MODIFICATIONS AND IMPLEMENTATION OF THE SHOR-KHACHIAN ALGORITHM FOR LINEAR PROGRAMMING†

by

Donald Goldfarb*

and

Michael J. Todd

TR 80-406

Department of Computer Science
Cornell University
Ithaca, New York 14853

†This report is also available as Technical Report 446 from the School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853.

*The work of this author was supported in part by the U.S. Army Research Office under Grant DAAG 29-77-G-0114.

Modifications and Implementation of the
Shor-Khachian Algorithm for Linear Programming

by Donald Goldfarb and Michael J. Todd

Abstract. We give some modifications of the recent Shor-Khachian algorithm for linear programming and describe a numerically stable implementation. We are concerned with practical problems where user-supplied bounds can usually be provided. Our implementation allows constraint dropping and updates bounds on the optimal value, and should be able to terminate with an indication of infeasibility or with a provably good feasible solution in a moderate number of iterations.

1. Introduction

We are concerned here with efficient implementation of the Shor-Khachian algorithm for linear programming. This algorithm was first introduced by Shor* [23] to solve convex optimization problems. Shor showed that if an a priori bound could be given for the distance from an initial trial point to an optimal solution, then a sequence $\{E_k\}$ of ellipsoids could be constructed, each containing an optimal solution. He showed that the volume of the E_k's decreased at a certain rate depending only on the dimension n and not on the particular functions involved. Khachian [13] applied this algorithm to the solution of a system of linear inequalities. He showed how an a priori bound for the distance of a solution from the origin, if a solution existed, could be derived from the original data. He also demonstrated that after slightly perturbing the right hand sides of the inequalities, an a priori positive lower bound on the volume of the feasible region could also be obtained. Combining these bounds with the rate at which the volumes of the E_k were shrinking gave a polynomial bound for the number of iterations to determine whether such a system of inequalities had a solution or not. Khachian also discussed how such an algorithm could be implemented on a finite-precision machine.

Gacs and Lovasz [6] expounded the algorithm in a rather simpler form and gave proofs for the claims made in Khachian's paper. They showed precisely how the perturbations to the original inequalities to guarantee a positive volume could be removed to give a solution to the original problem. They also stated explicitly one way in which linear programming problems could be reduced to solving linear inequalities. Gacs and Lovasz assumed in their

---

*Those searching the literature are advised that Shor's name is frequently transliterated as Šor, e.g. in <u>Math. Reviews</u>.

proofs that exact arithmetic was used, though they noted that the results remained true with slight modifications for finite precision.

The discovery of a polynomial algorithm for linear programming solved an important theoretical question. Previously it had only been known that linear programming (or, to be more precise, the feasibility problem for linear inequalities) lay in the class NP of problems solvable in polynomial time by a nondeterministic Turing machine. This class of problems includes some notoriously hard combinatorial problems for which it is strongly suspected that no (deterministic) algorithms terminating in less than an exponential number of steps exist. However, it seemed clear that linear programming was much easier than these hard problems. For one thing, linear programming was also in the class co-NP of problems, and it was strongly suspected that NP $\cap$ (co-NP) was much smaller than NP, and consisted of much easier problems. Secondly, linear programming is closely related to solving linear equations, for which polynomial algorithms exist. Finally, Dantzig's well-known simplex algorithm performed very well on almost all LP problems, although there were some for which it required an exponential number of iterations [14].

The bound for the Shor-Khachian algorithm is of the order of $6n^2L$ iterations, where n is the dimension of the problem and L the length of the input data in bits. Each iteration requires $O(mn+2n^2)$ operations, where m is the number of inequalities in the system. The presence of the factor L in the number of iterations is rather unsatisfactory. Of course, it is understood that the size of the numbers involved must affect the running time of the algorithm, but one might hope that the dependence would only be through the time to perform the individual sums, products, etc.

The L appears in the iteration bound for two reasons: the initial ellipsoid E_0 depends on the a priori bound on the norm of a solution to the inequalities, which in turn depends on L; and the algorithm can be terminated with an indication of infeasibility only when the volume of E_k has shrunk to an amount smaller than the a priori lower bound on the volume of feasible solutions, which also depends on L.

This paper is concerned with an efficient and practical implementation of a modification of the Shor-Khachian algorithm. We wish to solve

$$\text{minimize } c^Tx \text{ subject to } x \in X \equiv \{y \in R^n | a_i^Ty \leq \beta_i, \ i = 1,2,\ldots,m\}. \quad (1.1)$$

We do not wish to assume that the initial data is integer, and we will operate in floating-point arithmetic. We cannot therefore provide a priori bounds based on the data. Instead we will assume that the user can provide bounds on the solution to (1.1), possibly in the form of upper and lower bounds on the variables. Our algorithm need not be run until the volume of the ellipsoid E_k is almost infinitesimal. Instead we will use bounds generated during its progress to halt the algorithm when it has obtained a feasible solution provably close to an optimal solution in objective function value.

The remainder of the paper is organized as follows. In section 2 we discuss the basic operation of the algorithm, the generation of a new ellipsoid E_+ from a previous one E and a "cut" giving a half-space H, so that $E_+ \supseteq E \cap H$. We describe how E_+ can be obtained even if the center of E does not lie on the bounding hyperplane of H (deeper cuts). We also show how bounds on linear forms over E can be used to eliminate constraints that are not active at the optimal solution. Finally we show

how linear inequalities can sometimes be combined (surrogate cuts) and a particular "optimal" combination (best surrogate cut) can be formed under certain conditions. Section 3 outlines our approach towards incorporating the objective function in our algorithm. Since we work in floating-point arithmetic (double-precision is recommended) it is important to use methods guaranteeing numerical stability. Sections 4 and 5 describe how our method uses and updates a Cholesky factorization of a matrix determining the ellipsoid at each iteration. Section 6 describes our algorithm and gives a theorem on the number of iterations required for a certain degree of approximation. Finally in section 7 we suggest a method of dealing with problems with equality constraints. Most of our analysis (sections 2-5) is applicable also to convex (not necessarily differentiable) optimization when subgradients are available--see [23].

We conclude this section with an attempt to put the Shor-Khachian algorithm in some historical perspective. It is, in fact, a fairly natural outgrowth of a substantial literature mainly concerned with subgradient methods for nonlinear programming. These methods grew out of the relaxation method for linear inequalities introduced by Agmon [1] and Motzkin and Schoenberg [15]. Agmon demonstrated linear convergence of his method, by showing that each iterate came closer by some fixed ratio to the set of feasible solutions. The similarity to the Shor-Khachian algorithm is striking; a sequence of shrinking balls $\{S_k\}$ centered at the current iterates x_k could be defined, each containing a feasible solution. The differences were that no a priori bounds were given on S_0 or the volume of the feasible region and (most importantly) that the ratio of shrinkage depended on the data of the original problem. (For details on this ratio, see Agmon [1], Hoffman [12], Goffin [10] and Todd [24].) The subgradient method for minimizing a convex, not necessarily differentiable function was apparently first introduced by Shor [19], with important refinements by Ermolev [4] and Polyak [16]. (A good selection of references to this area appears in the bibliography of [2].) Later versions (Polyak [17], Shor [20]) attained linear convergence rates for "most" problems, but again the ratio depended on the particular function involved. Shor [21,22] seems to have been the first to observe that improvements could be made by working in a transformed space. The idea is exactly that which leads from the steepest descent algorithm (with linear convergence rate, the ratio depending on the function) to Newton's method (with quadratic convergence for smooth functions) and quasi-Newton algorithms (with superlinear convergence for smooth functions). Shor [21] describes precisely the difficulty with the linear convergence of his earlier method (see the discussion of "essentially gully functions" on page 7 of [21]). He provides a modified algorithm based on shrinking ellipsoids whose convergence rate depends on a ratio of two numbers M and N. These numbers do depend on the function involved, but they are invariant with respect to linear transformations. If the function is quadratic and strictly convex, then M and N can be taken as 2 and Shor's method [21] in its limit becomes a method of conjugate gradients--see p. 14 of [21]. Shor's most recent paper [23] builds on [21,22,27] and provides, as noted before, a sequence of ellipsoids shrinking at a linear rate depending only on the dimension n.

