Mathematical Programming 43 (1989) 31-44
North-Holland

AFFINE-SCALING FOR LINEAR PROGRAMS WITH FREE VARIABLES

R.J. VANDERBEI
AT&T Bell Laboratories, Murray Hill, NJ 07974, USA

Received 20 January 1987
Revised manuscript received 10 October 1987

The affine-scaling modification of Karmarkar's algorithm is extended to solve problems with free variables. This extended primal algorithm is used to prove two important results. First the geometrically elegant feasibility algorithm proposed by Chandru and Kochar is the same algorithm as the one obtained by appending a single column of residuals to the constraint matrix. Second the dual algorithm as first described by Adler et al., is the same as the extended primal algorithm applied to the dual.

Key words: Affine-scaling, Karmarkar's algorithm, dual algorithm, feasibility algorithm.

1. Introduction

The affine-scaling variant of Karmarkar's algorithm [2-7] solves linear programming (LP) problems formulated with equality constraints:

$$\begin{aligned} (\mathrm{P}) \qquad & \min \quad c \cdot x \\ & \text{subject to} \quad Ax = b, \\ & \qquad\qquad 0 \leq x \leq u, \end{aligned}$$

starting from a strictly interior point ξ: $0 < \xi < u$, $A\xi = b$. The matrix A is an $m \times n$ matrix, which we assume has full row rank.

In this paper, we extend the affine-scaling algorithm to solve problems with *free variables* (i.e. not bounded above or below). We use this free variable algorithm (derived in Section 2) to establish two important results. First, the feasibility algorithm proposed by Chandru and Kochar [6] is the same as the one obtained by appending a single column to the constraint matrix. Second, the dual algorithm described by Adler et al. in [1] is the same as the primal algorithm applied to the dual. The free variable algorithm is not itself of practical interest, since there are well known techniques for handling free variables such as splitting them into the difference between their positive and negative parts.

We begin by discussing the feasibility problem: how do we find the interior starting point ξ? For those familiar with the usual tricks of linear programming, this question has an easy answer involving adding one column to the constraint matrix A and trying to drive the corresponding artificial variable to zero. This results in the

so-called *phase 1* (or, *feasibility*) algorithm, which has been described in many papers (see, e.g. [7]). However, the column just added is generally a dense column and this wreaks havoc with the sparse matrix inversion that is the heart of the affine-scaling algorithm. The trick to avoid this problem is to invert the matrix making believe that this column is not present and then do a rank-one update to fix things up.

There is another phase 1 algorithm that also looks attractive. It was first described by Chandru and Kochar in [6]. Its derivation is based on simple geometric ideas as well as the idea of affine-scaling. The first important result of this paper is that these two algorithms are one and the same. This is proved by algebraically carrying out the rank-one update mentioned above and performing algebraic simplifications until one recognizes that the first feasibility algorithm is actually the same as the second. The proof works by formulating the first approach as an LP with free variables (the artificial variable is free) and then applying (in Section 3) the general algorithm for free variables that we derive in Section 2.

Our second result is that the dual-affine-scaling algorithm as described by Adler et al., in [1] is the same algorithm as the primal algorithm applied to the dual problem. The dual of (P) is an LP with free variables (and inequality constraints) and so we must apply the primal algorithm that we derive in Section 2 for problems with free variables.

In [1], the dual algorithm was derived from scratch using the affine-scaling idea but not relating it directly to the primal algorithm. The importance of the equivalence established here is that observations about the primal algorithm can be translated into observations about the dual algorithm. For example, it has been observed [8] that for primal degenerate problems, the primal algorithm might not generate feasible dual variables as it converges to an optimal vertex. Hence, as an immediate corollary of our equivalence, we see that the dual algorithm might fail to generate feasible primal variables if the primal problem is dual degenerate.

We end this introductory section with a brief review of the primal affine-scaling algorithm. Given an interior feasible point x (i.e., $0 < x < u$ and $Ax = b$), the algorithm proceeds as follows:

(B.1) Calculate slacks for the upper bounds

$$y = u - x.$$

(B.2) Calculate the *dual variables*

$$w = (AD_x^2 A^T)^{-1} AD_x^2 c,$$

where[1] $D_x = \operatorname{diag}(x \wedge y)$. The invertibility of $AD_x^2 A^T$ follows from our assumptions that A has full row rank and $0 < x < u$.

