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

TECHNICAL REPORT NO. 857

August 1989
(Revised June 1990)

THE EFFECTS OF DEGENERACY AND NULL AND UNBOUNDED VARIABLES ON VARIANTS OF KARMARKAR'S LINEAR PROGRAMMING ALGORITHM

By

Michael J. Todd*

*Research supported in part by NSF grants ECS-8602534 and DMS-8904406 and ONR contract N00014-87-K0212. The computation was carried out in the Cornell Computational Optimization Laboratory with support from NSF grant DMS-8706133.

ABSTRACT

We examine the effects of primal and dual degeneracy and the presence of null and unbounded variables on the computational performance of two variants of Karmarkar's primal projective algorithm for linear programming. The methods are applied to randomly generated problems with prescribed degrees of these features. The most striking effect is the poor performance of the algorithms when unbounded variables (variables that can be unbounded in any optimal solution) are present.

Key words: linear programming, interior-point methods, computational testing, random linear programming problems

1. Introduction

Interior-point methods have been much investigated since Karmarkar's seminal paper [8]. Several computational studies [1, 9, 10] have confirmed that the number of iterations is quite low, while the CPU time required is generally less than that for the well-known MINOS code [11] (which implements the simplex method for linear programming); the advantage of interior-point approaches seems to increase with problem size. However, certain features of linear programming problems appear to cause difficulties for interior-point methods. Degeneracies lead to ill-conditioning in the linear systems that must be solved at each iteration, with the condition number increasing as one approaches the optimum. The presence of null variables (variables that are zero in every feasible solution) also can cause difficulties; clearly the original problem has no interior and modifications must be made. Finally, the polynomial-time convergence theory demands that the set of optimal solutions be bounded. While this can be arranged by reformulating problems with integer data, it is preferable to treat problems in their original form, and also problems with real data. Thus the presence of unbounded variables (variables that are unbounded on the set of optimal solutions) is also of interest. (Unbounded variables turn out to be the dual of null variables -- see [14].)

The aim of this paper is to investigate the effect of these features of linear programming problems on the computational behavior of primal projective interior-point methods. We use two algorithms: the standard-form projective variant of Anstreicher [2], Gay [5], Gonzaga [7], Jensen and Steger [12] and Ye and Kojima [15], with an artificial variable with high cost added to achieve feasibility; and the combined phase I - phase II projective algorithm of Anstreicher [3] as modified by Todd [13], which seeks feasibility and optimality simultaneously without needing an explicit cost for the artificial variable. These algorithms are sketched in section 2. The algorithm of de Ghellinck and Vial [4] is also closely related; when presented with a feasible solution it coincides with the methods in [2,5,7,12,15] (which then need no artificial variable), and in general it seeks feasibility and optimality simultaneously as in [3].

The problems are generated randomly using probabilistic models from [14]. They always have optimal solutions, and the degree of primal and dual degeneracy at the optimum can be precisely controlled. In addition, the second model allows the introduction of null and/or unbounded variables. The models are described in section 3.

Section 4 presents the results of numerical testing of the algorithms on problems of size 50×100 up to 300×600. (Since the problems are completely dense, the latter must be regarded as reasonably large.) The results confirm the slow growth of the number of iterations with the size of problem. None of sparsity, primal, or dual degeneracy had a very large effect on the number of iterations, although the presence of both primal and dual degeneracy had a noticeable detrimental influence. Null variables, also, had a relatively small effect. However, the presence of unbounded variables has a very great effect, and we were obliged to relax our convergence criterion significantly in order to solve a reasonable fraction of the problems. Alternatively, giving the algorithms the optimal value of the problems as their initial lower bound rendered the problems innocuous. Note that both algorithms are driven by seeking reductions in a suitable potential function. In problems with unbounded variables, the potential function can be driven to minus infinity either by approaching optimality or by holding the objective function bounded above and all variables bounded away from zero and increasing the unbounded variables without limit. In some cases the algorithms exercise the latter option.

2. The algorithms

Both algorithms are designed to solve problems in standard form

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

where A is m×n, and both introduce a homogenizing variable σ and an artificial variable τ. Let

$$\begin{aligned}
\tilde{x}^T &= (x^T, \sigma, \tau) \\
\tilde{A} &= [A, -b, b - Ae] \\
\tilde{c}^T &= (c^T, 0, 0) \\
\tilde{d}^T &= (0^T, 1, 0), \quad \text{and} \\
\tilde{\xi}^T &= (0^T, 0, 1),
\end{aligned}$$

where e denotes a vector of ones of appropriate dimension.