2. Cuts

Let $\hat{E}$ denote the unit ball in R^n $\{\hat{x} \mid \|\hat{x}\| \leq 1\}$ (all norms are Euclidean). Consider a half-space which we will denote $\hat{H} = \{\hat{x} \in R^n \mid \hat{a}^T\hat{x} \leq -\alpha(\hat{a}^T\hat{a})^{1/2}\}$. Note that the distance from the origin to the bounding hyperplane of $\hat{H}$ is $|\alpha|$. If $\alpha \leq -1$, $\hat{E} \subset \hat{H}$; if $\alpha > 1$, $\hat{E} \cap \hat{H} = \emptyset$; and if $-1 < \alpha \leq 1$, $\hat{E} \cap \hat{H}$ is a nonempty slice of the unit ball. We assume now that $-1 < \alpha < 1$.

The point where $\hat{a}^T\hat{x}$ is minimized over $\hat{E}$ is $-\hat{a}/(\hat{a}^T\hat{a})^{1/2}$. Consider all possible ellipsoids $\hat{E}_+ = \{\hat{x} \in R^n \mid (\hat{x}-\hat{x}_1)^T\hat{B}_+^{-1}(\hat{x}-\hat{x}_1) \leq 1\}$, where $\hat{x}_1 \in R^n$ is the center and $\hat{B}_+$ a positive definite symmetric matrix, which satisfy the two conditions:

($\hat{\text{i}}$) $\hat{a}^T\hat{x}$ is minimized over $\hat{E}_+$ by $-\hat{a}/(\hat{a}^T\hat{a})^{1/2}$;

($\hat{\text{ii}}$) $\hat{E}_+ \cap \partial\hat{H} = \hat{E} \cap \partial\hat{H}$, where $\partial\hat{H}$ denotes the bounding hyperplane of $\hat{H}$.

It is easy to see from ($\hat{\text{i}}$) and ($\hat{\text{ii}}$) that $\hat{B}_+$ has $\hat{a}$ as an eigenvector, that all other eigenvectors have the same eigenvalue associated with them, and that $\hat{x}_1$ is a multiple of $\hat{a}$. Thus we may set $\hat{x}_1 = -\lambda\hat{a}/(\hat{a}^T\hat{a})^{1/2}$, and $\hat{B}_+ = \delta(I - \sigma\hat{a}\hat{a}^T/\hat{a}^T\hat{a})$. Conditions ($\hat{\text{i}}$) and ($\hat{\text{ii}}$) then imply that $\delta = (1-\lambda)^2(1+\alpha)/(1+\alpha-2\lambda)$ and $\sigma = 2\lambda/(1+\alpha)$, so that varying λ gives one degree of freedom. We will choose λ to minimize the volume of $\hat{E}_+$.

If σ is less than 1 and δ is positive, $\hat{B}_+$ is positive definite and can be decomposed as $\hat{B}_+ = \hat{J}_+\hat{J}_+^T$, with $\hat{J}_+$ nonsingular. Then $\hat{E}_+$ is the image of the unit ball under the transformation $z \to \hat{x}_1 + \hat{J}_+ z = \hat{x}$. Thus the volume of $\hat{E}_+$ is that of the unit ball times the Jacobian of the transformation, $\det \hat{J}_+ = \sqrt{\det \hat{B}_+}$. We wish therefore to minimize $\det \hat{B}_+$ which is the product of the eigenvalues of $\hat{B}_+$; these are δ with multiplicity $n-1$ and $\delta(1-\sigma)$ with multiplicity 1. It is now a simple matter of calculus to determine the ellipsoid of minimum volume; we set $\lambda = (1+n\alpha)/(1+n)$ and hence $\delta = n^2(1-\alpha^2)/(n^2-1)$ and $\sigma = 2(1+n\alpha)/(n+1)(1+\alpha)$. It can also be confirmed that if $\alpha \geq -1/n$, $\hat{E} \cap \hat{H} \subseteq \hat{E}_+$. From the analysis above, the volume of $\hat{E}_+$ is smaller than that of $\hat{E}$ by a ratio

$$r(\alpha) = \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} \frac{n}{n+1} (1-\alpha^2)^{\frac{n-1}{2}} (1-\alpha).$$

Now consider an affine transformation $\hat{x} \to x_0 + J\hat{x} = x$ of R^n, with J a nonsingular $n\times n$ matrix. Under this transformation, $\hat{E}$ becomes the ellipsoid

$$E = \{x \in R^n | (x-x_0)^T B^{-1}(x-x_0) \leq 1\} \tag{2.1}$$

where $B = JJ^T$. The halfspace $\hat{H}$ becomes the halfspace

$$H = \{x \in R^n | a^T(x-x_0) \leq -\alpha(a^TBa)^{1/2}\}, \tag{2.2}$$

where $a = J^{-T}\hat{a}$ (so that $a^TBa = \hat{a}^T\hat{a}$).

We now find that a^Tx is minimized over E by $x_0 - Ba/(a^TBa)^{1/2}$, the image of $-\hat{a}/(\hat{a}^T\hat{a})^{1/2}$. We may again consider all ellipsoids

$$E_+ = \{x \in R^n | (x-x_1)^T B_+^{-1}(x-x_1) \leq 1\} \tag{2.3}$$

that satisfy

(i) $a^T x$ is minimized over E_+ by $x_0 - Ba/(a^T Ba)^{1/2}$,

(ii) $E_+ \cap \partial H = E \cap \partial H$. (2.4)

These ellipsoids will be exactly the images of the ellipsoids $\hat{E}_+$ satisfying ($\hat{\text{i}}$) and ($\hat{\text{ii}}$). Since the transformation multiplies all volumes by the constant $|\det J|$, the choice of B_+ and x_1 to minimize the volume of E_+ must be $J\hat{B}_+J^T$ and $x_0 + J\hat{x}_1$. We thus obtain

<u>Theorem 2.1</u>. Of all the ellipsoids of the form (2.3) that satisfy (2.4)(i) and (ii), with E given by (2.1) and H by (2.2) where $-1 < \alpha < 1$, that with the smallest volume is given by

$$x_1 = x_0 - \frac{(1+n\alpha)}{(n+1)} \frac{Ba}{(a^T Ba)^{1/2}} \tag{2.5}$$

$$B_+ = \delta(B - \sigma \frac{Baa^T B}{a^T Ba}), \tag{2.6}$$

where

$$\delta = \frac{n^2}{n^2-1}(1-\alpha^2) \quad \text{and} \quad \sigma = \frac{2(1+n\alpha)}{(n+1)(1+\alpha)}. \tag{2.7}$$

The ratio of the volume of this E_+ to that of E is

$$r(\alpha) = \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}} \frac{n}{n+1} (1-\alpha^2)^{\frac{n-1}{2}} (1-\alpha). \tag{2.8}$$

Moreover, $r(\alpha)$ is increasing for $-1 < \alpha \leq -\frac{1}{n}$ up to 1, and decreasing for $-\frac{1}{n} \leq \alpha < 1$. For $\alpha \geq -\frac{1}{n}$, the ellipsoid E_+ given by (2.3), (2.5)-(2.7) contains $E \cap H$. For $\alpha = \frac{1}{n}$, $E_+ = E$.

We will not give a complete proof of the theorem, since the analysis above clearly indicates all the steps. The formulae (2.5)-(2.7) for the case $\alpha = 0$ were presented by Gacs and Lovasz [6] to describe the Shor-Khachian algorithm.

Several workers (including ourselves and Gacs and Lovasz) have obtained independently the formulae above for $\alpha > 0$. The case $\alpha < 0$ has been alluded to in Yudin and Nemirovskii [27].