[1] If x and y are vectors, we denote by $x \wedge y$ the vector with components equal to the minimum of the corresponding components of x and y: i.e., $(x \wedge y)_i = \min(x_i, y_i)$. Similarly, $x \vee y$ denotes the componentwise maximum.

(B.3) Calculate the *step direction*

$$z = D_x^2(c - A^{\mathrm{T}}w).$$

(B.4) Calculate the *complementary slackness*[1]

$$\gamma = \max\left(\frac{z}{x} \vee -\frac{z}{y}\right).$$

(B.5) Calculate the *dual feasibility*

$$\delta = -\min\left(\frac{z}{x^2} \wedge -\frac{z}{y^2}\right).$$

(B.6) Check the stopping rule: if

$$\gamma + M\delta < \frac{\varepsilon}{n}(|c \cdot x| + 1),$$

where $M = \max(x \wedge y)$, then stop. The current solution x is *ε-optimal* in the sense that

$$\frac{c \cdot x - c \cdot x^*}{|c \cdot x| + 1} < \varepsilon$$

where x^* denotes an optimal point. (This is actually an approximation. See [7] for a rigorous development.)

(B.7) Replace x with a new interior feasible point

$$x \leftarrow x - \frac{\alpha}{\gamma} z,$$

where $0 < \alpha < 1$ is fixed, and go back to (B.1).

A few remarks are in order. First, the affine-scaling algorithm, as discussed in [1-7], treats LP formulations where the variables do not have upper bounds. The algorithm presented here, however, does handle upper bounds. The version with upper bounds is derived as follows. First, the upper bound constraints are included in the collection of equality constraints by introducing slack variables: $x + y = u$ $(x, y \geqslant 0)$. This inclusion results in an enlarged problem:

$$\begin{aligned} \text{min} \quad & \begin{bmatrix} c \\ 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}, \\ \text{subject to} \quad & \begin{bmatrix} A & 0 \\ I & I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ u \end{bmatrix}, \\ & \begin{bmatrix} x \\ y \end{bmatrix} \geqslant 0. \end{aligned}$$

[1] If x and y are vectors, we let x/y denote the vector consisting of the ratios x_i/y_i. If x is a vector, let $\max(x)$ denote the maximum of the elements of x.

The algorithm described in [7] is then applied to this problem. Performing appropriate block matrix algebra, the algorithm is expressed in terms of the matrix AD^*A^T where D^* is the diagonal matrix given by

$$D_{ii}^* = \frac{x_i^2 y_i^2}{x_i^2 + y_i^2}.$$

Finally, we replace $xy/\sqrt{x^2+y^2}$ by $\min(x, y)$ since, as x tends to either zero or its upper bound, these two expressions both approach zero at the same rate. Also, the second expression is simpler.

The *primal objective value* is $c \cdot x$ and the corresponding *dual objective value* is $w \cdot b + (z/y^2) \cdot u$. Note that as u tends to infinity, the dual objective value tends to $w \cdot b$.

There are two minor modifications of problem (P) that we should briefly discuss. First, if the problem is a maximization instead of a minimization, then the only change to the algorithm is that the step direction z is the negative of what is given in (B.3). The second modification is to let the variables have non-zero lower bounds l. In this case, the x's in the formulas for D_x, γ, δ, and M are replaced by $x - l$.

In the next section, we consider LP's like (P) except that some of the variables are free. We derive an algorithm for this type of LP problem by placing upper and lower bounds on the free variables and letting these bounds go to infinity. Then, in each subsequent section, we apply this free-variable algorithm to specific problems in which free variables arise naturally. In particular, in Section 3 we consider the feasibility problem for (P) and, in Section 4, we study the dual of (P). Then, in Section 5, we derive a general algorithm for the feasibility problem associated with problems having free variables. This algorithm is then applied, in Section 6, to the derivation of an algorithm for the feasibility problem for the dual of (P).

