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

TECHNICAL REPORT NO. 720

October 1986

(Revised October 1987)

EXPLOITING SPECIAL STRUCTURE IN KARMARKAR'S LINEAR PROGRAMMING ALGORITHM

By

Michael J. Todd*

*Research supported in part by NSF Grant ECS-8602534 and the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University.

Exploiting Special Structure in Karmarkar's Linear Programming Algorithm

Michael J. Todd*

Abstract. We propose methods to take advantage of specially-structured constraints in a variant of Karmarkar's projective algorithm for standard form linear programming problems. We can use these constraints to generate improved bounds on the optimal value of the problem and also to compute the necessary projections more efficiently, while maintaining the theoretical bound on the algorithm's performance. It is shown how various upper-bounding constraints can be handled implicitly in this way. Unfortunately, the situation for network constraints appears less favorable.

Key words: Linear programming, Karmarkar's algorithm, special structure.

Abbreviated title: Special structure in Karmarkar's algorithm.

*Research supported in part by National Science Foundation Grant ECS-8602534, ONR Contract N00014-87-K-0212 and the U.S. Army Research Office through the Mathematical Sciences Institute of Cornell University.

1. Introduction.

In [15], Karmarkar introduced a new polynomial projective algorithm for linear programming, which has considerable promise for solving large sparse problems. In particular, very impressive results have been reported for its application to extremely large network planning problems [16], which have a time-staged formulation that is known to lead to poor performance for Dantzig's simplex method [4]. On the other hand, other special structures arising frequently in very large problems have been exploited very successfully in various implementations of the simplex method. In particular, we mention that simple [4], generalized [6], and variable [20,21,23] upper bounds can be handled implicitly using compact inverse techniques, and more generally structured problems can also be dealt with using (representations of) the inverse of matrices far smaller than the basis matrix (see [5,12,17,18]. This paper is concerned with investigating the extent to which specially-structured constraints can also be exploited in Karmarkar's algorithm.

We confine ourselves to variants of Karmarkar's original projective algorithm for which polynomial time bounds can be established. We do not consider the affine variant, described by several authors including Barnes [2], Vanderbei, Meketon and Freedman [26], and Chandru and Kochar [3]. The last two papers give a modification of the affine algorithm for problems with simple upper bounds, which are also considered by Gill, Murray, Saunders, Tomlin and Wright [10]. We address only polynomial versions of the algorithm, and our interest is in implementations of these variants which will preserve the polynomial bound while handling special constraints efficiently. We discuss two general techniques for exploiting such constraints. The first, by using more constrained subproblems to generate improved bounds and hence better directions, is likely to result in fewer iterations of the overall algorithm; the subproblems resemble those that arise in Dantzig-Wolfe decomposition [4,7]. The second shows that the projection step, which is the major computational requirement at each iteration, can be performed as the composition of two simpler projections. This technique hence makes each iteration cheaper; it is reminiscent of compact-inverse methods for the simplex method.

Other related papers include Kapoor and Vaidya [14], which discusses a theoetical improvement of Karmarkar's algorithm for multi-commodity flow problems, and Rinaldi [19], which considers problems with simple upper bounds.

In Section 2 we describe a variant of Karmarkar's projective algorithm which can be applied more directly to standard form problems. Section 3 discusses techniques to exploit special constraints in this algorithm in a general framework, and section 4 provides further details on some of the necessary steps. Finally, in section 5 we consider specific examples, showing that upper bounds (simple, generalized or variable) can be handled efficiently, while more general situations, for example, involving flow conservation constraints, are less favorable.

2. A variant of Karmarkar's algorithm.

Here we describe a variant of the projective algorithm due essentially to Anstreicher [1], Gay [9], Gonzaga [11], Jensen and Steger [13,22] and Ye and Kojima [27]; this variant easily handles problems in standard form. (See also de Ghellinck and Vial [8] for a related method.) Our development is slightly different from those in the references above, but is chosen to facilitate the treatment of special constraints. In particular, the notation is selected to avoid as far as possible special symbols for the transformed problem.

The original problem of interest is

$$
(P_0) \qquad \min \quad c_0^T x \qquad
\begin{aligned}
A_0 x &= 0 \\
g_0^T x &= 1 \\
x &\geq 0,
\end{aligned}
$$

where A_0 is $m \times n$ of rank m, and c_0 and g_0 are n-vectors, with g_0 nonnegative and nonzero. We assume that the feasible region of (P_0) is bounded and nonempty, and that we know a strictly positive vector x_0 contained in it. This implies that (P_0) has an optimal solution; we assume further that we know a lower bound z_0 on its optimal value z_*. Without loss of generality, we may assume that $x_0 = e$, the vector of ones; this makes our notation below consistent. In later sections we are interested in exploiting a subdivision of the rows of A_0 into "general" constraint rows H_0 and "special" constraint rows M_0.

Given a standard form linear programming problem

$$(\tilde{P}) \quad \min \tilde{c}^T\tilde{x}, \quad \tilde{A}\tilde{x} = \tilde{b}, \quad \tilde{x} \geq 0,$$

we can derive an "equivalent" problem of form (P_0) as follows. We augment the variables $\tilde{x}$ with a new variable ξ ($\equiv 1$, to handle the right-hand side) and, if necessary, an artificial variable λ (to make e feasible in the resulting problem). If we know a strictly positive feasible solution $\tilde{x}$ to $(\tilde{P})$, we may scale so that $\tilde{x} = e$ and then λ is unnecessary; in this case the last column of the following matrices can be deleted.