Now suppose it is known that the polyhedron

$$X = \{x \in R^n | a_i^T x \leq \beta_i, \quad i = 1,\ldots,m\} \tag{2.9}$$

is contained in E, but the center x_0 of E is not in X. We can define, for each i, the number

$$\alpha_i = (a_i^T x_0 - \beta_i)/(a_i^T B a_i)^{1/2} \tag{2.10}$$

Then $\alpha_i > 0$ for those constraints i violated at x_0, and α_i represents the (algebraic) distance from x_0 to the bounding hyperplane of the half-space $H_i = \{x \in R^n | a_i^T x \leq \beta_i\}$ in the metric corresponding to the matrix B. The following observation is crucial:

Proposition 2.2. If $\alpha_i < -1$, $E \subset H_i$ and the ith inequality is inessential in defining X. If $\alpha_i > 1$, $E \cap H_i$ is empty and thus X is empty. If, for each $-\frac{1}{n} \leq \alpha_i < 1$, we determine the ellipsoid E_+^i as in Theorem 2.1 using a_i and α_i, then E_+^i has the smallest volume when i is chosen to maximize α_i. If $\alpha_i = 1$, $E \cap H_i$ degenerates to a point.

It is also possible to choose some nonnegative combination of violated inequalities to form the cut; examples show that in some cases a deeper cut will result. We now make this precise.

Let $\bar{A}$ be a matrix whose columns are some subset of the a_i's for which $0 < \alpha_i < 1$, and let $\bar{b}$ be the corresponding vector of β_i's. Then the inequality $u^T\bar{A}^T x \leq u^T\bar{b}$ holds for any $x \in X$ and any $u \geq 0$. If $a = \bar{A}u$ and $\beta = u^T\bar{b}$, we want to choose $u \geq 0$ so that

$$\alpha = (a^T x_0 - \beta)/(a^T B a)^{1/2} = u^T(\bar{A}^T x_0 - \bar{b})/(u^T\bar{A}^T B\bar{A}u)^{1/2} \tag{2.11}$$

is maximized. Assume that $\bar{A}$ has full column rank; then α is maximized over all u (not just nonnegative u's) by the vector $u = (\bar{A}^T B\bar{A})^{-1}(\bar{A}^T x_0 - \bar{b})$, as a simple application of Lagrange multipliers verifies. Todd [24] has shown that $\bar{A}^T B\bar{A}$ is positive definite and has nonnegative inverse if X is nonempty and $\bar{A}^T B\bar{A}$ has nonpositive off-diagonal entries*; since $\bar{A}^T x_0 > \bar{b}$, this implies that u as given above is nonnegative. (In fact, the proof in [24] is for the case $B = I$; for the general case, write $\bar{A}^T B\bar{A}$ as $(J^T\bar{A})^T(J^T\bar{A})$ where $B = JJ^T$.) The condition that $\bar{A}^T B\bar{A}$ have nonpositive off-diagonal entries means that, in terms of the current metric given by B, the constraint normals in $\bar{A}$ are mutually obtuse. Summarizing, we have:

<u>Theorem 2.3</u>. Suppose X in (2.9) is nonempty and contained in E given by (2.1). Let I be a set of indices such that $a_i^T x_0 > \beta_i$ for each $i \in I$ and $a_i^T B a_j \leq 0$ for each $i \neq j$, $i,j \in I$. Let $\bar{A}$ be the matrix whose columns are those a_i's with $i \in I$ and $\bar{b}$ be the corresponding vector of β_i's. Then the cut induced by $a = \bar{A}u$, $\beta = u^T\bar{b}$, $u \geq 0$, is deepest, i.e. α in (2.11) is maximized, when

$$u = (\bar{A}^T B\bar{A})^{-1}(\bar{A}^T x_0 - \bar{b}). \tag{2.12}$$

---

*Such a matrix is called a Stieltjes-matrix; e.g., see [25].

We call any cut using $a = \bar{A}u$ and $\beta = u^T\bar{b}$ with $u \geq 0$ a "surrogate" cut; if the u given by (2.12) is nonnegative, the cut is called the "best surrogate cut." Theorem 2.3 gives sufficient conditions for u in (2.12) to be nonnegative. However, it is important to note that for any set I if (2.12) is nonnegative then the surrogate cut based upon this u is both valid and deepest with respect to that set.

Of course, computationally, it is expensive to search for a large set I satisfying the hypotheses of theorem 2.3, and one does not wish to solve a large system of equations to obtain u as in (2.12). However, it is simple to check a 2-element set I and in this case it is easy to obtain u.

Let us note another justification for finding such an index set I. Suppose I satisfies the hypotheses above. If we use a cut based on a single a_i, then x_0 moves to $x_1 = x_0 - \lambda B a_i$, some $\lambda > 0$. Since $a_j^T B a_i \leq 0$ for all $j \neq i$ in I, x_1 violates all inequalities indexed by $I \backslash \{i\}$, indeed by more than x_0. Using the formulae (2.6) - (2.7) we find that $a_j^T B_+ a_k < \delta a_j^T B a_k \leq 0$ for all $j \neq k$ in I. Further, it can be shown that the constraint normals in $\bar{A}$ other than a_i are more obtuse in the new metric B_+ than they are in the old metric B. Hence successive additions of cuts, each based on a single a_j, $j \in I$, will worsen all the remaining inequalities in I. It follows that at least $|I|$ cuts will be necessary to attain feasibility in all these constraints. On the other hand, the cut based on $a = \bar{A}u$ and $\beta = u^T\bar{b}$, with u as in (2.12), gives a point x_1 satisfying all these constraints, as is easily seen.

3. Objective Function Cuts

If $\bar{x}$ is any point in the feasible region $X = \{x \in R^n | a_i^T x \leq \beta_i, i = 1,\ldots,m\} \subset E$, where the ellipsoid E is defined by (2.1), then an optimal solution to (1.1) must lie in $E \cap H_0$ where

$$H_0 = \{x \in R^n | c^T x \leq \bar{z}\} = \{x \in R^n | c^T(x-x_0) \leq -\alpha_0(c^TBc)^{1/2}\}, \quad (3.1)$$

$$\alpha_0 = (c^Tx_0-\bar{z})/(c^TBc)^{1/2} \quad (3.2)$$

and

$$\bar{z} = c^T\bar{x}. \quad (3.3)$$

As in the case of the inequality cuts corresponding to H_i, $i = 1,\ldots,m$, we would like to make α_0 as large as possible by choosing $\bar{x}$ so that $\bar{z}$ is as small as possible. If x_0, the center of E, is feasible, then choosing $\bar{x} = x_0$ gives $\alpha = 0$ and H_0 above yields a standard Shor-Khachian cut. Clearly, a deeper cut can be obtained by moving from x_0 in a direction s which is downhill with respect to z. Specifically, let

$$\bar{x} = x_0 + \theta s \quad (3.4)$$

where $s^Tc < 0$ and θ is a scalar ≥ 0. For a particular downhill direction, s, the deepest cut is obtained by choosing θ as large as possible subject to the restriction that $\bar{x} = \bar{x}(\theta)$ in (3.4) remains feasible. This gives

$$\theta = \min_{a_i^Ts>0} \{-(a_i^Tx_0-\beta_i)/a_i^Ts\} \geq 0 \quad (3.5)$$

corresponding to the point where the ray $\{\bar{x}|\bar{x} = x_0 + \theta s, \theta \geq 0\}$ first violates one of the feasibility constraints.

The natural choices for s are $-c$, the direction of steepest descent and $-Bc$, the direction of steepest descent in the transformed space $\{\hat{x} \in R^n | \hat{x} \leftarrow J^{-1}(x-x_0)\}$. Notice that the latter is a Newton-like direction and is the direction along which the center of the new ellipsoid E_+ lies. (See equation (2.5) with a replaced by c and α by α_0.) Either of these directions may result in the deeper cut. If we use the notation $\theta = \theta(s)$ to indicate the dependence of θ in (3.5) upon the choice of s, we have that

$$\alpha_0 = \max\{\frac{\theta(-c)c^Tc}{(c^TBc)^{1/2}}, \quad \theta(-Bc)(c^TBc)^{1/2}\}$$

and

$$\begin{aligned}\bar{z} &= c^Tx_0 - \alpha_0(c^TBc)^{1/2} \\ &= c^Tx_0 - \max\{\theta(-c)c^Tc, \quad \theta(-Bc)c^TBc\}.\end{aligned}$$

for the deeper of the cuts obtained by using s equal to $-c$ or $-Bc$.