2. Free variables

In this section, we extend the affine-scaling algorithm to solve problems having free variables. Before starting, we record for future reference two important matrix identities well known in the folklore of mathematics:

$$\text{(I.1)} \qquad AB(C^{-1} + B^T AB)^{-1} = (A^{-1} + BCB^T)^{-1} BC,$$

$$\text{(I.2)} \qquad (C^{-1} + B^T AB)^{-1} = C - CB^T (A^{-1} + BCB^T)^{-1} BC.$$

In these formulas, A, B, and C are matrices with compatible shapes and the inverted matrices are assumed to be square and non-singular. These formulas are verified as follows. The first is checked by cross multiplying the inverted expressions, and the second is checked by multiplying by $C^{-1} + B^T AB$.

In this section, we will derive an algorithm for solving the following *free-variable* problem:

$$\text{(F)} \qquad \min \quad \begin{bmatrix} c_A \\ c_F \end{bmatrix} \cdot \begin{bmatrix} x_A \\ x_F \end{bmatrix},$$

$$\text{subject to} \quad [A \;\; F] \begin{bmatrix} x_A \\ x_F \end{bmatrix} = b,$$

$$0 \leq x_A \leq u.$$

We see that the components of x_F are free variables (that is, unbounded). In order to derive an algorithm for this problem we will first assume that x_F has finite upper and lower bounds, and then let these bounds go to infinity.

Fix $r > 0$ and suppose that $-r\mathbf{1} < x_F < r\mathbf{1}$, where $\mathbf{1}$ denotes the vector of all ones. With these bounds added, we can apply the basic algorithm described at the end of the previous section. For the dual variables, we get

$$w(r) = (AD_{x_A}^2 A^{\mathrm{T}} + FD_{x_F}^2 F^{\mathrm{T}})^{-1}(AD_{x_A}^2 c_A + FD_{x_F}^2 c_F), \tag{2.1}$$

where $D_{x_A} = \operatorname{diag}(x_A \wedge y_A)$ and $D_{x_F} = \operatorname{diag}([r\mathbf{1} + x_F] \wedge [r\mathbf{1} - x_F])$. Using (I.2), we see that

$$(AD_{x_A}^2 A^{\mathrm{T}} + FD_{x_F}^2 F^{\mathrm{T}})^{-1} AD_{x_A}^2 c_A = \{B - BF(D_{x_F}^{-2} + F^{\mathrm{T}} BF)^{-1} F^{\mathrm{T}} B\} AD_{x_A}^2 c_A, \tag{2.2}$$

where

$$B = (AD_{x_A}^2 A^{\mathrm{T}})^{-1}.$$

The only dependence on r in the right-hand side of (2.2) is through the diagonal matrix $D_{x_F}^{-2}$, which clearly goes to zero as r goes to infinity. Now using (I.1), we see that

$$(AD_{x_A}^2 A^{\mathrm{T}} + FD_{x_F}^2 F^{\mathrm{T}})^{-1} FD_{x_F}^2 c_F = BF(D_{x_F}^{-2} + F^{\mathrm{T}} BF)^{-1} c_F. \tag{2.3}$$

Substituting (2.2) and (2.3) into (2.1), we get

$$w(r) = \{B - BF(D_{x_F}^{-2} + F^{\mathrm{T}} BF)^{-1} F^{\mathrm{T}} B\} AD_{x_A}^2 c_A + BF(D_{x_F}^{-2} + F^{\mathrm{T}} BF)^{-1} c_F. \tag{2.4}$$

Now let r tend to infinity to get

$$w = \lim_{r \to \infty} w(r)$$

$$= BAD_{x_A}^2 c_A + BF(F^{\mathrm{T}} BF)^{-1}(c_F - F^{\mathrm{T}} BAD_{x_A}^2 c_A).$$

Note that the first term in this formula is exactly the formula we would use if there were no free variables.