The standard-form variant of [2,5,7,12,15], with a high cost ν associated with the artificial variable τ, will be called algorithm A. It solves the problem

$$(P_A) \qquad \begin{aligned} \text{minimize} \quad & (\tilde{c} + \nu\tilde{\xi})^T\tilde{x} \\ \text{subject to} \quad & \tilde{A}\tilde{x} = 0, \\ & \tilde{d}^T\tilde{x} = 1, \\ & \tilde{x} \geq 0, \end{aligned}$$

and generates a sequence $\{\tilde{x}^k\}$ of strictly positive feasible solutions with $\tilde{x}^0 = e$ and a sequence $\{z^k\}$ of lower bounds to the optimal value of (P_A). Given the kth iterate $\tilde{x}^k$ and the kth lower bound z^k, we proceed as follows.

First let $\tilde{X}_k = \text{diag}(\tilde{x}^k)$ be the diagonal matrix with the components of $\tilde{x}^k$ down its diagonal, and define

$$\begin{aligned}
\bar{A} &= \tilde{A}\tilde{X}_k, \\
\bar{c} &= \tilde{X}_k(\tilde{c} + \nu\tilde{\xi}), \quad \text{and} \\
\bar{d} &= \tilde{X}_k\tilde{d}.
\end{aligned}$$

Then, in terms of the scaled variables $\bar{x} = \tilde{X}_k^{-1}\tilde{x}$, (P_A) becomes

$$(\bar{P}_A) \qquad \begin{aligned} &\text{minimize} && \bar{c}^T\bar{x} \\ &\text{subject to} && \bar{A}\bar{x} = 0, \\ & && \bar{d}^T\bar{x} = 1, \\ & && \bar{x} \geq 0, \end{aligned}$$

and the current iterate $\tilde{x}^k$ becomes $\bar{x} = e$. For any vector $\bar{v}$, let $\bar{v}_p$ denote its projection into the null space of $\bar{A}$. If $(\bar{c} - z\bar{d})_p \geq 0$ for some z, it is easy to see that z is a valid lower bound for $(\bar{P}_A)$ and hence (P_A); indeed, it is part of a feasible dual solution. Hence $z^{k+1} := \bar{z} := \max\{z^k, \max\{z: \bar{c}_p - z\bar{d}_p \geq 0\}\}$ is a new valid lower bound. We then update $\bar{x}$ to $\bar{x}_+ = e - \beta(\bar{c} - \bar{z}\bar{d})_p$ for some $\beta > 0$, normalize it so that $\bar{d}^T\bar{x} = 1$, and then scale back to the original space to get

$$\tilde{x}^{k+1} = \tilde{X}_k\bar{x}_+/\bar{d}^T\bar{x}_+.$$

Here β is chosen to (approximately) minimize the potential function

$$\bar{f}(\bar{x}; \bar{c} - \bar{z}\bar{d}) := \sum_j \ell n \frac{(\bar{c} - \bar{z}\bar{d})^T\bar{x}}{\bar{x}_j}.$$

In fact, we use a monotonic variation due to Anstreicher [2], which ensures that $(\tilde{c} + \nu\tilde{\xi})^T\tilde{x}^{k+1} \leq (\tilde{c} + \nu\tilde{\xi})^T\tilde{x}^k$. The algorithm is terminated when the duality gap and the artificial variable are sufficiently small.

The second method, a modification [13] of Anstreicher's combined phase I-phase II method [3], will be called algorithm B. It handles the artificial variable τ more directly by solving the problem

$$
(P_B) \qquad \begin{array}{ll} \text{minimize} & \tilde{c}^T\tilde{x} \\ \text{subject to} & \tilde{A}\tilde{x} = 0, \\ & \tilde{d}^T\tilde{x} = 1, \\ & \tilde{\xi}^T\tilde{x} = 0, \\ & \tilde{x} \geq 0, \end{array}
$$

generating a sequence $\{\tilde{x}^k\}$ of strictly positive iterates satisfying all constraints except $\tilde{\xi}^T\tilde{x} = 0$, with $\tilde{x}^0 = e$, and a sequence $\{z^k\}$ of lower bounds.