Let

$$x^T = (\tilde{x}^T, \ \xi, \ \lambda) \tag{2.1a}$$

$$c_0^T = (\tilde{c}^T, \ 0, \ \nu) \tag{2.1b}$$

$$A_0 = (\tilde{A}, \ -\tilde{b}, \ \tilde{b}-\tilde{A}e) \tag{2.1c}$$

$$g_0^T = (0, \ 1, \ 0). \tag{2.1d}$$

where υ is a large constant. Note that, except for its last one or two possibly dense columns, A_0 inherits the structure of $\tilde{A}$. Special techniques are necessary to handle the artificial variable λ, e.g. dropping it if necessary. For simplicity, we usually assume below that an artificial variable is not needed.

The algorithm for (P_0) proceeds as follows. At the kth iteration we will have a strictly positive feasible solution x_k and we wish to generate its successor. We also have a lower bound z_k on z_*. Let $X_k = \mathrm{diag}(x_k)$ and consider the affine transformation

$$T_k(x) \rightarrow X_k^{-1}x \tag{2.2}$$

so that $T_k(x_k) = e$. Let

$$A = A_k \equiv A_0X_k \tag{2.3a}$$

$$c = c_k \equiv X_kc_0, \text{ and} \tag{2.3b}$$

$$g = g_k \equiv X_kg_0; \tag{2.3c}$$

then (P_0) is equivalent to the rescaled problem

$$
(P) \qquad \begin{aligned} \min \quad & c^T x \\ & Ax = 0 \\ & g^T x = 1 \\ & x \geq 0 \end{aligned}
$$

for which $x = e$ is feasible. We stress that A, c and g refer to <u>scaled</u> matrices and thus vary from iteration to iteration. Other papers use overbars (e.g. [1]) or hats [24,25] to indicate scaled quantities, but this notation would become cumbersome with the operations used below.

We denote by $v(\cdot)$ the optimal value of problem $(\cdot)$, with the convention that the value is $+\infty$ for an infeasible (minimization) problem and $-\infty$ for a feasible problem with objective function unbounded below. Thus $z_* = v(P_0) = v(P)$. Now replacing c by $c - z_* g$ in (P) (or c_0 by $c_0 - z_* g_0$ in (P_0)) gives an equivalent problem with optimal value zero. In this problem, any normalization instead of $g^T x = 1$ that yields a bounded feasible region will also result in an optimal value of zero, since every feasible solution of one problem is a positive scalar multiple of a feasible solution of the other, and hence has an objective function value of the same sign. (Choosing a different normalizing constraint amounts to performing a projective transformation; the cone $\{x \geq 0: Ax = 0\}$ is cut by a different hyperplane than $\{x: g^T x = 1\}$ and points are radially projected from one to the other.)

In particular, we consider the normalization $e^T x = n$ and the parametric problem (where z is not necessarily equal to z_*)

$$(P(z)) \qquad \begin{aligned} \min \quad & (c-zg)^T x \\ & Ax = 0 \\ & e^T x = n \\ & x \geq 0 \end{aligned}$$

for which $x = e$ is still feasible. Our next iterate corresponds to taking a step of the standard Karmarkar algorithm applied to $(P(z))$ for some $z \leq z_*$.

We use the following notation: the constraint matrix in $(P(z))$ is

$$B = \begin{bmatrix} A \\ e^T \end{bmatrix}; \tag{2.4}$$

for any matrix F, P_F denotes the orthogonal projection onto the null space of F; and for any $u \in R^n$,

$$u_p = P_B u \quad \text{and} \quad u_q = P_A u. \tag{2.5}$$

Note that, since $Ae = A_0 x_k = 0$,

$$P_B = P_A P_{e^T} = P_{e^T} P_A, \tag{2.6}$$

so that

$$u_p = u_q - (e^T u_q/n)e. \tag{2.7}$$

The iteration then involves the following three steps:

1) Obtaining a lower bound $\underline{z} = z_{k+1} \geq z_k$ on z_*;

2) Setting

$$d = -(c_p - \underline{z}g_p) \quad \text{and} \tag{2.8}$$

$$x = e + \alpha d/\|d\| \tag{2.9}$$

for suitable $\alpha > 0$ so that $x > 0$; and

3) Letting

$$x_{k+1} = \frac{X_k x}{g^T x}. \tag{2.10}$$

Thus d is a search direction in the null space of B; x is a feasible solution to $P(z)$ obtained by moving in this direction from the "current solution" e; and x_{k+1} is the result of scaling by $g^T x$ to get a feasible solution to (P) and then transforming back to get a feasible solution to (P_0).

We remark in passing that setting $d' = -(c_q - \underline{z}g_q)$ and $x' = e + \alpha' d'/\|d'\|$ gives points that are positive scalar multiples of the points x given by (2.8)-(2.9), at least if α is small or $d' \geq 0$. In a sense, then, it is unnecessary to add the normalization $e^T x = 1$ to $(P(z))$ at all, except that it simplifies the analysis.

To make the iteration precise, we need to specify how $\underline{z}$ is obtained and how α is selected. The lower bound is based on the simple problem

$$(\mathrm{SP}(z)) \qquad \begin{aligned} \min\ & (c_q - zg_q)^T x \\ & e^T x = n \\ & x \geq 0. \end{aligned}$$

Note that $c_q = c - A^T y_c$ for some y_c (in fact, $y_c = (AA^T)^{-1}Ac$) and similarly $g_q = g - A^T y_g$ for some y_g; thus

$$(c_q - zg_q) = (c - zg) - A^T(y_c - zy_g).$$

It follows that $(\mathrm{SP}(z))$ is a partial Lagrangean relaxation or Dantzig-Wolfe subproblem of $(\mathrm{P}(z))$; that is, some of the constraints of $(\mathrm{P}(z))$ are removed and placed with multipliers in the objective function.

