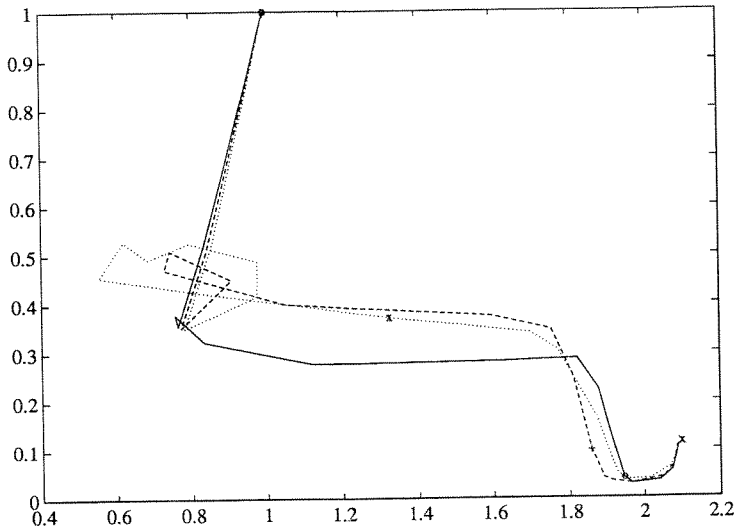

Figure 3(a)

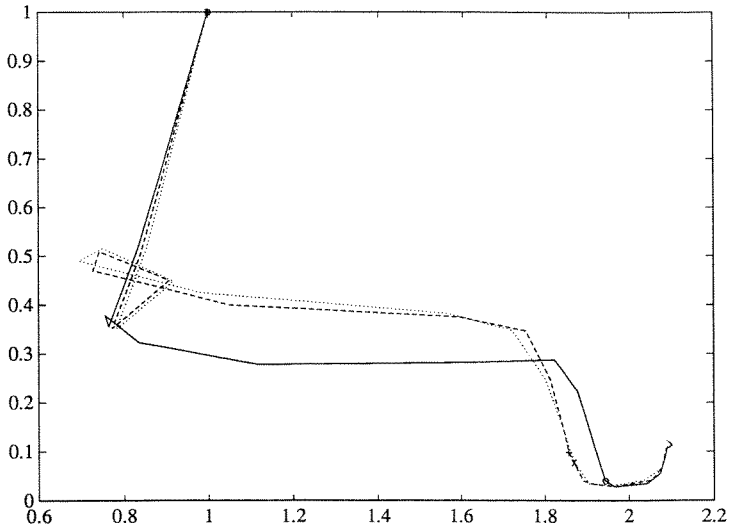

Figure 3(b)

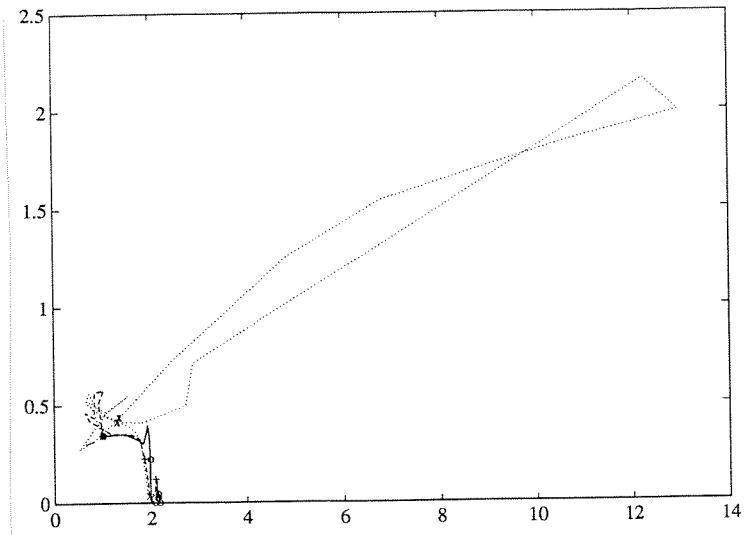

Figure 3(c)