To derive formulas for the step direction, we first partition it into two pieces

$$z = \begin{bmatrix} z_A \\ z_F \end{bmatrix}$$

corresponding to the constrained and free variables. The formula for z_A is simply

$$z_A = D_{x_A}^2(c_A - A^T w).$$

To get a formula for z_F, we must return to the prelimiting formulas, do some algebra, and then take limits again. So, let

$$z_F(r) = D_{x_F}^2(c_F - F^T w(r)).$$

From (2.4), we see that

$$z_F(r) = D_{x_F}^2[I - F^T BF(D_{x_F}^{-2} + F^T BF)^{-1}]\{c_F - F^T BAD_{x_A}^2 c_A\}. \tag{2.5}$$

Writing the identity matrix I as $(D_{x_F}^{-2} + F^T BF)(D_{x_F}^{-2} + F^T BF)^{-1}$, we see that the bracketed expression above reduces to $D_{x_F}^{-2}(D_{x_F}^{-2} + F^T BF)^{-1}$. Substituting this into (2.5) and letting r tend to infinity, we see that

$$\begin{aligned} z_F &= \lim_{r\to\infty} z_F(r) \\ &= (F^T BF)^{-1}(c_F - F^T BAD_{x_A}^2 c_A). \end{aligned}$$

After letting r go to infinity, the free-variable portions of the formulas for γ and δ vanish. The stopping rule and updating part of the algorithm change in the obvious way.

To summarize, given an x_A and x_F satisfying

$$0 < x_A < u,$$

$$[A \;\; F]\begin{bmatrix} x_A \\ x_F \end{bmatrix} = b,$$

the algorithm proceeds as follows:

(F.1) Calculate slacks for the upper bounds

$$y_A = u_A - x_A.$$

(F.2) Calculate the free-variables step direction

$$z_F = (F^T BF)^{-1}(c_F - F^T BAD_{x_A}^2 c_A)$$

where $B = (AD_{x_A}^2 A^T)^{-1}$.

(F.3) Calculate the dual variables

$$w = BAD_{x_A}^2 c_A + BFz_F.$$

(F.4) Calculate the constrained-variables step direction

$$z_A = D_{x_A}^2(c_A - A^T w).$$

(F.5) Calculate the complementary slackness

$$\gamma = \max\left(\frac{z_A}{x_A} \vee -\frac{z_A}{y_A}\right).$$

(F.6) Calculate the dual feasibility

$$\delta = -\min\left(\frac{z_A}{x_A^2} \wedge -\frac{z_A}{y_A^2}\right).$$

(F.7) Check the stopping rule: if

$$\gamma + M\delta < \frac{\varepsilon}{n}(|c_A \cdot x_A + c_F \cdot x_F| + 1),$$

where $M = \max(x_A \wedge y_A)$, then stop. The current solution is ε-optimal.

(F.8) Update the current solution

$$x_A \leftarrow x_A - \frac{\alpha}{\gamma} z_A,$$

$$x_F \leftarrow x_F - \frac{\alpha}{\gamma} z_F,$$

and go back to (F.1).

3. Phase 1 algorithm

In this section, we establish the equivalence between the phase 1 algorithm obtained by appending a column to the constraint matrix and the one described by Chandru and Kochar in [6].

To find an initial interior point for problem (P), we consider the following problem

$$\begin{aligned} (\text{P1}) \qquad \min \quad & \zeta \\ \text{subject to} \quad & [A \ \ \rho]\begin{bmatrix} x \\ \zeta \end{bmatrix} = b, \\ & 0 \leqslant x \leqslant u, \end{aligned}$$

where

$$\rho = b - A\xi$$

and ξ is any vector satisfying $0 < \xi < u$. Note the following observations:

(a) The variable ζ is a free variable.

(b) The point $x = \xi$, $\zeta = 1$ is feasible for (P1).

(c) Any feasible point for (P1) that satisfies $\zeta = 0$ also satisfies $Ax = b$.

(d) If the minimum value of ζ is positive, then (P) is infeasible.

Observations (b) and (c) explain why we want to minimize ζ. If we can find a feasible pair (x, ζ) with ζ negative, then we can take a linear combination of that point with $(\xi, 1)$ to find a new point which has $\zeta = 0$ exactly. This then gives us a starting point for the basic algorithm described in Section 1.

Applying the results on free variables to (P1), the step direction becomes

$$z = \begin{bmatrix} -D_\xi^2 A^T B\rho \\ 1 \end{bmatrix},$$

where $B = (AD_\xi^2 A^T)^{-1}$. (We have omitted a scale factor $(\rho \cdot B\rho)^{-1}$ which does not play a role here since z is merely a direction vector.) The stopping parameters γ and δ are calculated just as in the previous section and the stopping rule is changed to account for the omitted scale factor. If $\gamma < \alpha$, then the step $x \leftarrow x - z$ takes us exactly to $\zeta = 0$. Otherwise, we take the usual step: $x \leftarrow x - (\alpha/\gamma)z$.