We can now describe how $\underline{z}$ is obtained. If $v(\mathrm{SP}(z_k)) \leq 0$, we merely set $\underline{z} = z_k$. Suppose now that $v(\mathrm{SP}(z_k)) > 0$. Now since $(\mathrm{SP}(z))$ is a relaxation of $(\mathrm{P}(z))$, $v(\mathrm{SP}(z_*)) \leq v(\mathrm{P}(z_*)) = 0$; more directly, a positive scalar multiple of the optimal solution of (P) is feasible in $(\mathrm{SP}(z_*))$ with objective value zero. Hence we can find $\underline{z}$ with

$$z_k < \underline{z} \leq z_* \quad \text{and} \quad v(\mathrm{SP}(\underline{z})) = 0. \tag{2.11}$$

Indeed, $v(\mathrm{SP}(z))$ is trivial to compute; it is the concave piecewise-linear function

$$v(SP(z)) = n \min_j \; e_j^T(c_q - zg_q) \qquad (2.12)$$

where e_j is the jth unit vector. Hence $\underline{z}$ is unique and can be obtained by a simple minimum-ratio test.

If $\underline{z}$ is updated in this way, let

$$\underline{y} = y_c - \underline{z}y_g; \qquad (2.13)$$

then from (2.11) and (2.12) we have

$$c - \underline{z}g - A^T\underline{y} = c_q - zg_q \geq 0.$$

Hence we have proved

Lemma 2.1. $(\underline{y},\underline{z})$ is feasible in the dual to (P):

(D) $\max z,\ A^Ty + gz \leq c,$

and in the dual to (P_0):

(D_0) $\max z,\ A_0^Ty + g_0z \leq c_0.$

Thus the lower bounds obtained correspond to dual feasible solutions.

We now use $\underline{z}$ to compute our next iterate x_{k+1}. Progress towards the solution is measured by Karmarkar's potential function

$$f(x;\ h) = \sum_j \ell n(h^T x / e_j^T x)$$

for a suitable objective function $h^T x$. Any α in $(0,1)$ suffices to guarantee $x_{k+1} > 0$, but a suitable choice ensures a reasonable decrease in f. Indeed, the arguments of Todd and Burrell [25] (using those of Karmarkar [15] as well as linear programming duality) yield

<u>Lemma 2.2</u>. Suppose $\underline{z} \leq z_*$ with $v(SP(\underline{z})) \leq 0$. Then if x_{k+1} is obtained from (2.8)-(2.10) with $\alpha = 1/3$,

$$f(x_{k+1};\ c_0 - \underline{z}g_0) \leq f(x_k;\ c_0 - \underline{z}g_0) - 1/5. \quad (2.14)$$

Using (2.14) repeatedly, and the inequality

$$f(x_k;\ c_0 - z_{k+1}g_0) \leq f(x_k;\ c_0 - z_k g_0)$$

since $z_{k+1} \geq z_k$, we deduce that

$$f(x_\ell;\ c_0 - z_\ell g_0) \leq f(x_0;\ c_0 - z_0 g_0) - \ell/5. \quad (2.15)$$

Now $-\Sigma\ \ell n\ (e_j^T x) \geq -n\ \ell n(e^T x/n)$ by the arithmetic-geometric mean inequality, and so from (2.15) we can deduce

$$c_o^T x_\ell - z_\ell \leq (e^T x_\ell / n)\exp(-\ell/5n)(c_0^T x_0 - z_0). \quad (2.16)$$

The first factor on the right-hand side is bounded by assumption, while the second goes to zero like a geometric progression. Thus the duality gap converges to zero linearly.

In practice, it is preferable to choose α by some line search technique to encourage larger decreases in the potential function than that in (2.14).

Given that $v(SP(z))$ is easy to compute by (2.12), and that c_p and g_p are easily obtained from c_q and g_q respectively using (2.7), the major computational burden in each iteration lies in calculating c_q and g_q. Gay [9] has pointed out that it may be cheaper to compute first $(c-z_k g)_q$; then if $\underline{z}$ is not updated, a second projection need not be computed.

In the next sections we discuss how to exploit special structure in the constraints $A_0 x = 0$ within this framework.

3. Exploiting special structure in A_0.

Suppose that the constraints of (P_0) are divided into "general" and "special" constraints, corresponding to a partition

$$A_0 = \begin{bmatrix} H_0 \\ M_0 \end{bmatrix} \quad (3.1)$$

with $H_0 x = 0$ the "general" (or hard) and $M_0 x = 0$ the "special" (or easy) constraints. At iteration k, setting

$$H = H_k \equiv H_0 X_k, \quad M = M_k \equiv M_o X_k, \tag{3.2}$$

we have

$$A = \begin{bmatrix} H \\ M \end{bmatrix}. \tag{3.3}$$

Our (transformed) problem can be written as

$$\text{(P)} \qquad \begin{aligned} \min \quad & c^T x \\ & Hx = 0 \\ & Mx = 0 \\ & g^T x = 1 \\ & x \geq 0, \end{aligned}$$

and, by partitioning the dual variables as

$$y^T = (s^T, t^T),$$

the dual problem is

$$\text{(D)} \qquad \begin{aligned} \max \quad & z \\ & H^T s + M^T t + gz \leq c. \end{aligned}$$

We wish to exploit the structured matrix M in two ways:

(i) in improving the lower bounds $\underline{z}$; and

(ii) in computing projections u_q efficiently.