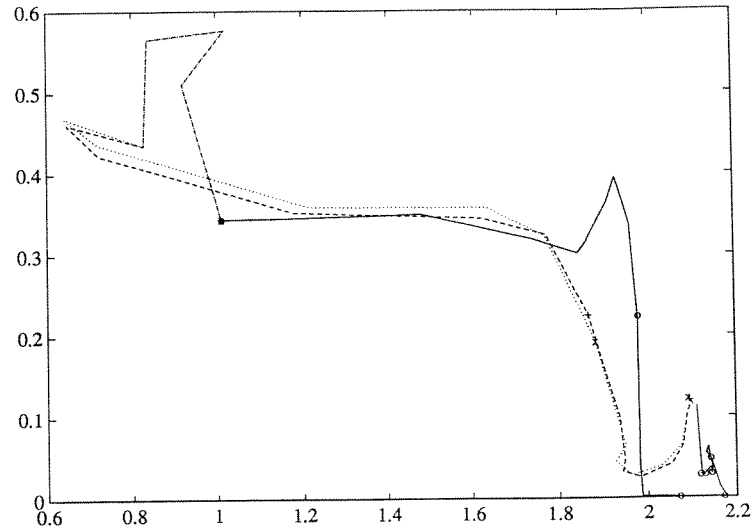

Figure 3(d)

The table below gives the average number of iterations required for five random problems and the three algrithms discussed above. For the affine-scaling algorithm, the average proportion of the way to the boundary was (of course) .95. For variant 2, it ranged from .96 to .99, and for variant 3, from .88 to .99. The advantage of the latter algorithms is that their use of a potential function allows further flexibility: short steps will be taken if necessary, longer steps if safe.

	100×200	200×400	300×600	400×800
affine-scaling	11.8	12.8	13.8	14.4
variant 2	12.2	13.6	13.8	14.4
variant 3	13.6	16.0	18.2	19.2

We see that in an algorithm with a polynomial-time guarantee it generally pays to choose a direction between that of the pure potential reduction algorithm and that of the affine-scaling algorithm, and that the penalty of using variant 2 instead of the latter is very slight in terms of number of iterations. (Unfortunately, variant 2 needs an extra projection at each iteration.)

Finally, all the runs with variant 2, and to a lesser extent, the other methods, show a strong symmetry between the objective function values and the lower bounds generated -- see Figure 2 for a typical example, where the solid line shows the progress of the objective and the dotted line that of the bound; at each iteration, both have roughly the same accuracy.

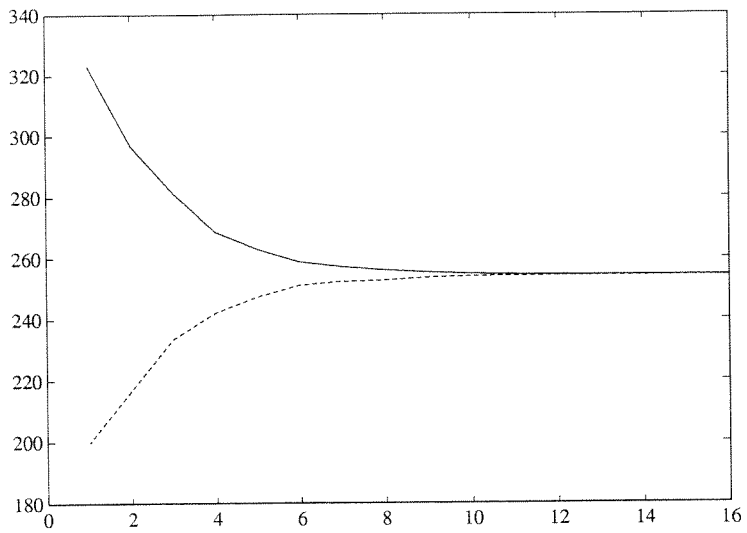

Figure 2.