Given $\tilde{x}^k$, define the scaled data as above, except that $\bar{c} = \tilde{X}_k\tilde{c}$ (there is no explicit high cost ν on the artificial variable). Then $z^{k+1} = \bar{z}$ is set to the maximum of z^k and the optimal value of the relaxed scaled problem

$$
\begin{array}{ll} \text{minimize} & \bar{c}_p^T\bar{x} \\ \text{subject to} & \bar{d}_p^T\bar{x} = 1, \\ & \bar{\xi}_p^T\bar{x} = 0, \\ & \bar{x} \geq 0. \end{array}
$$

Again, this lower bound corresponds to a feasible dual solution. The search direction $\bar{g}$ in the transformed space is found by solving a direction-finding subproblem, which is equivalent to minimizing the angle between e and $e + \bar{g}$ subject to $(\bar{c} - \bar{z}\bar{d})_p^T(e + \bar{g}) \leq 0$ and $\bar{\xi}_p^T(e + \bar{g}) \leq 0$. (The resulting $\bar{g}$ is proportional to that which would arise from algorithm A with an implicit iteration-dependent value for the high artificial cost ν, related to the Lagrange multipliers in the direction finding subproblem.) The constraint $\bar{A}(e + \bar{g}) = 0$ is automatically satisfied, and if $e + \bar{g}$ were nonnegative, it would be (after normalizing) optimal in the scaled problem, since it would achieve the lower bound z^{k+1} and feasibility. Usually, $e + \bar{g} \not\geq 0$, and so a search is made in the direction $\bar{g}$ to minimize $\bar{f}(\cdot; \bar{c} - \bar{z}\bar{d})$ or $\bar{f}(\cdot; \bar{\xi})$. Again, a slight modification is made to ensure monotonicity. The stopping criterion is as in algorithm A.

The probabilistic models discussed in the next section produce well-scaled problems, so it is appropriate to use $\tilde{x} = e$ as an initial starting solution. The optimal value is usually at most 5m, so we chose the cost ν of the artificial variable in algorithm A as 10^{10}. Both algorithms can automatically produce lower bounds. However, we only used this capability in algorithm B. (When $z^{k+1} = -\infty$, use $\bar{d}$ instead of $\bar{c} - z^{k+1}\bar{d}$ in the direction-finding subproblem.) In algorithm A, we initialized with $z^0 = -10^5$. This has a very attractive consequence. When the artificial variable τ is large (of order 1) so that the objective function is of order 10^{10}, the lower bound is effectively zero, appropriate for reducing τ. When τ is so small that its effect on the objective function is minimal, the lower bound, at -10^5, is suitably negative for well-scaled problems with optimal value at least an order of magnitude smaller in absolute value.

3. Probabilistic models

Both models generate an $m \times n$ matrix A, nonnegative n-vectors $\hat{x}$ and $\hat{s}$ with $\hat{x}^T\hat{s} = 0$, and an m-vector $\hat{y}$, and then set $b = A\hat{x}$ and $c = A^T\hat{y} + \hat{s}$. Thus, by complementary slackness, $\hat{x}$ is optimal in (P) and $\hat{y}$, with slack vector $\hat{s}$, in its dual. The models differ primarily in the form of A. We can control the degree of primal and dual degeneracy at the optimal solution by specifying how many components of $\hat{x}$ or $\hat{s}$ are positive. This scheme usually produces degeneracy only at the optimal solutions [14], but interior-point methods started well in the interior of the feasible region are expected only to approach the boundary closely near the optimal vertex (or points in the optimal face).

Model 1 has no null or unbounded variables. For a dense problem, we choose each entry of A independently from a standard Gaussian distribution. (If we desired a sparse problem with density approximately α, we could generate a dense matrix A as above and then set entries of A independently to zero with probability $1 - \alpha$, and repeat the process if the result has any zero rows or columns. This random sparsity appears very different from the structured sparsity of real problems, but is chosen for lack of any convincing alternative. We made some runs with $\alpha = .2$, but the results

seemed comparable to those with $\alpha = 1$, and we concluded that this form of sparsity has little or no effect on the algorithms.) Besides m and n, two other parameters m_1 and m_2, with $0 < m_1 \leq m \leq m_2 < n$, determine the distribution. We set the first m_1 components of $\hat{x}$ positive, and the last $n - m_2$ components of $\hat{s}$ positive, with the rest zero. The positive components are chosen independently as absolute values of standard Gaussian random variables. The algorithms used are independent of $\hat{y}$ (where $c = A^T\hat{y} + \hat{s}$), except for a very mild dependence in the termination criterion. We choose each component of $\hat{y}$ independently from a standard Gaussian distribution.