The methods used are reminiscent of Dantzig-Wolfe decomposition [4,7] (i) and compact-basis techniques (e.g., [6,12,17,18,20,21,23]) (ii).

An ideal solution to (ii) would be found if

$$P_A = P_H P_M = P_M P_H \quad \text{(compare (2.6));}$$

but this fails in general since $HM^T \neq 0$. Let us therefore define

$$\hat{H} = H\, P_M \tag{3.4}$$

so that the rows of $\hat{H}$ are the projections of the rows of H orthogonal to the rows of M. Then the nullspace of A coincides with that of

$$\hat{A} = \begin{bmatrix} \hat{H} \\ M \end{bmatrix} \tag{3.5}$$

moreover, since we have $\hat{H}M^T = 0$

$$P_A = P_{\hat{A}} = P_{\hat{H}} P_M = P_M P_{\hat{H}}. \tag{3.6}$$

For any $u \in R^n$, we write

$$u_r = P_{\hat{H}}\, u; \tag{3.7}$$

Thus u_r, u_q and u_p are projections of u onto progressively smaller (and nested) subspaces. We will use (3.6) later in computing $c_q - \underline{z}'g_q$ for some $\underline{z}'$ with $z_k \leq \underline{z}' = z_{k+1} \leq z_*$.

We obtain our improved lower bound $\underline{z}'$ by considering

$$
(\mathrm{SP}'(z)) \qquad
\begin{aligned}
\min \quad & (c_r - zg_r)^T x \\
& Mx = 0 \\
& e^T x = n \\
& x \geq 0.
\end{aligned}
$$

From (3.4), $\hat{H} = H - KM$ for some K, and therefore c_r can be written as

$$c_r = c - \hat{H}^T s_c \tag{3.8a}$$

$$= c - H^T s_c - M^T \tilde{t}_c \tag{3.8b}$$

for some s_c, $\tilde{t}_c$, and similarly

$$g_r = g - H^T s_g - M^T \tilde{t}_g \tag{3.9}$$

for some s_g, $\tilde{t}_g$. Hence $(\mathrm{SP}'(z))$ can be viewed as a partial Lagrangian relaxation or Dantzig-Wolfe subproblem for $(P(z))$, in which only the hard constraints $Hx = 0$ are eliminated, but all the constraints $Ax = 0$ are incorporated into the objective function with multipliers.

We now show that $(SP'(z))$ is a restriction of $(SP(z))$. Indeed, by (3.6) we have for any $u \in R^n$

$$u_q = u_r - M^T t_u$$

for some t_u, and thus the objective functions of $(SP'(z))$ and $(SP(z))$ are equal for any x in the null space of M. Hence $(SP'(z))$ would be unchanged if its objective function were replaced by $(c_q - zg_q)^T x$, and because of the additional constraints $Mx = 0$ we deduce that

$$v(SP'(z)) \geq v(SP(z)). \qquad (3.10)$$

Now $\underline{z}'$ is defined as follows. If $v(SP'(z_k)) \leq 0$, set $\underline{z}' = z_k$. Otherwise, by parametrically solving $SP'(z)$, find $\underline{z}'$ with

$$z_k < \underline{z}' \text{ and } v(SP'(\underline{z}')) = 0. \qquad (3.11)$$

Suppose $\underline{z}$ is calculated using $(SP(z))$ as in section 2.

<u>Lemma 3.1</u>. $z_k \leq \underline{z} \leq \underline{z}' \leq z_*$, and $v(SP(\underline{z}')) \leq 0$.

<u>Proof</u>. If $v(SP(z_k)) \leq 0$, then $\underline{z} = z_k$ and hence $\underline{z}' \geq \underline{z}$. Otherwise, $v(SP'(z_k)) \geq v(SP(z_k)) > 0$ by (3.10) so that both $\underline{z}$

and $\underline{z}'$ are updated. Moreover, $v(SP'(\underline{z})) \geq v(SP(\underline{z})) = 0$, so that $\underline{z}' \geq \underline{z}$ (as the optimal value of a parametric objective function linear programming problem, $v(SP'(z))$ is piecewise linear and concave). It remains to show that $\underline{z}'$ exists and satisfies $\underline{z}' \leq z_*$ when it is updated. Now the optimal solution of (P), after multiplying by a positive scalar, yields an optimal solution to $(P(z_*))$ with objective value zero. Since $(SP'(z_*))$ is a Lagrangean relaxation of $(P(z_*))$, $v(SP'(z_*)) \leq 0$. With $v(SP'(z_k)) > 0$ and concavity, this establishes the existence of a unique $\underline{z}'$ satisfying (3.11), and also that $\underline{z}' \leq z_*$.