Even deeper cuts can be obtained by projecting the direction s onto the hyperplane (with normal a_i) which first becomes violated at $x_0 + \theta s$ and moving from that point in the direction $Ps = s - a_i(a_i^Ts)/(a_i^Ta_i)$ where P is a projection operator, until a second constraint is violated. This process can be repeated until the descent direction obtained is the zero vector. Several variations of this approach are possible. A projection could be made only if the new direction obtained is sufficiently downhill and the cosine of the angle between it and the current direction is close enough to 1. Constraints could be dropped as well as added, leading to a gradient projection type of approach (see [11] and [18]), and ultimately, once a vertex is reached, to the simplex method. Finally we note that when $s = -c$ it is natural to use orthogonal projections but when $s = -Bc$, a projection weighted by the matrix B -- i.e., $P = I - B\bar{A}(\bar{A}^TB\bar{A})^{-1}\bar{A}^T$, where $\bar{A}$ is the matrix of active constraint normals -- should be used.

4. Using the LDL^T Factorization

In order to use the cuts of sections 2 and 3 to move from ellipsoid E to ellipsoid E_+, we must calculate Ba and $(a^TBa)^{1/2}$ and use these to update the center of the ellipsoid. However there is no need to update the matrix B; indeed doing so causes round-off error that can lead to the computed quantity a^TBa being zero or negative. Here we show how a factorization of B can be used and updated.

Any positive definite matrix B can be written as

$$B = LDL^T \tag{4.1}$$

with L a lower triangular matrix with unit diagonal and D a diagonal matrix with positive diagonal (e.g., see [26]). Obtaining such a factorization for an $n \times n$ matrix B requires in general about $n^3/6$ additions and multiplications. We show in the next section that this work can be avoided by updating the factors L and D at each iteration using only $O(n^2)$ additions and multiplications.

If several cuts based on different a_i's are possible, we would like to compute

$$\alpha_i = (a_i^T x - \beta_i)/(a_i^T B a_i)^{1/2} \tag{4.2}$$

for each i, and then select that a_i with maximal α_i. To avoid the main part of the work of computing each α_i at each iteration, we will also update estimates of the quantities $a_i^T B a_i$ at each iteration. These formulae again are given below, and we assume they have been used to determine the cut to be used.

We therefore have $B = LDL^T$ with L and D as above, and $a \in R^n$. The operations performed are then the following. Compute

$$\hat{a} = L^T a \quad \text{and} \quad \gamma = \hat{a}^T D\hat{a}; \tag{4.3}$$

note that even though there are errors in the computed $\hat{a}$, γ will be a positive quantity as long as a is nonzero. Also, $\gamma = a^T Ba$; while we may have an updated estimate of this number, it should be computed afresh for the particular cut used. Next we compute

$$\alpha = \frac{a^T x_0 - \beta}{\gamma^{1/2}}, \quad v = D\hat{a}/\gamma^{1/2} \quad \text{and} \quad w = Lv. \tag{4.4}$$

From these we obtain the new center of the ellipsoid

$$x_1 = x_0 - \frac{1+n\alpha}{n+1} \frac{Ba}{(a^T Ba)^{1/2}} = x_0 - \frac{1+n\alpha}{n+1} w \tag{4.5}$$

as given in (2.5).

Before describing the update of the matrix factors, we show how to obtain $\gamma_i^+ = a_i^T B_+ a_i$ from $\gamma = a_i^T Ba_i$. Indeed we have from (2.6), (4.3) and (4.4) that

$$\begin{aligned} \gamma_i^+ = a_i^T B_+ a_i &= \delta(a_i^T Ba_i - \sigma(a_i^T Ba)^2/(a^T Ba)) \\ &= \delta(\gamma_i - \sigma(a_i^T w)^2), \end{aligned} \tag{4.6}$$

where δ and σ are given in (2.7). These estimates γ_i^+ are used at the next iteration to choose a suitable cut. The accumulation of roundoff errors may eventually cause some of these γ_i to be inaccurate and even become negative. However, as long as these are computed afresh for effecting a cut, or when terminating (some $\alpha_i > 1$), or dropping the ith constraint from further consideration $(\alpha_i < -1)$, any inaccuracy in the γ_i's cannot cause failure of the algorithm; it can only affect which cuts are chosen.

To prevent any γ_i from becoming less than or equal to zero we can use the fact that

$$\gamma_i \geq d_{\ell(i)} a_{i,\ell(i)}^2 \tag{4.7}$$

where $a_{i,\ell(i)}$ is the last nonzero component of a_i and $d_{\ell(i)}$ is the corresponding diagonal element of D. Thus, whenever (4.7) is violated, γ_i can be computed afresh from its definition.

5. Updating the LDL^T Factorization

On each iteration, the matrix B is modified by a rank-one correction and then scaled according to (2.6). Since the resulting matrix B_+ is known, a priori, to be positive definite, there exist several methods for computing the factors of B_+ given that

$$B = LDL^T$$

(e.g. see Bennett [3], Gentleman [7], Gill, Golub, Murray and Saunders [8], Gill, Murray and Saunders [9], and Fletcher and Powell [5]).

From (2.6), we have that

$$B_+ = L(\delta[D-\sigma vv^T])L^T \tag{5.1}$$

where δ and σ are scalars given by (2.7) and v is given by (4.4). Thus we need only compute the factorization

$$D-\sigma vv^T = \tilde{L}\tilde{D}\tilde{L}^T \tag{5.2}$$

to obtain the factors,

$$L_+ = L\tilde{L} \text{ and } D_+ = \delta\tilde{D}, \tag{5.3}$$

of B_+. Using the algorithm of Gill, Murray and Saunders given in section 5.2 of [9], we can compute the diagonal matrix $\tilde{D} = \mathrm{diag}(\tilde{d}_1,\ldots,\tilde{d}_n)$ and the unit lower triangular matrix $\tilde{L} = \{\tilde{\ell}_{ij}\}$ using the recurrence relations:

i) set $t_{n+1} = 1 - \sigma v^T D^{-1} v = 1-\sigma = (\frac{n-1}{n+1})(\frac{1-\alpha}{1+\alpha})$

ii) for $j = n,n-1,\ldots,1$ set

$$\begin{aligned} t_j &= t_{j+1} + \sigma v_j^2/d_j \\ \tilde{d}_j &= d_j t_{j+1}/t_j \\ \beta_j &= -\sigma v_j/(d_j t_{j+1}), \end{aligned} \tag{5.4}$$

where $\tilde{\ell}_{ij} = v_i\beta_j$, $j < i$.

When α is very close to 1 so is σ and $1-\sigma$ may be computed as zero. However, the last expression for t_{n+1} in (5.4)(i) shows that as long as $-1+\mu \leq \alpha < 1-\mu$, where $\mu > 0$ is the machine precision, $t_{n+1} > 0$ and, consequently, $\tilde{D}$ and B_+ are positive definite.