We generated 10 replications from each setting of the parameters. We considered 20 settings of the parameters. The pair (m,n) was one of the pairs (50, 100), (100, 200), (150, 300), (200, 400) or (300, 600). For each (m,n) we took m_1 either m (primal nondegenerate) or $\frac{1}{2}$ m (primal degenerate) and m_2 either m (dual nondegenerate) or $\frac{3}{2}$ m (dual degenerate).

Model 2 has null and/or unbounded variables. Assume first we wish to have both null and unbounded variables, and that $n = 100k$ (again, n will be 100, 200, 300, 400 or 600 in our tests). Then the first 25k variables will be unbounded and the last 25k null. Let the entries of

$$[\bar{A}_1, A_2, \bar{A}_3]$$

be independently drawn from a Gaussian distribution, where $\bar{A}_1$ and $\bar{A}_3$ are (m-1) × 25k, and A_2 (m-1) × 50k. Let

$$A_j = \bar{A}_j\left(I - \frac{ee^T}{25k}\right), \quad j = 1,3,$$

so that the rows of $\bar{A}_j$ are projected orthogonal to the vector e of ones, here of dimension 25k. Then $A_1e = 0$, and similarly $A_3e = 0$. Finally, let

$$A = \begin{bmatrix} 0^T & 0^T & e^T \\ A_1 & A_2 & A_3 \end{bmatrix}.$$

Now let $\hat{x}$ be generated as above, except that its positive components will be either the 2nd through (50k)th (primal almost nondegenerate) or the (12k+1)st through (37k)th (primal degenerate). Similarly, the positive components of $\hat{s}$ are either its (50k+1)st through (n - 1)st, or its (63k+1)st through (88k)th. This choice is so that there is some overlap between positive variables in an optimal basis and unbounded variables, and similarly in the dual. We choose $\hat{y} = 0$ so that the optimal value is zero. An optimal basis consists of the 2nd through (50k)th column of A and its last column. (We need one of the last columns to avoid a zero row in the basis matrix.)

Note that the first entry of $b = A\hat{x}$ is zero, so the first constraint forces the last 25k components of any feasible solution to be zero; hence these are null variables. Also, the first 25k components of c are zero, while $A_1 e = 0$, so that adding a constant to all of the first components of any feasible solution x leaves it feasible with the same objective function value. Hence the first 25k variables are unbounded variables; they are unbounded in the set of optimal solutions.

If we want null but not unbounded variables we proceed as above but use $\bar{A}_1$ instead of A_1. If we want unbounded but not null variables, we proceed as above but generating matrices with m, not m - 1 rows, use $\bar{A}_3$ instead of A_3, and omit the row of 0's and 1's in A. In all cases we generate $\hat{x}$ and $\hat{s}$ as above.

For model 2 we chose (m,n) as in model 1. Sometimes we generated all combinations of primal and dual degeneracy, but often we only considered the (almost) nondegenerate problems.

Theoretical properties of these two models are discussed further in [14].

4. The results

Here we describe the results of our computational testing of algorithms A and B on models 1 and 2. Both methods were coded in FORTRAN using double precision. The projections at each iteration were performed by obtaining a QR factorization of the scaled matrix $\bar{A}^T$ and using the orthogonal matrix Q (see, e.g., [6]). Most of the ill-conditioning is caused by different scales for different rows of $\bar{A}^T$. We therefore ordered the rows to correspond to decreasing components of $\tilde{x}^k$. Furthermore, the result of each iteration was projected again in order to help preserve accuracy. Successful termination took place when

$$\max\left\{10\tau,\ \frac{|\tilde{c}^T\tilde{x}^k - z^k|}{\max\{1,\ |\tilde{c}^T\tilde{x}^k|\}}\right\} \leq \epsilon,$$

where $\epsilon > 0$ is the objective function tolerance. (Recall that τ is the artificial variable.) We also terminated (unsuccessfully) if the potential function could not be reduced during an iteration or if accuracy was lost: for some i

$$\max\left\{.95\ \frac{|(\tilde{A}\tilde{x}^k)_i|}{\max_j\ |\tilde{a}_{ij}|},\ 10\tau,\ \frac{|\tilde{c}^T\tilde{x}^k - z^k|}{\max\{1,|\tilde{c}^T\tilde{x}^k|\}}\right\} = .95\ \frac{|(\tilde{A}\tilde{x}^k)_i|}{\max_j\ |\tilde{a}_{ij}|} > \epsilon.$$

All runs were made on a Sun SPARCstation 1.