The lemma shows that $(SP'(z))$ provides a possibly improved lower bound for use in the algorithm. Lemma 2.2 applies with $\underline{z}$ replaced by $\underline{z}'$. We therefore replace d in (2.8) by

$$d = -(c_p - \underline{z}'g_p) \tag{3.12}$$

and obtain x_{k+1} from (2.9)-(2.10). Here d is calculated via

$$\begin{aligned} d &= -P_{e^T} P_M P_{\hat{H}} (c - \underline{z}'g) \\ &= -P_{e^T} P_M (c_r - \underline{z}'g_r). \end{aligned} \tag{3.13}$$

In order that this approach be reasonably efficient, it is assumed that the parametric problem $(SP'(z))$ can be easily solved, and that projection onto the null space of M (briefly, M-projection) be simple. In this case, it is straightforward to compute the improved lower bound $\underline{z}'$ (limited computational experience (see [24]) suggests that the improved directions resulting from better bounds can save from 1 to 3 iterations). Moreover, we have replaced two A-projections (to get c_q and g_q) by several M-projections (to get $\hat{H}$), two $\hat{H}$-projections (yielding c_r and g_r) and one final M-projection (giving $P_M(c_r-\underline{z}'g_r) = P_A(c-\underline{z}'g)$). This trade-off is similar to that arising in compact inverse methods in linear programming, in which a few operations involving (a representation of the inverse of) a large basis matrix (corresponding to A) are replaced by several operations involving a simple part of the basis (corresponding to M) and a few involving a small working basis (corresponding to $\hat{H}$).

Before we consider specific examples in which the assumptions of the previous paragraph are valid, we describe how the lower bounds $\underline{z}'$ can be validated by duality (compare Lemma 2.1), and then discuss two general techniques useful in computing $\underline{z}'$ and M-projections.

First, recall from (3.8)-(3.9) that

$$c_r = c - H^T s_c - M^T \tilde{t}_c$$

and

$$g_r = g - H^T s_g - M^T \tilde{t}_g;$$

moreover, s_c and s_g are usually obtained in the computation of c_r and g_r ($s_c = (\hat{H}\hat{H}^T)^{-1}\hat{H}c$ and similarly for s_g), and $\tilde{t}_c = -K^T s_c$ and $\tilde{t}_g = -K^T s_g$ are then available if K with $\hat{H} = H-KM$ is known. Now let $(\hat{t},0)$ be the optimal solution to the dual of $(SP'(\underline{z}'))$ - the last component is zero since $v(SP'(\underline{z}')) = 0$. It follows that

$$(c_r - \underline{z}'g_r) - M^T \hat{t} \geq 0,$$

so that, with

$$\underline{s} = s_c - \underline{z}'s_g$$

$$\underline{t} = (\tilde{t}_c - \underline{z}'\tilde{t}_g) + \hat{t},$$

$$c - H^T\underline{s} - M^T\underline{t} - g\underline{z}' \geq 0.$$

Thus $(\underline{s},\underline{t},\underline{z}')$ is feasible in (D) and (D_0) as in Lemma 2.1.

If K is not known, then we can replace c_r and g_r in $(SP'(z))$ by $\hat{c}_r = c - H^T s_c$, $\hat{g}_r = g - H^T s_g$; if $(\bar{t},0)$ is the corresponding optimal dual solution for $z = \underline{z}'$, then again $(\underline{s},\underline{t},\underline{z}')$ is feasible in (D) and (D_0).

4. Two general observations.

In this section, we discuss an easier way to find $\underline{z}'$ and a method to reduce M-projection to a usually simpler projection.

4.1. Computing $\underline{z}'$.

Recall that $\underline{z}' = z_k$ if $v(SP'(z_k)) \leq 0$, while

$$z_k < \underline{z}' \quad \text{and} \quad v(SP'(\underline{z}')) = 0$$

otherwise. Hence we are interested in parametrically calculating the sign of $v(SP'(z))$. Consider instead the problem

$$(SP''(z)) \qquad \begin{aligned} \min \quad & (c_r - zg_r)^T x \\ & Mx = 0 \\ & g^T x = 1 \\ & x \geq 0 \end{aligned}$$

where the normalization $g^T x = 1$ has replaced $e^T x = n$. By scaling the variables, we see that $(SP''(z))$ has the same optimal value as

$$(SP'''(z)) \qquad \begin{aligned} \min \quad & (\tilde{c}_r - z\tilde{g}_r)^T x \\ & M_0 x = 0 \\ & g_0^T x = 1 \\ & x \geq 0 \end{aligned}$$

involving the original data, where $\tilde{c}_r = X_k^{-1} c_r$, $\tilde{g}_r = X_k^{-1} g_r$.

Lemma 4.1. $v(SP''(z)) < 0$ iff $v(SP'(z)) < 0$.

Proof. First suppose $v(SP''(z)) < 0$. Then either $(SP''(z))$ has an optimal solution $x \neq 0$ or it is unbounded below. In the first case, nx/e^Tx is feasible in $(SP'(z))$ with a negative objective value, while in the second there is some ray $\bar{x} \neq 0$ with $(c_r - zg_r)^T\bar{x} < 0$, $M\bar{x} = 0$ and $g^T\bar{x} = 0$; then $n\bar{x}/e^T\bar{x}$ is feasible in $(SP'(z))$ with negative objective value. Hence $v(SP'(z)) < 0$. Conversely, assume $v(SP'(z)) < 0$, and let x be optimal in $(SP'(z))$. Note that both $(SP'(z))$ and $(SP''(z))$ have nonempty feasible regions, since they contain (scalar multiples of) feasible solutions of (P); moreover, the former is bounded. If $g^Tx > 0$, then x/g^Tx is feasible in $(SP''(z))$ with negative objective value. On the other hand, $g^Tx = 0$ implies that $(SP''(z))$ is unbounded below, since it is feasible and $(c_r - zg_r)^Tx < 0$, $Mx = 0$, $g^Tx = 0$, $x \geq 0$. Hence in either case $v(SP''(z)) < 0$.