Gill, Golub, Murray and Saunders [8] have shown that the special structure of $\tilde{L}$ can be exploited to efficiently compute the product $L^+ = L\tilde{L}$ in $n^2 + O(n)$ operations. In particular, the following backward recurrence relations can be used to compute L_+ in a numerically stable fashion.

i) Set $q_n^{(n)} = v_n$;

ii) for $j = n-1,\ldots,1$ set

$$q_j^{(j)} = v_j \tag{5.5}$$

$$\left.\begin{aligned} \ell_{ij}^{+} &= \ell_{ij} + \beta_j q_i^{(j+1)} \\ q_i^{(j)} &= q_i^{(j+1)} + v_j \ell_{ij} \end{aligned}\right\} \quad i = j+1,\ldots,n.$$

An alternative approach to storing L and updating it on each iteration is to store a "product form" of L. If we assume that the initial B, B_0, is diagonal (i.e. $L_0 = I$), then after k iterations we have that

$$L_k = \tilde{L}_1 \tilde{L}_2 \ldots \tilde{L}_k. \tag{5.6}$$

If for each $\tilde{L}_j$ we store only the two vectors $v^{(j)}$ and $\beta^{(j)}$, then either $L_k y$ or $L_k^T y$ can be computed in $2k(n-1)$ operations. This follows from the facts that $u = \tilde{L}y$ can be computed from the forward recurrence:

i) set $s = 0$, $u_1 = y_1$;

ii) for $i = 1,\ldots,n-1$ set

$$\begin{aligned} s &= s + \beta_i y_i \\ u_{i+1} &= v_{i+1} s + y_{i+1}, \end{aligned} \tag{5.7}$$

and $u = \tilde{L}^T y$ can be computed from the backward recurrence:

i) set $s = 0$, $u_n = y_n$;

ii) for $i = n, n-1, \ldots, 2$ set

$$s = s + v_i y_i$$
$$u_{i-1} = y_{i-1} + \beta_{i-1} s. \tag{5.8}$$

Assuming that all matrices and vectors are dense (i.e., all elements are nonzero), the total number of operations involving L during the first n iterations is $2n^3 + O(n^2)$ whether L is updated or stored in product form. If less (more) than n iterations are performed, then the product form is less (more) expensive than the updating version. The product of the δ's times the spectral radius of B_0 provides an upper bound ρ_k for the spectral radius of B_k. If $B_0 = \rho_0 I$, and $k < n$, then ρ_k is the spectral radius of B_k. Consequently, if n is large and α is close to one on most iterations so that ρ_k is small, it may be worthwhile resetting B_k to $\rho_k I$ if the product form is used. Although this may increase the number of iterations required to find a solution, it will decrease the work required per iteration. Also, if n is so large that L must be stored on a secondary storage medium then the product form has the advantage that the L file need not be accessed for the updating step.

We also describe here an alternative factorization that can be used and has some advantages when used in product form. This method is basically that of Shor [23] and Khachian [13]. We maintain a factorization $B = \kappa JJ^T$ where κ is a positive scalar and J a nonsingular $n \times n$ matrix. With this factorization we can compute

$$\tilde{a} = J^T a \quad \text{and} \quad \gamma = \kappa \tilde{a}^T \tilde{a}$$

and then

$$\alpha = (a^T x_0 - \beta)/\gamma^{1/2}, \quad \tilde{v} = (\kappa/\gamma)^{1/2} \tilde{a} \quad \text{and} \quad w = \kappa^{1/2} J \tilde{v}.$$

Just as in section 4 we have $\gamma = a^T Ba$ and $w = Ba/\gamma^{1/2}$, so that x_0 can be updated to x_1 and the γ_i's updated as in (4.5)-(4.6). To update the factorization of B_+, note that

$$\begin{aligned} B_+ &= \delta(B - \sigma w w^T) \\ &= \delta(\kappa J J^T - \sigma \kappa J \tilde{v}\tilde{v}^T J^T) \\ &= \delta\kappa J(I - \sigma \tilde{v}\tilde{v}^T) J^T \\ &= (\delta\kappa)[J(I - \pi \tilde{v}\tilde{v}^T)][J(I - \pi \tilde{v}\tilde{v}^T)]^T \end{aligned}$$

where $1-\pi = (1-\sigma)^{1/2} = [(\frac{n-1}{n+1})(\frac{1-\alpha}{1+\alpha})]^{1/2}$. Thus we set

$$\begin{aligned} \kappa_+ &= \delta\kappa \\ J_+ &= J(I - \pi \tilde{v}\tilde{v}^T) = J - \pi\kappa^{-1/2} w \tilde{v}^T. \end{aligned}$$

In this implementation, the work involved per iteration is $3n^2 + O(n)$; n^2 to calculate $\tilde{a}$, n^2 for w and n^2 to obtain J_+. Using the LDL^T factorization, the work involved is only $2n^2 + O(n)$; $n^2/2$ for $\hat{a}$, $n^2/2$ for w and n^2 to update L and D. However, consider now the product form. The LDL^T implementation in product form requires about $4kn$ operations and stores two new vectors $v^{(k)}$ and $\beta^{(k)}$ at the kth iteration. In contrast, consider the κJJ^T implementation above. If B_0 is diagonal, we may choose J_0 to be diagonal also. Then

$$J_k = J_0(I-\pi_1\tilde{v}_1\tilde{v}_1^T)(I-\pi_2\tilde{v}_2\tilde{v}_2^T)\ldots(I-\pi_k\tilde{v}_k\tilde{v}_k^T)$$

where π_j, $\tilde{v}_j$ are the π and $\tilde{v}$ generated at the jth iteration. It can easily be verified that $\tilde{a}$ and w can be computed in about $2kn$ operations each, so that again about $4kn$ operations are required at the kth iteration. However, we now need only store one new vector $\tilde{v}_j$ and scalar π_j at each iteration.

The principal drawback of this factorization is that the accumulation of roundoff errors may cause J to have rank less than n which can result in $\tilde{a} = 0$ and $\gamma = 0$. If such a situation is detected one could always restart using the current x and κ, replacing the current J by the identity matrix I.

6. The Algorithm

We describe here a method to solve or approximately solve the linear programming problem

$$\text{minimize } z = c^T x \text{ subject to } A^T x \leq b. \tag{6.1}$$

The columns of A are denoted $a_1, a_2, \ldots, a_m$, and the entries of x and b are $\xi_1, \xi_2, \ldots, \xi_n$ and $\beta_1, \beta_2, \ldots, \beta_m$, respectively. We use a_0 as a synonym for c.

We first add bounds to all variables. It seems reasonable to suppose that bounds $\ell \leq x \leq u$ can be provided so that: (i) if (6.1) has a feasible solution, it has a feasible solution satisfying $\ell \leq x \leq u$; and (ii) if (6.1) has an optimal solution, it has one satisfying $\ell \leq x \leq u$. We now choose a tolerance $\nu > 0$ and add to (6.1) the "artificial" bounds

$$\bar{\ell} = \ell-\nu(u-\ell) \leq x \leq u+\nu(u-\ell) = \bar{u} \ . \tag{6.1a}$$

We denote by (6.1)' the problem (6.1) with the bounds (6.1a) adjoined. Note that, by our assumptions, if problem (6.1)' is infeasible, then (6.1) is infeasible; and if all optimal solutions to problem (6.1)' have $\xi_j < \ell_j$ (or $\xi_j > u_j$) for some j , then (6.1) is unbounded. Finally, if (6.1)' has an optimal solution with $\bar{\ell}_j < \xi_j < \bar{u}_j$ for all j , this optimal solution also solves (6.1). For notational purposes we write (6.1a) in the form $a_i^T x \leq \beta_i$, for $i = m + 1, m + 2, \ldots$.

It is easy to construct an ellipsoid E_0 that contains $\{x \in R^n | \bar{\ell} \leq x \leq \bar{u}\}$. E_0 is given as in (2.1) with $x_0 = (\bar{\ell}+\bar{u})/2$ and $B = B_0 = L_0 D_0 L_0^T$, where $L_0 = I$ and $D_0 = \mathrm{diag}(n(\bar{u}_1-\bar{\ell}_1)^2/4,\ldots,n(\bar{u}_n-\bar{\ell}_n)^2/4)$. Since the size of this initial ellipsoid affects the performance of the algorithm, it may be worthwhile to perform some preprocessing to improve the bounds ℓ and u . For example, if a_{ij} is positive, we may deduce that

$$\xi_j \leq (\beta_i - \sum_{a_{ij}>0,k\neq j} a_{ik}\ell_k - \sum_{a_{ij}<0} a_{ik}u_k)/a_{ij} \ .$$

A similar lower bound can be deduced if a_{ij} is negative.