For model 1 we set ϵ to 10^{-7}. Almost all the problems were successfully solved by both algorithms; for a very few, we could only achieve accuracy 10^{-6}. The average number of iterations required (recall that ten problems were generated and solved for each setting of the parameters) is given in Table 1. The results indicate a curious effect of degeneracy. Dual degeneracy alone appears to help both algorithms, while primal degeneracy had a slightly negative effect on algorithm A and a positive one on algorithm B; but their combined effect is strongly detrimental. We therefore performed a regression to determine a least-squares fit to the data, including an interaction term for primal and dual degeneracy.

Results for Model 1

m	n	pd	dd	Average no. of iterations Algorithm A	Average no. of iterations Algorithm B
50	100	0	0	23.6	22.7
50	100	1	0	23.7	21.0
50	100	0	1	20.4	21.9
50	100	1	1	24.5	24.5
100	200	0	0	24.5	25.3
100	200	1	0	24.4	21.8
100	200	0	1	23.3	24.3
100	200	1	1	31.7	30.6
150	300	0	0	26.5	27.8
150	300	1	0	27.5	22.7
150	300	0	1	21.6	22.2
150	300	1	1	32.9	29.2
200	400	0	0	28.5	28.0
200	400	1	0	28.7	23.8
200	400	0	1	23.6	23.8
200	400	1	1	38.5	32.3
300	600	0	0	29.8	28.4
300	600	1	0	30.9	27.3
300	600	0	1	24.8	26.5
300	600	1	1	40.8	36.4

pd is 1 if primal degenerate, 0 otherwise
dd is 1 if dual degenerate, 0 otherwise

Table 1

For algorithm A, including all regressors, the fit is

$$\underset{(1.32)}{it = 21.18} + \underset{(.0056)}{.0337m} + \underset{(1.36)}{.46pd} - \underset{(1.36)}{3.84dd} + \underset{(1.93)}{10.48pdd}$$

where the standard error is shown below each estimate, "it" is the average number of iterations and pdd = pd.dd is 1 iff both primal and dual degeneracy are present. The p-values for the hypotheses

that the parameters are zero are .01% for the intercept, (the coefficient of) m, and pdd, 1.3% for dd, and 74.03% for pd. If we omit the least significant regressor pd, we get the fit

$$\begin{array}{c} \text{it} = 21.41 + .0337\text{m} - 4.07\text{dd} + 10.94\ \text{pdd} \\ (1.09)\quad (.0054)\quad (1.15)\qquad (1.32) \end{array}$$

with p-values .01% for the intercept, m and pdd and .27% for dd. The value of R^2 decreased from .873 with all regressors to .872, so the fit remains very good.

For algorithm B, the fit with all regressors is

$$\begin{array}{c} \text{it} = 22.26 + .0261\text{m} - 3.12\text{pd} - 2.70\text{dd} + 9.98\text{pdd}. \\ (.94)\quad (.0040)\quad (.98)\quad (.98)\qquad (1.38) \end{array}$$

The p-values for the hypotheses that the parameters are zero are .01% for the intercept, (the coefficient of) m and pdd, .6% for pd, and 1.4% for dd. The value of R^2 is .883, so again the fit is very good.

Overall, the results show a mild increase in the number of iterations with dimension (about 3 extra iterations with an increase of 100 in the number of rows, although it would be dangerous to extrapolate far) and a reasonable decrease when dual degeneracy is present (the optimal solution set is larger, and easier to approach from the interior). Algorithm B deals with primal degeneracy alone better, perhaps because it treats infeasibilities and optimality more symmetrically. However, why primal degeneracy should help algorithm B is a mystery, as is the strongly detrimental effect of the simultaneous presence of both degeneracies.