Once we have taken a step to a new infeasible point, we might as well recalculate the residue, $\rho = b - Ax$, and begin the next iteration using a different LP which again starts with $\zeta = 1$. This change results in an algorithm that has a simple geometric interpretation, which we will discuss shortly.

To summarize, given an x which satisfies $0 < x < u$, the algorithm is:

loop

$y = u - x$

$\rho = b - Ax$

$z = -D_x^2 A^T B\rho$ where $B = (AD_x^2 A^T)^{-1}$

$$\gamma = \max\left(\frac{z}{x} \vee -\frac{z}{y}\right).$$

$$\delta = -\min\left(\frac{z}{x^2} \wedge -\frac{z}{y^2}\right).$$

$M = \max(x \wedge y)$

if $\gamma + M\delta < \beta\varepsilon/n$ where $\beta = \rho \cdot B\rho$

stop—problem is infeasible

else if $\gamma < \alpha$

$x \leftarrow x - z$

go to Phase 2

else

$$x \leftarrow x - \frac{\alpha}{\gamma} z$$

end if

end loop

If we compare the step direction z with that given by Chandru and Kochar in [6], we see that they agree. They arrived at this algorithm from simple geometric

arguments. Indeed, given an infeasible point x that lies between its upper and lower bounds, find the closest point on the hyperplane defined by the equality constraints. The notion of "closest" corresponds to the inner product $\langle a, b\rangle = a \cdot D_x^{-2} b$ as opposed to the usual Euclidean inner product. If the point on the hyperplane is also between the bounds, then it is feasible for (P). If, on the other hand, this point violates some of the bounds, then the current point is replaced by a point on the line segment connecting the current point with the point on the hyperplane. Even though this new point is not feasible, it is closer to feasibility than the previous point.

4. Dual algorithm

In this section, we apply the free variable algorithm derived in Section 2 to the dual of (P) and arrive at a *dual algorithm.* In the case where all the upper bounds are infinite, it will turn out that our dual algorithm coincides with the algorithm described by Adler et al. in [1]. After explicitly accounting for the constraints $x \leqslant u$ and adding slacks where appropriate, we see that the dual of (P) is

$$\text{(D)} \qquad \max \quad \begin{bmatrix} b \\ -u \\ 0 \end{bmatrix} \cdot \begin{bmatrix} w \\ v \\ s \end{bmatrix},$$

$$\text{subject to} \quad [A^{\mathrm{T}} \; -I \; I] \begin{bmatrix} w \\ v \\ s \end{bmatrix} = c,$$

$$v, s \geqslant 0.$$

Note that w is a free variable and so we can apply the algorithm derived in Section 2. Hence, given w, v, and s satisfying

$$v, s > 0,$$

$$A^{\mathrm{T}} w - v + s = c,$$

the algorithm proceeds as follows:

(D.1) Calculate the free-variables step direction[2]

$$z_w = (AD^* A^{\mathrm{T}})^{-1}(AD^* \Delta_{v^2} u - b)$$

where

$$D^* = \Delta_{(v^2+s^2)^{-1}}.$$

Note that, since the problem at hand is a maximization, the formulas for the various components of the step direction are all negated.

[2] If x and y are two vectors and f is a function from $\mathbb{R}^2$ into $\mathbb{R}$, we denote by $\Delta_{f(x,y)}$ the diagonal matrix whose ith diagonal element is $f(x_i, y_i)$.

(D.2) Calculate the primal variables

$$x = D^*(\Delta_v^2 u - A^T z_w).$$

(D.3) Calculate the constrained-variables step direction

$$z_v = \Delta_v^2(u - x),$$

$$z_s = \Delta_s^2 x.$$

(D.4) Calculate the complimentary slackness

$$\gamma = \max(v(u - x) \vee sx).$$

(D.5) Calculate the primal feasibility

$$\delta = -\min((u - x) \wedge x).$$

Note that this measure of primal feasibility detects the largest violation of the inequality constraints in the primal ($0 \leq x$ and $x \leq u$) but it does not measure any violation of the equality constraints $Ax = b$. The reason is that the equality constraints are automatically satisfied. To see this, take the formula for the primal variables given in (D.2), multiply by A and simplify using the formula for z_w given in (D.1).