The algorithm constructs a sequence $\{E_k\}$ of ellipsoids. E_k is given by formula (2.1) with x_k replacing x_0 and B_k replacing B . We maintain B_k in the factored form $B_k = L_k D_k L_k^T$, with L_k unit lower triangular and D_k diagonal. The updating formulae of sections 2 to 5 are used to proceed from iteration k to iteration $k+1$; for instance B_{k+1} will be given by B_+ in (2.6), etc.

Clearly there are many ways to combine the ingredients of sections 2 and 3. Below we describe one possibility.

Initialization Choose x_0, B_0, L_0 and D_0 as described above. Choose tolerances $\varepsilon_1 > \varepsilon_2 > 0$ for the objective function value. (The algorithm may terminate with a feasible solution to (6.1) that comes within ε_1 of the optimal value of (6.1). It may also terminate with a feasible solution to (6.1)' that comes within ε_2 of the optimal value of (6.1)'; in this case, (6.1) may be unbounded.) Define $I = \{1,2,\ldots,m\}$, $\bar{I} = \phi$ and $\bar{\bar{I}}$ = the index set of artificial bounds. ($I \cup \bar{I}$ is the index set of all original constraints; $\bar{I}$ consists of constraints of (6.1) deemed to be nonbinding at optimal solutions.) For each $i \in I \cup \bar{\bar{I}} \cup \{0\}$, compute $\eta_i = a_i^T c$ ($\eta_i = \pm c_j$ for some j, for all $i \in \bar{\bar{I}}$), and for each $i \in I \cup \{0\}$ compute $\gamma_{0i} = a_i^T B_0 a_i$ as in (4.3). Set $\underline{z} \leftarrow c^T x_0 - \gamma_{00}^{1/2}$, $\beta_0 \leftarrow +\infty$ and $k \leftarrow 0$. ($\underline{z}$ is a lower bound on the optimal value of z when the bounds (6.1a) are added; β_0 is an upper bound.) If x_0 is feasible, set $\bar{x} \leftarrow x_0$ and $\beta_0 \leftarrow c^T \bar{x} < \infty$.

Iteration k:

Compute $s_{ki} = a_i^T x_k - \beta_i$ and $\alpha_{ki} = s_{ki}/\gamma_{ki}^{1/2}$ for all $i \in I \cup \{0\}$.

If for any $i \in I$, $\alpha_{ki} > 1$, STOP ; (6.1) is infeasible.

If for any $i \in I$, $\alpha_{ki} < -1$, transfer i from I to $\bar{I}$ if $i \le m$ or $\bar{\bar{I}}$ if $i > m$.

If $|I| = n$ go to Vertex routine if desired.

Case 1: Some α_{ki} , $i \in I$ is positive.

Choose $p \in I$ such that $\alpha_{kp} = \max\{\alpha_{ki} | i \in I\}$.

If $\alpha_{k0} > 0$ and u as in (2.12) with $\bar{A} = [a_0, a_p]$ is defined and nonnegative, find a and β from the best surrogate cut as in Section 2.

Otherwise, if $\alpha_{k0} > \alpha_{kp}$ set $a \leftarrow a_0$ and $\beta \leftarrow \beta_0$; else set $a \leftarrow a_p$ and $\beta \leftarrow \beta_p$.

Go to Update.

<u>Case 2</u>: $\alpha_{ki} \le 0$ for all $i \in I$.

Compute $s_{ki} = a_i^T x_k - \beta_i$ for all $i \in \bar{I} \cup \bar{\bar{I}}$.

If any s_{ki} is positive, transfer i from $\bar{I}$ or $\bar{\bar{I}}$ to I , recompute γ_{ki} as in (4.3), set $\alpha_{ki} = s_{ki}/\gamma_{ki}^{1/2}$ and go to <u>Case 1</u>.

Otherwise, set $a \leftarrow a_0$ and calculate $\hat{a}$, γ, v and w as in (4.3) - (4.4).

Compute $\hat{\theta} = \min_{i \in I \cup \bar{I} : \eta_i < 0} \{s_{ki}/\eta_i, \infty\}$,

$$\hat{\phi} = \min_{i \in I \cup \bar{I} : a_i^T w < 0} \{s_{ki}/a_i^T w, \infty\} .$$

If $\hat{\theta}$ or $\hat{\phi}$ equals ∞ , STOP; (6.1) is unbounded.

Otherwise, compute

$$\theta = \min_{i \in \bar{\bar{I}} : \eta_i < 0} \{s_{ki}/\eta_i, \hat{\theta}\} ,$$

$$\phi = \min_{i \in \bar{\bar{I}} : a_i^T w < 0} \{s_{ki}/a_i^T w, \hat{\phi}\}.$$

Set $\alpha \leftarrow \max \{\alpha_{k0}, \theta\eta_0/\gamma^{1/2}, \phi\}$ and

$\beta_0 \leftarrow c^T x_k - \gamma^{1/2} \alpha$.

If $\alpha = \theta\eta_0/\gamma^{1/2}$, set $\bar{x} \leftarrow x_k - \theta c$

if $\alpha = \phi$, set $\bar{x} \leftarrow x_k - \phi w$.

<u>Update</u>:

Update x_k and γ_{ki} , $i \in I \cup \{0\}$, from (4.3) - (4.6).

Update D_k and L_k as in (5.3) - (5.5).

Set $\underline{z} \leftarrow \max \{\underline{z}, c^T x_{k+1} - \gamma_{k+1,0}^{1/2}\}$.

<u>Convergence</u>

If $\beta_0 > \underline{z} + \varepsilon_1$, proceed to iteration $k + 1$.

Else, compute $\hat{\ell}_j = \xi_j^k - (b_{jj}^k)^{1/2}$ and $\hat{u}_j = \xi_j^k + (b_{jj}^k)^{1/2}$, where b_{jj}^k is the j-th diagonal element of B_k , for $j = 1,\dots,n$. $\hat{\ell}_j$ and $\hat{u}_j$ are lower and upper bounds for the j-th component of a point in E_k .

If, for some j , $\hat{\ell}_j > u_j$ or $\hat{u}_j < \ell_j$ then STOP; (6.1) is unbounded.

If, for all j , $\bar{\ell}_j < \hat{\ell}_j$ and $\bar{u}_j > \hat{u}_j$, STOP;

$\bar{x}$ is within ε_1 of the optimal value of (6.1).

If $\beta_0 \le \underline{z} + \varepsilon_2$, STOP. The feasible solution $\bar{x}$ to (6.1)' is within ε_2 of the optimal value of (6.1)'. Either $\bar{x}$ is close to optimal for (6.1) or (6.1) is unbounded.

Else, proceed to iteration $k + 1$.

<u>Vertex</u>:

If the a_i , $i \in I$ are linearly dependent, return.

Otherwise, solve $a_i^T x_k = \beta_i$ for $i \in I$ to get $\hat{x}$.

If $a_i^T \hat{x} > \beta_i$ for any $i \in \bar{I}$ return;

$\hat{x}$ is an infeasible vertex.

Else, solve $\sum_{i \in I} \lambda_i a_i = a_0$.

If any λ_i is positive, return; $\hat{x}$ is a nonoptimal vertex.

Otherwise, set $\bar{x} \leftarrow \hat{x}$ and STOP; $\bar{x}$ is an optimal solution to (6.1).

<u>Remarks</u>: 1) The assignment of a constraint to I, $\bar{I}$, or $\bar{\bar{I}}$ is tentative and dynamic. Since Case 1 predominates, a tentative choice of I saves work at most iterations at the possible cost of missing the most violated constraint. When a (tentatively) feasible point x_k is found, Case 2 applies and the constraints in $\bar{I} \cup \bar{\bar{I}}$ assumed to be nonbinding are tested for feasibility. Also, for simplicity we have ignored the possibility that the constraint set of problem (6.1) may include lower (or upper) bounds on some of the variables. If this is the case, artificial lower (or upper) bounds for these variables should not be included in problem (6.1)' and the tests in the <u>Convergence</u> section of the algorithm should be performed only on the artificial bounds.