As we remarked in Section 2, Algorithm 2 can be viewed as making an implicit choice for the cost ν of the artificial variable at each iteration. This choice is determined adaptively, and is usually much smaller than the fixed value 10^{10} chosen in algorithm A. For example, in the 200×400 problems, the equivalent value for ν ranged from 0 to 620, except for fewer than 2% of the iterations, when it was effectively infinite.

Now we turn to model 2 and examine first problems with both null and unbounded variables. Here it was found almost impossible to solve the problems with $\epsilon = 10^{-7}$. The unbounded variables became very large, the linear systems became very ill-conditioned, and the algorithms terminated with inaccurate solutions or with an inability to reduce the potential function. We therefore relaxed ϵ to 2×10^{-2}. In this case algorithm B performed adequately, solving 29 (out of 40) of the 50×100 problems, 27 of the 100×200 problems, 19 of the 150×300 problems, 17 of the 200×400 problems, and 10 of the 300×600 problems. However, algorithm A had great difficulty. We applied it to just the nondegenerate problems, on which it solved 4 (out of 10) of the 50×100 problems but none of the larger problems. The data are given in Table 2.

In order to determine the source of the difficulty, more runs were made. Here I shall give the results only for the nondegenerate problems. If unbounded variables were present but no null variables, the situation was similar. Algorithm A solved 4 of the 50×100 problems, 4 of the 100×200, and none of the larger problems. Algorithm B solved 9 of the smallest problems, 1 of the 100×200, 5 of the 150×300, 6 of the 200×400 problems and 2 of the 300×600. All these results were again with $\epsilon = 2\times10^{-2}$. Suppose next that there are null variables but no unbounded variables. Then neither algorithm had difficulties, even with $\epsilon = 10^{-7}$. The average number of iterations was 36.5, 38.4, 39.6, 40.6 and 42.2 for algorithm A, and 22.0, 22.9, 24.5, 25.6 and 26.6 for algorithm B, all averaged over the ten nondegenerate problems of each size from 50×100 up to 300×600 respectively. Thus, as with primal degeneracy, algorithm B copes much better with null variables. However, null variables create no hard problems for either method; the presence of unbounded variables is very detrimental to both methods, whether or not null variables are introduced. Algorithm B has a significant advantage in robustness.

Incidentally, the results for model 2 should be taken with a large grain of salt. Just changing the optimizer option (the runs quoted used Sun's f77 compiler with optimization option 03) made significant differences in the results, although the overall pattern was similar. The most sensitive part of the codes appears to be the generation of the lower bounds in the presence of ill-conditioning.

				Algorithm A			Algorithm B		
m	n	pd	dd	No. of problems solved (out of 10)	Average no. of iterations (successful)	Average no. of iterations (all)	No. of problems solved (out of 10)	Average no. of iterations (successful)	Average no. of iterations (all)
50	100	0	0	4	44.0	46.2	9	27.1	27.5
50	100	1	0	-	-	-	10	26.9	26.9
50	100	0	1	-	-	-	4	26.5	29.6
50	100	1	1	-	-	-	6	27.5	28.7
100	200	0	0	0	-	42.8	5	28.2	29.8
100	200	1	0	-	-	-	9	28.1	29.4
100	200	0	1	-	-	-	8	28.4	29.0
100	200	1	1	-	-	-	5	28.0	29.6
150	300	0	0	0	-	45.9	2	30.5	31.7
150	300	1	0	-	-	-	5	29.6	31.7
150	300	0	1	-	-	-	8	29.7	30.1
150	300	1	1	-	-	-	4	29.0	30.6
200	400	0	0	0	-	45.8	6	30.8	31.6
200	400	1	0	-	-	-	1	31.0	32.5
200	400	0	1	-	-	-	6	30.0	30.9
200	400	1	1	-	-	-	4	31.0	30.9
300	600	0	0	0	-	44.1	1	31.0	30.7
300	600	1	0	-	-	-	3	31.0	34.2
300	600	0	1	-	-	-	4	30.5	30.9
300	600	1	0	-	-	-	2	30.5	31.2

