COMBINATORICA 4 (4) (1984) 373—395

A NEW POLYNOMIAL-TIME ALGORITHM FOR LINEAR PROGRAMMING

N. KARMARKAR

Received 20 August 1984
Revised 9 November 1984

We present a new polynomial-time algorithm for linear programming. In the worst case, the algorithm requires $O(n^{3.5}L)$ arithmetic operations on $O(L)$ bit numbers, where n is the number of variables and L is the number of bits in the input. The running-time of this algorithm is better than the ellipsoid algorithm by a factor of $O(n^{2.5})$. We prove that given a polytope P and a strictly interior point $\mathbf{a} \in P$, there is a projective transformation of the space that maps P, $\mathbf{a}$ to P', $\mathbf{a}'$ having the following property. The ratio of the radius of the smallest sphere with center $\mathbf{a}'$, containing P' to the radius of the largest sphere with center $\mathbf{a}'$ contained in P' is $O(n)$. The algorithm consists of repeated application of such projective transformations each followed by optimization over an inscribed sphere to create a sequence of points which converges to the optimal solution in polynomial time.

1. Informal outline

In this Section we give an informal discussion of the main ideas involved in the algorithm. A more rigorous description will be given in the next Section.

1.1. Problem definition

We shall reduce the general linear programming problem over the rationals to the following particular case:

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

$$\text{subject to} \quad \begin{cases} A\mathbf{x} = \mathbf{0} \\ \sum x_i = 1 \\ \mathbf{x} \geqq \mathbf{0}. \end{cases}$$

Here $\mathbf{x}=(x_1, \ldots, x_n)^T \in \mathbf{R}^n$, $\mathbf{c} \in \mathbf{Z}^n$, $A \in \mathbf{Z}^{m \times n}$.

This is a substantially revised version of the paper presented at the Symposium on Theory of Computing, Washington D. C., April 1984.

AMS subject classification (1980): 90 C 05

Let Ω denote the subspace $\{\mathbf{x}|A\mathbf{x}=\mathbf{0}\}$. Let Δ denote the simplex $\Delta = \{x|x \geqq 0, \sum x_i = 1\}$ and let Π be the polytope defined by

$$\Pi = \Omega \cap \Delta.$$

Thus the problem we consider is

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

$$\text{subject to} \quad \mathbf{x} \in \Pi.$$

1.2. Optimization over a sphere

If we replace the constraint $\mathbf{x} \in \Delta$ by a constraint of the form $\mathbf{x} \in S$, where S is a sphere, then it is trivial to see that the optimum can be found by solving a linear system of equations.

The intersection of a sphere and an affine space is a lower dimensional sphere inside that affine space. Hence the problem becomes:

$$\min \mathbf{c}'^T\mathbf{x}, \quad \text{subject to} \quad \mathbf{x} \in S' \text{ (a sphere)}$$

where $\mathbf{c}'$ was obtained by projecting $\mathbf{c}$ orthogonally onto Ω.

But this problem is trivial: From the center of the sphere, take a step along the direction $-\mathbf{c}'$, of length equal to the radius of the sphere.

1.3. Bounds on the objective function

Let $\mathbf{a}_0$ be strictly interior point in the polytope P. Suppose we draw an ellipsoid E with center $\mathbf{a}_0$, that is contained in the polytope and solve the optimization problem over the restricted region E instead of P. How good a solution do we get as compared to the optimal solution? To derive a bound, we create another ellipsoid E' by magnifying E by a sufficiently large factor v so that E' contains P.

$$E \subset P \subset E', \quad E' = vE.$$

Let $f_E, f_P, f_{E'}$ denote the minimum values of objective function $f(\mathbf{x})$ on E, P, and E' respectively.

$$f(\mathbf{a}_0) - f_E \leqq f(\mathbf{a}_0) - f_P \leqq f(\mathbf{a}_0) - f_{E'} = v[f(\mathbf{a}_0) - f_E].$$

The last equation follows from the linearity of $f(\mathbf{x})$.

$$\frac{f(\mathbf{a}_0) - f_E}{f(\mathbf{a}_0) - f_P} \geqq \frac{1}{v}$$

$$\frac{f_E - f_P}{f(\mathbf{a}_0) - f_P} \leqq \left(1 - \frac{1}{v}\right).$$

Thus by going from $\mathbf{a}_0$ to the point, say $\mathbf{a}'$, that minimizes $f(\mathbf{x})$ over E we come closer to the minimum value of the objective function by a factor $1 - v^{-1}$. We can repeat the same process with $\mathbf{a}'$ as the center. The rate of convergence of this method depends on v, the smaller the value of v, the faster the convergence.

1.4. Projective transformations

Projective transformations of $\mathbf{R}^n$ are described by formulas of the form

$$\mathbf{x} \mapsto \frac{C\mathbf{x}+\mathbf{d}}{\mathbf{f}^T\mathbf{x}+g}$$

where $C \in \mathbf{R}^{n\times n}$, $\mathbf{d}, \mathbf{f} \in \mathbf{R}^n$, $g \in \mathbf{R}$ and the matrix

$$\begin{bmatrix} C & \mathbf{d} \\ \mathbf{f}^T & g \end{bmatrix}$$

is non-singular.

We shall be interested in projective transformations of the hyperplane

$$\Sigma = \{\mathbf{x} | \sum_i x_i = 1\}.$$

It is easy to see that these transformations are described by

$$\mathbf{x} \mapsto \frac{C\mathbf{x}}{\mathbf{e}^T C\mathbf{x}}$$

where $\mathbf{e}=(1, 1, \ldots, 1)^T$ and C is a non-singular $n\times n$ matrix.