2) It follows from Theorem 2.1 that each E_k contains all feasible points for problem (6.1)' that satisfy $c^T x \le \beta_0$; (i.e., all points x for which $c^T x \le c^T \bar{x}$ if a feasible solution $\bar{x}$ has been constructed, or $c^T x \le \infty$, otherwise). Hence, termination with some $\alpha_{ki} > 1$ can only occur if a feasible $\bar{x}$ has not been found and the constraints of (6.1)' are inconsistent. If termination occurs in routine Vertex, then the vertex $\bar{x}$ computed is clearly optimal. If $\hat{\theta}$ or $\hat{\phi}$ constructed in Case 2 is $+\infty$, (6.1) is unbounded since $x_k - \theta c$, $\theta \to \infty$, or $x_k - \phi w$, $\phi \to \infty$, remains feasible and has objective function value tending to $-\infty$. Now consider termination in Convergence. If $\hat{\ell}_j \ge u_j$ (or $\hat{u}_j \le \ell_j$) then all optimal solutions to (6.1)' have their jth component between u_j and $\bar{u}_j$ (or between $\bar{\ell}_j$ and ℓ_j). By our assumption, this implies that (6.1) is unbounded. If $\bar{\ell}_j < \hat{\ell}_j$ and $\hat{u}_j < \bar{u}_j$ then no optimal solution of (6.1)' has any of these bounds active; thus optimal solutions are also optimal solutions of (6.1). The other kind of termination in Convergence ($\beta_0 \le z + \varepsilon_2$) is unlikely to occur. Usually in this case E_k is thin in the "c" direction, while for some j , E_k intersects $x_j = \bar{\ell}_j$ and $x_j = \ell_j$ or $x_j = u_j$ and $x_j = \bar{u}_j$; hence it is fat in some direction (one major axis has length at least $\nu(u_j - \ell_j)$) . Termination in this case corresponds generally to some constraint normal a_i being almost parallel to c . This situation is somewhat analogous to termination in the simplex method with some reduced costs $\bar{z}_j$ less than some small tolerance ε in absolute value. For numerical reasons the test for optimality in the simplex method is usually $\bar{z}_j \ge -\varepsilon$ for all j .

3) We note that it is necessary to compute a square root of γ_{ki} only for the cut actually made on each iteration since all tests performed on α_{ki} can be made on the basis of $\alpha_{ki}^2 = s_{ki}^2/\gamma_{ki}$ and the sign of s_{ki} .

Also for simplicity we have not indicated where in the algorithm a fresh computation of $\gamma_{ki} = a_i^T B_k a_i$ is required as this should be evident from our remarks in section 4.

4) If α is set equal to α_{k0} in Case 2, this means that the most recently used objective function cut is deeper than one obtained by moving in either of the directions $-c$ or $-B_k c$. Although the same cut then gets used again, it is applied to a new ellipsoid.

5) We have described an "all-primal" algorithm that modifies β_0 and $\underline{z}$ based only on feasible solutions. Gacs and Lovasz [6] suggest combining primal and dual constraints with an inequality enforcing strong duality ($c^T x \geq b^T y$). The latter approach increases the dimensionality to $m+n$ and forces the feasible region to have zero volume, so that perturbation is necessary for finite convergence--for details, see [6]. An alternative approach is by binary search. As soon as β_0 is reduced from $+\infty$ and a feasible solution $\bar{x}$ is constructed, we may set $\hat{\beta} \leftarrow (\beta_0+\underline{z})/2$ and attempt to find a feasible solution with $c^T x \leq \hat{\beta}$. If a feasible solution is found, β_0 is lowered and the process continued. If the new problem is infeasible, $\underline{z}$ is increased (at least to the trial $\hat{\beta}$) and again the process continues. Notice that without the infeasibility test of our algorithm, such an approach would be very inefficient, since if $\hat{\beta}$ was chosen too small, many iterations would be necessary to show the problem infeasible. Binary search should also incorporate sophisticated methods to update upper and lower bounds on the optimal value. For example, if a feasible solution $\bar{x}$ is found, then setting $\beta_0 \leftarrow c^T\bar{x} - \gamma^{1/2}\alpha$, where γ and α are determined as in Case 2 of our algorithm, gives an upper bound that is less than or equal to $c^T\bar{x}$, which is itself less than or equal to $\hat{\beta}$. Also, whether the problem

corresponding to $c^Tx \leq \hat{\beta}$ is feasible or not, the hyperplane $\{x|c^Tx = \underline{\beta}\}$, where $\underline{\beta} = c^Tx_k - \gamma_{k0}$, tangent to the current ellipsoid E_k at the kth iteration, furnishes a lower bound $\underline{\beta}$ for the optimal value of z as long as $\alpha_{k0} \leq 1$; if $\alpha_{k0} > 1$, the current problem is infeasible and so $\hat{\beta}$ is a lower bound.

We next give a theorem that bounds the number of iterations to attain a certain accuracy when the feasible region is sufficiently "fat". We are not concerned with a priori bounds on volumes derived from integer data A, b and c: rather we take the practical viewpoint of user-supplied bounds.

Suppose we apply the algorithm to solve

$$\text{minimize } c^Tx \text{ subject to } A^Tx \leq b,\ 0 \leq x \leq u, \tag{6.2}$$

initializing as described above in remark (1). Suppose also that it is known that the feasible region of (6.2) is "fat", so that there is some point $\hat{x}$ with $a_i^T\hat{x} \leq \beta_i - ||a_i||\lambda$, $i = 1,\ldots,m$ and $\lambda \leq \hat{x}_j \leq u_j - \lambda$ for some known positive λ.

<u>Theorem 6.1</u>. Suppose the algorithm is run for $k \geqq 2n(n+1)[\sum_{j=1}^{n} \log u_j/n + \frac{1}{2}\log n - 1 + \log(1/\lambda) + \log(1/\eta)]$ iterations, where $\eta < 1$. Then it will have constructed a feasible solution $\bar{x}$ with $c^T\bar{x} \leq z^*(1-\eta) + \eta\bar{z}$, where z^* is the optimal value of (6.2) and $\bar{z} = \sum\{c_ju_j|c_j > 0\}$.

<u>Proof</u>. If μ is the volume of the unit n-ball, then the volume of E_0 is $\mu\cdot(\prod_{j=1}^{n} u_j)(\sqrt{n}/2)^n$, from the form of D_0 given above. As shown in Gacs and Lovasz [6], the volume of the ellipsoid E_k is at most $e^{-k/2(n+1)} \leqq [(\prod_{j=1}^{n} u_j)\cdot(\sqrt{n}/2)^n\cdot\lambda^{-n}\cdot\eta^{-n}]^{-1}$ times that of E_0, i.e. at most $(\lambda\eta)^n\mu$. Now the ball S of radius λ around $\hat{x}$ is entirely within the feasible region

of (6.2). Hence if no case 2 iterations have occurred, S, with volume $\lambda^n\mu$, would be contained in E_k, a contradiction. Thus some feasible solution $\bar{x}$ has been generated, and E_k contains the set $T = \{x \in R^n | A^T x \le b,\ 0 \le x \le u,\ c^T x \le c^T \bar{x}\}$. Now consider $S(\alpha) = \{x^* + \alpha(x-x^*) | x \in S\}$ for $0 < \alpha \le 1$, where x^* is any optimal solution to (6.2). Clearly $S(\alpha)$ is feasible and its volume is $\alpha^n \lambda^n \mu$. Thus if $\alpha > \eta$, $S(\alpha) \not\subseteq T$, so that $c^T\bar{x} < z^*(1-\alpha) + \alpha \max\{c^T x | x \in S\} \le z^*(1-\alpha) + \alpha\bar{z}$. Hence $c^T\bar{x} \le z^*(1-\eta) + \eta\bar{z}$ as desired.