Table 2

Finally, it was noticed that all the problems with unbounded variables had great difficulty in obtaining lower bounds. Kurt Anstreicher (private communication) has pointed out that this is to be expected; such problems have duals whose feasible regions have no interiors, and obtaining or updating lower bounds requires feasible dual solutions. However, as the unbounded variables increase, they force the dual solutions considered toward the appropriate affine subspace, so that with an appropriate small tolerance for dual feasibility, lower bounds can be found and updated; this accounts for our modest

success in solving these problems. Frequently, the unbounded variables had increased so far before a finite lower bound was generated (typically, to around 10^{10}, with a comparable condition number for the upper triangular matrix R) that it was very hard for the algorithms to recover and/or maintain accuracy. Hence we resolved the problems with both null and unbounded variables where we set the initial lower bound z^0 to the optimal value, which was zero for all these problems. This alleviated all the difficulties, and all problems were successfully solved with $\epsilon = 10^{-7}$. The average number of iterations was 32.7, 33.9, 35.0, 35.5 and 36.7 for algorithm A, and 17.3, 18.1, 19.4, 20.7 and 20.9 for algorithm B, all averaged again over the ten nondegenerate problems of each size from 50×100 to 300×600 respectively. We see again a significant advantage for algorithm B here, but also the value of having a good (here perfect) lower bound for the objective function value.

We also used our code to try the algorithms of Anstreicher [3], on which algorithm B was based. The original choice of direction in [3] turned out to be very poor. All ten nondegenerate 50×100 problems generated by model 1 were solved, but only to an accuracy of 10^{-3} to 10^{-4}, and the average number of iterations was 103.1. When the size was increased to 100×200, two problems reached the iteration limit of 199 and eight were solved to a similar accuracy; the average number of iterations grew to 155.1. None of the nondegenerate 50×100 problems generated by model 2 with null and unbounded variables were solved; all hit the iteration limit of 199. However, when we implemented the improved direction described at the end of section 4 of [3] when appropriate (this is similar to one case of the direction choice in algorithm B) the method improved to be roughly comparable to algorithm B. For the nondegenerate problems generated by model 1, the average number of iterations required was 22.7, 28.0, 32.0, 31.7 and 30.5 for the five sizes of problem (compared to 22.7, 25.3, 27.8, 28.0 and 28.4 for algorithm B). For the nondegenerate problems generated by model 2 with null and unbounded variables, the algorithm successfully solved 9 of the 10 50×100, 6 of the 100×200 problems, 3 of the 150×300, 4 of the 200×400 problems and 1 of the 300×600 problems. The number of iterations required ranged from 25 to 35. Overall, the expanded

logic in the direction-finding technique in algorithm B appears to be worthwhile, particularly for larger problems.

Acknowledgement

I would like to thank Jorge Vera and Amy Riordan with help in coding the algorithms and in analyzing the results.

References

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

2. K.M. Anstreicher, "A monotonic projective algorithm for fractional linear programming," Algorithmica, 1 (1986), pp. 483-498.

3. K.M. Anstreicher, "A combined phase I - phase II projective algorithm for linear programming," Mathematical Programming, 43 (1989), pp. 209-223.

4. G. de Ghellinck and J.-Ph. Vial, "A polynomial Newton method for linear programming," Algorithmica, 1 (1986), pp. 425-453.

5. D.M. Gay, "A variant of Karmarkar's linear programming algorithm for problems in standard form," Mathematical Programming, 37 (1987), pp. 81-90.

6. G.H. Golub and C. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, MD, 1983.

7. C.C. Gonzaga, "Conical projection algorithms for linear programming," Mathematical Programming, 43 (1989), pp. 151-173.

8. N.K. Karmarkar, "A new polynomial time algorithm for linear programming," Combinatorica, 4 (1987), pp. 373-395.

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

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

11. B.A. Murtagh and M.A. Saunders, "MINOS 5.1 Users Guide," Technical Report SOL83-20R, Department of Operations Research, Stanford University, Stanford, CA, 1987.

12. A. Steger, "An extension of Karmarkar's algorithm for bounded linear programming problems," M.S. Thesis, SUNY at Stonybrook, New York, 1985.

13. M.J. Todd, "On Anstreicher's combined phase I-phase II projective algorithm for linear programming," Technical Report 776, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1989, to appear in Mathematical Programming.

14. M.J. Todd, "Probabilistic models for linear programming," Technical Report 836, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, 1989, to appear in Mathematics of Operations Research.

15. Y. Ye and M. Kojima, "Recovering optimal dual solutions in Karmarkar's polynomial algorithm for linear programming," Mathematical Programming, 39 (1987), pp. 305-317.