(D.6) Check the stopping rule: if

$$\gamma + M\delta < \frac{\varepsilon}{n}(|b \cdot w - u \cdot v| + 1),$$

where $M = \max(v \wedge s)$, then stop. The current solution is ε-optimal.

(D.7) Update the current solution

$$\begin{aligned} w &\leftarrow w - \frac{\alpha}{\gamma} z_w, \\ v &\leftarrow v - \frac{\alpha}{\gamma} z_v, \\ s &\leftarrow s - \frac{\alpha}{\gamma} z_s. \end{aligned} \tag{4.1}$$

Since the vector s is simply the slacks for the inequality constraints, it is possible to replace the update formula (4.1) with

$$s = c + v - A^T w.$$

This last formula is superior from an implementation point of view since it guarantees that each iteration preserves feasibility.

If some of the upper bounds in (P) are actually infinite, then the algorithm just presented must be changed by setting the corresponding surplus variables equal to zero and ignoring them in the calculation of complimentary slackness and primal feasibility.

We end this section with a brief discussion of the relative merits of the primal and the dual algorithms. Both algorithms involve inverting in each iteration an $m \times m$ matrix having the same symbolic structure as AA^T. Hence, the time for each iteration should be almost the same. Since both these algorithms tend to require about the same number of iterations, the total time to solve a problem appears to be about equal. There are some technical differences, however.

The primal algorithm first finds feasible primal variables and generates feasible duals only as the optimal vertex is approached. The dual algorithm, on the other hand, first finds feasible dual variables and generates feasible primals only as the optimum is reached. Hence, if primal feasibility is important, but optimality is less of a concern, then the primal algorithm is the one to use. However, if the problem is primal degenerate and optimality is important, then the primal algorithm might have trouble achieving dual feasibility and so a provably optimal solution won't be found. If this happens, the dual algorithm might be the answer. Another argument for the dual algorithm is that since every constraint has slack variables, it is possible to use fast inaccurate inversion techniques and still maintain dual feasibility. However, this only helps in early iterations, since as optimality is approached it is important to get accurate primal estimates and these are quickly lost with inaccurate inversions.

5. Free variables—Phase 1

This section is included merely to lay the groundwork for the derivation of the phase 1 algorithm for the dual which is derived in the next section.

To find an initial interior point for problem (F), we consider the following problem

$$\begin{aligned} (\text{F1}) \qquad & \min \quad \zeta \\ & \text{subject to} \quad [A \;\; F \;\; \rho] \begin{bmatrix} x_A \\ x_F \\ \zeta \end{bmatrix} = b, \\ & \qquad\qquad 0 \leq x_A \leq u, \end{aligned}$$

where

$$\rho = b - A\xi_A - F\xi_F,$$

ξ_A is any vector satisfying $0 < \xi_A < u$ and ξ_F is arbitrary.

We now apply the free variables algorithm derived in Section 2 to (F1) making the same simplifications which we made when deriving the primal phase 1 algorithm in Section 3. Hence, given x_A satisfying $0 < x_A < u$ and an arbitrary x_F, the algorithm is:

loop

$y_A = u - x_A$

$\rho = b - Ax_A - Fx_F$

$z_F = -(F^T B F)^{-1} F^T B \rho$ where $B = (A D_{x_A}^2 A^T)^{-1}$

$z_A = -D_{x_A}^2 A^T B(\rho + F z_F)$

$\gamma = \max\left(\frac{z_A}{x_A} \vee -\frac{z_A}{y_A}\right)$

$\delta = -\min\left(\frac{z_A}{x_A^2} \wedge -\frac{z_A}{y_A^2}\right)$

$M = \max(x_A \wedge y_A)$

if $\gamma + M\delta < \beta\varepsilon/n$ where $\beta = \rho \cdot B\rho + \rho \cdot BFz_F$