The lemma shows that $\underline{z}'$ can be found by setting $\underline{z}' = z_k$ if $v(SP''(z_k)) \leq 0$ and otherwise satisfying

$$z_k \leq \underline{z}' \quad \text{and} \quad \underline{z}' = \sup\{z: v(SP''(z)) \geq 0\}. \quad (4.1)$$

Note that, while $v(SP'(z))$ is piecewise-linear, concave and <u>continuous</u>, $v(SP''(z))$ may not be continuous - in particular, there may be no $\underline{z}'$ with $v(SP''(\underline{z}')) = 0$. Nevertheless, if $(SP''(z))$ or $(SP'''(z))$ is easily solved parametrically in z, $\underline{z}'$ can be calculated using (4.1).

<u>4.2. M-projection</u>.

If (P_0) arises from a standard form problem $(\tilde{P})$ as in (2.1), then A_0 (and H_0, M_0, A, H and M) will have one or two probably dense columns, with the rest inheriting their structure from $\tilde{A}$. Thus we will partition M as

$$M = [S,T] = [S,SU] \qquad (4.2)$$

where $T = SU$ contains the t dense columns of M. Note that $Me = 0$, so that if $t = 1$ the representation of T as SU is trivial - U is a column vector of minus ones. If $t = 2$ then using perhaps the previous iterate we can find another vector w in the null space of M which uses the last two columns, and again U can be obtained with $T = SU$.

We now show how M-projections can be carried out using simpler S-projections together with inverses of small matrices. We concentrate on projecting the rows of H, partitioned as

$$H = [J,K], \tag{4.3}$$

but the procedure is identical when projecting c or g. Let

$$V = (I - P_S)U, \quad W = (I + V^T V)^{-1}; \tag{4.4}$$

then

$$P_M = \begin{bmatrix} P_S + VWV^T & -VW \\ -WV^T & W \end{bmatrix}. \tag{4.5}$$

Now, if

$$\hat{H} = [\hat{J},\hat{K}], \tag{4.6}$$

we find that

$$\hat{J} = JP_S + (JV - K)WV^T \tag{4.7}$$

differs from JP_S by a matrix of rank t, while

$$\hat{K} = -(JV-K)W \tag{4.8}$$

is small (only t columns). Indeed, $\hat{H}$ differs from $[JP_S, 0]$ by a matrix of rank t. Hence, $\hat{H}$-projection can be performed easily if $[JP_S, 0]$-projection can. In the next section, we will generally show the structure of JP_S only.

5. Examples.

Here we present three situations where M-projection is simple and the parametric linear programming problem $(SP''(z))$ is easily solved. We also point out an example where the approach of section 3 does not appear fruitful. The favorable cases are of upper-bounding type, while the unfavorable case involves an embedded network.

5.1. Simple Upper Bounds.

Suppose that, in $(\tilde{P})$, the variables $\tilde{x}$ are partitioned into

$$\tilde{x}^T = (\tilde{x}_1^T, \tilde{x}_2^T, \tilde{x}_3^T), \qquad (5.1)$$

where $\tilde{x}_1$ consists of structural variables with upper bounds u, $\tilde{x}_2$ the corresponding slack variables, and $\tilde{x}_3$ the unbounded structural variables. We assume that variables $\tilde{x}_2$ appear only in the bounding constraints. After transforming to a problem of form (P_0) (we assume that an artificial variable is not needed) we find

$$H_0 = [H_{10}, 0, H_{30}, -b],$$

$$M_0 = [I, \quad I, 0, \quad -u],$$

where H_0 corresponds to the general and M_0 to the upper-bounding constraints. At a later iteration we have

$$H = [H_1, 0, H_3, -b], \tag{5.2}$$

$$M = [X_1, X_2, 0, -u], \tag{5.3}$$

where X_1 and X_2 are diagonal matrices whose diagonal entries are the current components of $\tilde{x}_1$ and $\tilde{x}_2$ respectively. (We are abusing notation somewhat in using subscripts for iteration numbers as well as partitioning, but no confusion should result since we drop the subscript for the iteration.) Let J and S consist of all but the last column of H and M, as in section 4. Then, setting

$$Z = [X_1^2 + X_2^2]^{-1}, \tag{5.4}$$

it is easy to calculate

$$P_S = \begin{bmatrix} X_2ZX_2 & -X_1ZX_2 & 0 \\ -X_2ZX_1 & X_1ZX_1 & 0 \\ 0 & 0 & I \end{bmatrix} \qquad (5.5)$$

(note that all blocks are diagonal) and hence

$$JP_S = [H_1X_2ZX_2,\ -H_1X_1ZX_2,\ H_3]. \qquad (5.6)$$

Hence JP_S differs from J only in that the initial columns H_1 are scaled, and a column scaling of H_1 also appears in the zero block of H. Using this structure, we see that finding $(JP_S)(JP_S)^T$ or its Cholesky factor, or a QR factorization of $(JP_S)^T$, is essentially no more costly than an equivalent operation on J. If $H_1 = H_{10}X_1$ and $H_3 = H_{30}X_3$, then

$$(JP_S)(JP_S)^T = H_{10}\, D_1^2\, H_{10}^T + H_{30}\, X_3^2\, H_{30}^T \qquad (5.7)$$

where

$$D_1 = X_1Z^{1/2}X_2, \qquad (5.8)$$

while

$$JJ^T = H_{10}\, X_1^2\, H_{10}^T + H_{30}\, X_3^2\, H_{30}^T. \qquad (5.9)$$

The diagonal matrix D_1 above corresponds to the scaling used in Gill, Murray, Saunders, Tomlin and Wright [10], while other authors have proposed a simpler choice of diagonal weighting,

$$D_1' = \min\{X_1, X_2\} \qquad (5.10a)$$

where the minimum is component-wise (see e.g. [26]) or

$$D_1'' = X_1(X_1+X_2)^{-1}X_2, \qquad (5.10b)$$

see [3].