The theorem gives a worst-case bound; in particular, the volumes of the ellipsoids shrink faster than at a geometric rate with ratio $e^{-1/2(n+1)}$ if deeper cuts occur. The terms in the bound for k can be split into three groups. First, $2n(n+1)[\sum_{j=1}^{n} \log u_j/n + \frac{1}{2}\log n - 1]$ iterations reduce the volume to about that of the unit ball; this bound can be reduced by obtaining tight upper bounds u_j on the variables. Next, $2n(n+1)\log(1/\lambda)$ iterations reduce the volume further to about $\lambda^n\mu$ and guarantee the generation of a feasible solution. This bound is reduced if the feasible region is fatter, e.g., by relaxing tolerances on the constraints. Finally, a further $2n(n+1)\log(1/\eta)$ iterations reduce the relative error in the objective function to about η times its maximum value--thus in the worst case about $2n^2$ iterations reduce this error by about a factor of e. Of course, these estimates are very crude since the behavior of the iterations with respect to feasibility and objective function values is far from monotonic. However, they do give some indication of the possible number of iterations for a certain accuracy in the solution.

7. Problems with Equality Constraints

A linear programming problem involving equality constraints, say

$$Ax = b \tag{7.1}$$

where A has dimension $m \times n$ and rank m, can be reduced to one involving only $n-m$ variables. To do this in a numerically stable way, let

$$A^T = [Q_1 \vdots Q_2]\begin{bmatrix} R \\ 0 \end{bmatrix} = Q\begin{bmatrix} R \\ 0 \end{bmatrix} \tag{7.2}$$

where Q is an orthogonal matrix, i.e. $Q^TQ = I$, and Q_1 and Q_2 have m and $n-m$ columns, respectively. The columns of Q_2 and Q_1 are respectively, orthogonal bases for the linear subspace $\{x | Ax = 0\}$ and its orthogonal complement since $AQ_2 = 0$ and $Q_1^TQ_2 = 0$. If we define y by

$$x = Qy = Q_1y_1 + Q_2y_2$$

then it follows from (7.1) and (7.2) that

$$b = Ax = AQy = R^Ty_1$$

and the original problem has been reduced to one in $n-m$ variables y_2. Linear forms such as a^Tx become $d^Ty_2 + \nu$ in terms of y_2 where $d = Q_2^Ta$, $\nu = d^Tw$, and $w = Q_1y_1 = Q_1R^{-T}b$. Also, the bounds $0 \leq x \leq u$ become $-w \leq Q_2y_2 \leq u-w$, and it is a simple matter to show that

$$||y_2||^2 \leq ||u||^2 - ||y_1||^2$$

and

$$||y_2||^2 \le \sum_i \max\{(u_i - w_i)^2,\ w_i^2\}.$$

Finally, we point out that for problems of the form

$$\text{minimize} \quad c^T x$$

$$\text{subject to} \quad A^T x \ge b \quad \text{and} \quad x \ge 0$$

where A^T has dimension $m \times n$ with $m \ll n$, it is preferable to solve the dual if bounds on the dual variable can be obtained.

Acknowledgment. We are grateful to Philip Wolfe for calling reference [23] to our attention.

References

[1] S. Agmon, "The Relaxation Method for Linear Inequalities," Canadian Journal of Mathematics 6 (1954), 382-392.

[2] M.L. Balinski and P. Wolfe (eds.), Nondifferentiable Optimization, Mathematical Programming Study 3, North-Holland, Amsterdam, 1975.

[3] J.M. Bennett, "Triangular Factors of Modified Matrices," Numerische Mathematik 7 (1965), 217-221.

[4] Yu. M. Ermolev, "Methods of Solution of Nonlinear Extremal Problems," Kibernetika 2(4) (1966), 1-17, translated in Cybernetics 2(4) (1966), 1-14.

[5] R. Fletcher and M.J.D. Powell, "On the Modification of LDL^T Factorizations," Harwell Report TP. 519, 1973.

[6] P. Gacs and L. Lovasz, "Khachian's Algorithm for Linear Programming," Technical Report STAN-CS-79-750, Computer Science Department, Stanford University, California (1979).

[7] W.M. Gentleman, "Least Squares Computations by Givens Transformations Without Square Roots," Research Report CSRR-2062, University of Waterloo, Waterloo, Ontario, Canada, 1973.

[8] P.E. Gill, G.H. Golub, W. Murray and M.A. Saunders, "Methods for Modifying Matrix Factorizations," Mathematics of Computation 28 (1974), 505-535.

[9] P.E. Gill, W. Murray and M.A. Saunders, "Methods for Computing and Modifying the LDV Factors of a Matrix," Mathematics of Computation 29 (1975), 1051-1077.

[10] J.-L. Goffin, "Acceleration in the Relaxation Method for Linear Inequalities and Subgradient Optimization," Working Paper 79-10, Faculty of Management, McGill University, Montreal, Canada, to appear in the proceedings of a task force on nondifferentiable optimization held at IIASA, Laxenburg, Austria, December 1978.

[11] D. Goldfarb, "Extension of Davidon's Variable Metric Method to Maximization under Linear Inequality and Equality Constraints," SIAM J. Appl. Math. 17 (1969), 739-764.

[12] A.J. Hoffman, "On Approximate Solutions of Systems of Linear Inequalities," J. Res. Natl. Bur. Standards 49 (1952), 263-265.

[13] L.G. Khachian, "A Polynomial Algorithm in Linear Programming," Doklady Akademiia Nauk SSSR 244: 5 (1979), 1093-1096, translated in Soviet Mathematics Doklady 20: 1 (1979), 191-194.

[14] V. Klee and G.L. Minty, "How Good is the Simplex Algorithm?" in O. Shisha (ed.) Inequalities III, Academic Press, New York, 1972, 159-175.

[15] T. Motzkin and I.J. Schoenberg, "The Relaxation Method for Linear Inequalities," Canadian Jounal of Mathematics 6 (1954), 393-404.

[16] B.T. Polyak, "A General Method for Solving Extremum Problems," Doklady Akademiia Nauk SSSR 174 (1967), 33-36, translated in Soviet Mathematics Doklady 8 (1967), 593-597.

[17] B.T. Polyak, "Minimization of Unsmooth Functionals," Zurnal Vychisditel'noi Matematiki i Matematicheskoi Fiziki 9 (1969), 509-521, translated in USSR Computational Mathematics and Mathematical Physics 9 (1969), 14-29.

[18] J.B. Rosen, "The Gradient Projection Method for Nonlinear Programming, Part I. Linear Constraints," J. SIAM 8 (1960), 181-217.

[19] N.Z. Shor, "On the Structure of Algorithms for the Numerical Solution of Optimal Planning and Design Problems," Dissertation, Cybernetics Institute, Academy of Sciences of the Ukrainian SSR, Kiev (1964).

[20] N.Z. Shor, "The Rate of Convergence of the Generalized Gradient Descent Method," Kibernetika 4(3) (1968), 98-99, translated in Cybernetics 4(3) (1968), 79-80.

[21] N.Z. Shor, "Utilization of the Operation of Space Dilatation in the Minimization of Convex Functions." Kibernetika 6(1) (1970), 6-12, translated in Cybernetics 6(1) (1970), 7-15.

[22] N.Z. Shor, "Convergence Rate of the Gradient Descent Method with Dilatation of the Space," Kibernetika 6(2) (1970), 80-85, translated in Cybernetics 6(2) (1970), 102-108.

[23] N.Z. Shor, "Cut-off Method with Space Extension in Convex Programming Problems," Kibernetika 13(1) (1977), 94-95, translated in Cybernetics 13(1) (1977), 94-96.

[24] M.J. Todd, "Some Remarks on the Relaxation Method for Linear Inequalities," Technical Report 419, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York (1979).

[25] R.S. Varga, Matrix Iterative Analysis, Prentice Hall, Englewood Cliffs, N.J., 1962.

[26] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, London, 1965.

[27] D.B. Yudin and A.S. Nemirovskii, "Informational Complexity and Effective Methods of Solution for Convex Extremal Problems," Ekonomika i Matematicheskie Metody 12 (1976), 357-369.