stop—problem is infeasible

else if $\gamma < \alpha$

$x_A \leftarrow x_A - z_A$

$x_F \leftarrow x_F - z_F$

go to Phase 2

else

$x_A \leftarrow x_A - \frac{\alpha}{\gamma} z_A$

$x_F \leftarrow x_F - \frac{\alpha}{\gamma} z_F$

end if

end loop

6. Dual algorithm—Phase 1

It is easy to find an interior starting point for the dual algorithm derived in Section 4 since every constraint has a slack and a surplus variable. For example, it is possible to simply put $w = 0$ and then adjust the slacks and surpluses as needed. However, if some or all of the upper bounds are infinite, then the corresponding surplus variable must be set to zero from the very start. In this case (which is almost always), finding a starting point is non-trivial. In this section we derive an algorithm for finding such a point by applying the phase 1 algorithm developed in Section 5 for problems with free variables.

We start by first assuming that all variables have upper bounds. Hence, given an arbitrary initial w (e.g., $w=0$) and positive initial values for v and s the algorithm proceeds as follows:

(D1.1) Calculate the residuals

$$\rho = c - A^{\mathrm{T}}w + v - s.$$

(D1.2) Calculate the step direction for w

$$z_w = -(AD^*A^{\mathrm{T}})^{-1}AD^*\rho$$

where

$$D^* = \Delta_{(v^2+s^2)^{-1}}.$$

(D1.3) Calculate the step direction for v and s

$$z_v = \Delta_{v^2}D^*(\rho + A^{\mathrm{T}}z_w),$$

$$z_s = -\Delta_{s^2}D^*(\rho + A^{\mathrm{T}}z_w)$$

(D1.4) Calculate the complimentary slackness

$$\gamma = \max\left(\frac{z_v}{v} \vee \frac{z_s}{s}\right).$$

(D1.5) Calculate the primal feasibility

$$\delta = -\min\left(\frac{z_v}{v^2} \wedge \frac{z_s}{s^2}\right).$$

(D1.6) Check the stopping rule: if

$$\gamma + M\delta < \beta\varepsilon/n,$$

where $M = \max(v \wedge s)$ and $\beta = \rho \cdot D^*\rho + \rho \cdot D^*A^{\mathrm{T}}z_w$, then stop. The problem (D) is infeasible.

(D1.7) Update the current solution. If $\gamma < \alpha$, then

$$w \leftarrow w - z_w,$$

$$v \leftarrow v - z_v,$$

$$s \leftarrow s - z_s.$$

We now have a starting point for phase 2. If, however, $\gamma \geqslant \alpha$, then the update is as above except that the step directions are all scaled down by a factor of α/γ. In this case, we are still infeasible and so we return to (D1.1).

It is interesting to note the similarity between the phase 1 dual algorithm and the phase 2 primal algorithm with which we started. There are also striking similarities between the phase 2 dual algorithm and the phase 1 primal algorithm (especially if upper bounds are set to infinity).

Finally, we again remark that if some of the upper bounds in (P) are infinite, then the corresponding surplus variable must be set to zero from the start. It will then automatically remain zero throughout this phase 1.

References

[1] I. Adler, A. Karmarkar, M.G.C. Resende and G. Veiga, "An implementation of Karmarkar's algorithm for linear programming," to appear in *Mathematical Programming.*

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

[3] D.A. Bayer and J. Lagarias, "Nonlinear geometry of linear programming: Parts I and II," to appear in *Transactions of the AMS.*

[4] T.M. Cavalier and K.C. Schall, "Implementing a projective algorithm for solving inequality-constrained linear programs," IMSE Working Paper 86-128, The Pennsylvania State University (1986).

[5] V. Chandru and B.S. Kochar, "A class of algorithms for linear programming," Research Memorandum No. 85-14, Purdue University (1985).

[6] V. Chandru and B.S. Kochar, "Exploring special structures using a variant of Karmarkar's algorithm," Research Memorandum No. 86-10, Purdue University (1986).

[7] R.J. Vanderbei, M.S. Meketon and B.A. Freedman, "A modification of Karmarkar's linear programming algorithm," *Algorithmica* 1 (1986) 395-407.

[8] R.J. Vanderbei, "The affine-scaling algorithm and primal degeneracy," talk presented at the Conference on Advances in Mathematical Programming held in Monterey, CA, March 2-5, 1987.