The parametric problem $(SP'''(z))$ is a linear programming problem subject only to bounds, and hence is trivially solvable.

5.2. Generalized Upper Bounds.

Next we consider generalized upper bounds, a set of constraints, typically with nonnegative coefficients, involving disjoint sets of variables (Dantzig and Van Slyke [6]). Collecting these as the special constraints, ℓ in number, we will have at a typical iteration

$$H = [H_1, H_2, \ldots, H_\ell, H_{\ell+1}, -b] \qquad (5.11)$$

$$M = \begin{bmatrix} f_1^T, 0, \ldots, 0, 0, & -u_1 \\ 0, f_2^T, \ldots, 0, 0, & -u_2 \\ & \\ 0, 0, \ldots, f_\ell^T, 0, & -u_\ell \end{bmatrix} . \qquad (5.12)$$

If J and S consist of all but the last columns of these, we easily find

$$P_S = \begin{bmatrix} I-f_1f_1^T/f_1^Tf_1 & & & 0 \\ & \ddots & & \\ & & I-f_\ell f_\ell^T/f_\ell^Tf_\ell & \\ 0 & & & I \end{bmatrix} \quad (5.13)$$

and

$$JP_S = [H_1-H_1f_1f_1^T/f_1^Tf_1,\ldots,H_\ell-H_\ell f_\ell f_\ell^T/f_\ell^Tf_\ell,H_{\ell+1}]. \quad (5.14)$$

Thus there is a rank-one change to each part of H corresponding to a GUB-set, a set of variables involved in one generalized upper bound. It is easy to check that, if each GUB-set has just two elements and each f_i is $(1,1)^T$ scaled, then the GUB's reduce to simple upper bounds and the results above to those in section 5.1.

In (5.14), each column of J is replaced by a linear combination of all columns of J corresponding to the same GUB-set. This contrasts with the compact-inverse simplex method for such problems, where the working basis consists of columns that are linear combinations of just two columns of J corresponding to the same GUB-set; see [6].

The parametric problem $(SP'''(z))$ separates into ℓ parametric knapsack problems and one trivial parametric problem with feasible region the nonnegative orthant (corresponding to the variables in no GUB-set). It is therefore easily solved.

5.3. Variable Upper Bounds.

Constraints of the form $\tilde{x}_j \le \tilde{x}_k$ are called variable upper bounds, with the condition that a variable can occur at most once on the left-hand side of such an inequality (as a "child") and then not on the right-hand side of any other (as a "parent"). These constraints occur frequently in linear programming relaxations of integer programming formulations. They were first studied by Schrage [20] in the context of an efficient implementation of the simplex method; see also [21] and [23], where extensions are considered.

Suppose there are ℓ parent variables occurring on the right-hand sides of VUB's. We order the variables as follows: first those bounded by the first parent variable, then those by the second, etc.; then slack variables; then the parents, in order; and finally all remaining variables. Grouping general constraints in H_0 and variable upper bounds in M_0, we have

$$H_0 = [H_{10},\ldots,H_{\ell 0},0,\ldots,0,h_{10},\ldots,h_{\ell 0},H_{\ell+1,0},-b] \qquad (5.15)$$

$$M_0 = \begin{bmatrix} I & & I & & -e & & & \\ & \ddots & & \ddots & & \ddots & 0 & 0 \\ & & I & & I & & -e & \end{bmatrix} \qquad (5.16)$$

where the identity blocks in M_0 as well as the vectors e of ones can have different dimensions. All unmarked blocks in M_0 and M below are zero. At a later iteration, we have

$$H = [H_1,\dots,H_\ell,0,\dots,0,h_1,\dots,h_\ell,H_{\ell+1},-b] \tag{5.17}$$

$$M = \left[\begin{array}{cccccccccc}
X_1 & & & W_1 & & & -f_1 & & & \\
& \ddots & & & \ddots & & & \ddots & 0 & 0 \\
& & X_\ell & & & W_\ell & & & -f_\ell &
\end{array}\right] \tag{5.18}$$

Let $Z_i = (X_i^2 + W_i^2 + f_i f_i^T)^{-1}$; since X_i and W_i are diagonal, Z_i is easy to obtain explicitly. We find $P_M =$

$$\left[\begin{array}{ccccccccccc}
I-X_1Z_1X_1 & & & -X_1Z_1W_1 & & & X_1Z_1f_1 & & & & \\
& \ddots & & & \ddots & & & \ddots & & & \\
& & I-X_\ell Z_\ell X_\ell & & & -X_\ell Z_\ell W_\ell & & & X_\ell Z_\ell f_\ell & & \\
-W_1Z_1X_1 & & & I-W_1Z_1W_1 & & & W_1Z_1f_1 & & & & \\
& \ddots & & & \ddots & & & \ddots & & 0 & \\
& & -W_\ell Z_\ell X_\ell & & & I-W_\ell Z_\ell W_\ell & & & W_\ell Z_\ell f_\ell & & \\
f_1^TZ_1X_1 & & & f_1^TZ_1W_1 & & & 1-f_1^TZ_1f_1 & & & & \\
& \ddots & & & \ddots & & & \ddots & & & \\
& & f_\ell^TZ_\ell X_\ell & & & f_\ell^TZ_\ell W_\ell & & & 1-f_\ell^TZ_\ell f_\ell & & \\
& & & & & & & & & I & \\
& & 0 & & & & & & & & 1
\end{array}\right] \tag{5.19}$$

where all the blocks are single rows, single columns or rank-one modifications of diagonal matrices. It follows that

$$\hat{H} = [\hat{H}_1,\ldots,\hat{H}_\ell,\tilde{H}_1,\ldots,\tilde{H}_\ell,H_{\ell+1},-b],$$

where both $\hat{H}_i$ and $\tilde{H}_i$ are rank-one modifications of column scalings of H_i.

Finally, (SP"'(z)) separates into ℓ parametric problems with only variable-upper-bound constraints and one with just non-negativities, and all of these are trivial to solve.

5.4. Network Constraints.

Now suppose that $(\tilde{P})$ contains flow-conservation constraints in a network. Assuming there are no bounds on the flows, we have

$$M_0 = [N_0, \ -u]$$

where N_0 is a node-edge incidence matrix and u is the vector of net supplies at the nodes. At a later iteration, M will be a column scaling of M_0, but we see no especially efficient way to exploit its structure in finding M-projections.

Once again, (SP"'(z)) can be used to provide improved lower bounds; this is a parametric network flow problem, and hence relatively easy to solve, although not as trivial as the subproblems arising in sections 5.1-5.3. Observe that, as in the discussion above lemma 3.1, (SP"(z)) would be unchanged if its objective function were replaced by $(c_q - zg_q)^T x$; similarly we can use $(\tilde{c}_q - z\tilde{g}_q)^T x$ in (SP"'(z)), where $\tilde{c}_q = X_k^{-1} c_q$, $\tilde{g}_q = X_k^{-1} g_q$. Thus it is unnecessary to compute c_r or g_r in order to obtain an improved bound. On the other hand, it is unclear whether such improved bounds, possibly saving 1 to 3 iterations, are worth the extra work of solving say 15 parametric network flow problems.

5.5 Conclusions.

The analysis of this and previous sections indicates that special structure in the constraints of a linear programming problem can be exploited in Karmarkar's algorithm. However, the examples above show that unless the structure is very special, the methods proposed are not likely to provide nearly as large an improvement as can be obtained for the simplex method. Clearly, computational testing will be required to evaluate the benefits arising from exploiting special structure in this way.

References

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

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

[3] V. Chandru and B. Kochar, "A class of algorithms for linear programming," manuscript, Department of Industrial Engineering, Purdue University, West Lafayette, IN (1985).

[4] G.B. Dantzig, Linear programming and extensions (Princeton University Press, Princeton, NJ, 1983).

[5] G.B. Dantzig, M.A.H. Dempster, and M.J. Kallio, eds., Large scale linear programming (IIASA, Laxenburg, Austria, 1981).

[6] G.B. Dantzig and R.M. Van Slyke, "Generalized upper bounding techniques," Journal of Computer System Sciences 1 (1967) 213-226.

[7] G.B. Dantzig and P. Wolfe, "The decomposition algorithm for linear programming," Econometrica 29 (1961) 767-778.

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

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

[10] P.E. Gill, W. Murray, M.A. Saunders, J.A. Tomlin and M.H. Wright, "On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method," Mathematical Programming 36 (1986) 183-209.

[11] C. Gonzaga, "A conical projection algorithm for linear programming," manuscript, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA (1985).

[12] G.W. Graves and R.D. McBride, "The factorization approach to large-scale linear programming," Mathematical Programming 20 (1976) 91-110.

[13] D. Jensen and A. Steger, private communication, Department of Applied Mathematics and Statistics, State University of New York at Stonybrook, Stonybrook, New York (1985).

[14] S. Kapoor and P.M. Vaidya, "Fast algorithms for convex quadratic programming and multicommodity flows," Proceedings of the 18th ACM Symposium on Theory of Computing (1986) 147-159.

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

[16] N. Karmarkar and L.P. Sinha, "Application of Karmarkar's algorithm to overseas telecommunications facilities planning," paper presented at XII International Symposium on Mathematical Programming, Boston (1985).

[17] L.S. Lasdon, Optimization theory for large systems (Macmillan, New York, 1970).

[18] B.A. Murtagh, Advanced linear programming: computation and practice (McGraw-Hill, New York, 1981).

[19] G. Rinaldi, "A projective method for linear programming with box-type constraints," Algorithmica 1 (1986) 517-527.

[20] L. Schrage, "Implicit representation of variable upper bounds in linear programming," Mathematical Programming Study 14 (1975) 118-132.

[21] L. Schrage, "Implicit representation of generalized variable upper bounds in linear programming," Mathematical Programming 14 (1978) 11-20.

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

[23] M.J. Todd, "An implementation of the simplex method for linear programming problems with variable upper bounds," Mathematical Programming 23 (1982) 34-49.

[24] M.J. Todd, "Improved bounds and containing ellipsoids in Karmarkar's linear programming algorithm," to appear in Mathematics of Operations Research.

[25] M.J. Todd and B.P. Burrell, "An extension of Karmarkar's algorithm for linear programming using dual variables," Algorithmica 1 (1986) 409-424.

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

[27] Y. Ye and M. Kojima, "Recovering optimal dual solutions in Karmarkar's polynomial algorithm for linear programming," to appear in Mathematical Programming.