For each interior point $\mathbf{a}$ of Δ there is a unique projective transformation $T(\mathbf{a}, \mathbf{a}_0)$ of the hyperplane Σ fixing all vertices of the simplex Δ and moving $\mathbf{a}=(a_1, \ldots, a_n)^T$ to the center $\mathbf{a}_0=(1/n)\mathbf{e}$. This transformation is given by

$$x_i' = \frac{x_i/a_i}{\sum_j x_j/a_j}.$$

Observe that the corresponding matrix C is now diagonal: $C=\operatorname{diag}\left(\frac{1}{a_1}, \ldots, \frac{1}{a_n}\right)$. The transformation $C(\mathbf{a}, \mathbf{a}_0)$ has the following properties:

1. T is one-to-one and maps the simplex onto itself. Its inverse is given by

$$x_i = \frac{a_i x_i'}{\sum_j a_j x_j'} \quad i = 1, 2, \ldots, n.$$

2. Each facet of the simplex given by $x_i=0$ is mapped onto the corresponding facet $x_i'=0$.

3. Image of the point $\mathbf{x}=\mathbf{a}$ is the center of the simplex given by $x_i'=\frac{1}{n}$.

4. Let A_i denote the ith column of A. Then the system of equations

$$\sum_i A_i x_i = \mathbf{0}$$

now becomes

$$\frac{\sum_i A_i a_i x_i'}{\sum_i a_i x_i'} = \mathbf{0}$$

or

$$\sum_i A_i' x_i' = \mathbf{0}$$

where

$$A_i' = a_i A_i.$$

We denote the affine subspace $\{\mathbf{x}' | A'\mathbf{x}' = \mathbf{0}\}$ by Ω'.

5. If $\mathbf{a} \in \Omega$ then the center of the simplex $\mathbf{a}_0 \in \Omega'$.

Let $B(\mathbf{a}_0, r)$ be the largest ball with center $\mathbf{a}_0$, the same as the center of the simplex, and radius r that is contained in the simplex. Similarly let $B(\mathbf{a}_0, R)$ be the smallest ball containing Δ. It is easy to show that $R/r = n-1$.

$$B(\mathbf{a}_0, r) \subseteq \Delta \subseteq B(\mathbf{a}_0, R)$$

$$B(\mathbf{a}_0, r) \cap \Omega' \subseteq \Delta \cap \Omega' \subseteq B(\mathbf{a}_0, R) \cap \Omega'.$$

But $\Delta \cap \Omega'$ is the image Π' of the polytope $\Pi = \Delta \cap \Omega$. The intersection of a ball and an affine subspace is a ball of lower dimension in that subspace and has the same radius if the center of the original ball lies in the affine subspace, which it does in our case. Hence we get

$$B'(\mathbf{a}_0, r) \subseteq \Pi' \subseteq B'(\mathbf{a}_0, R), \quad \frac{R}{r} = n-1$$

which proves that $v =$ dimension of the space can always be achieved by this method. (Note: Δ is an $n-1$ dimensional simplex.)

1.5. Invariant potential functions

The algorithm creates a sequence of points $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}, \ldots$ having decreasing values of the objective function. In the k^{th} step, the point $\mathbf{x}^{(k)}$ is brought into the center by a projective transformation. Then we optimize the objective function over the intersection of the inscribed ball and the affine subspace to find the next point $\mathbf{x}^{(k+1)}$. Based on the results of the previous Sections, we expect the objective function to be reduced by a factor of $(1-n^{-1})$ at least, in each step, but there is one more technicality we have to deal with. The set of linear functions is not invariant under a projective transformation, but ratios of linear functions are transformed into ratios of linear functions. With every linear objective function $l(\mathbf{x})$ we associate a "potential function" $g(\mathbf{x})$ expressed in terms of logarithms of ratios of linear functions, in order to measure the progress of the algorithm.

$$g(\mathbf{x}) = \sum_j \ln\left(\frac{l(\mathbf{x})}{x_j}\right) + k \quad (k \text{ constant}).$$

It has the following properties:

1. Any desired amount of reduction in the value of $l(\mathbf{x})$ can be achieved by sufficient reduction in the value of $g(\mathbf{x})$.
2. $g(\mathbf{x})$ is mapped into a function of the same form by projective transformations $T(\mathbf{a}, \mathbf{a}_0)$ described in 1.4.
3. Optimization of $g(\mathbf{x})$ in each step can be done approximately by optimizing a linear function (but a different linear function in different steps).

1.6. Complexity of the algorithm

The value of the objective function is reduced by a constant factor in $O(n)$ steps. As in the ellipsoid algorithm we define

$$L = \log(1+|D_{\max}|)+\log(1+\alpha)$$

where $D_{\max} = \max\{|\det(X)| \mid X \text{ is a square submatrix of constraint matrix } A\}$

$$\alpha = \max\{|c_i'|, |b_i| \quad i=1, \ldots, n\}.$$

Clearly, L is not greater than the length of the input.

During the course of the algorithm, we periodically check for optimality by converting the interior point solution to an extreme point solution without degrading the objective function value, and testing the extreme point for optimality. The interval between two successive checks can be adjusted so that time spent between checks is comparable to time spent in checking for optimality. Any two distinct values of the objective function on the vertices of the polytope have to differ by at least 2^{-kL} for some constant k [3]. Therefore, in at most $O(nL)$ steps we reach a situation in which any extreme point better than the current interior point has to be an exact optimal solution.

In each step, we have to solve a linear system of equations which takes $O(n^3)$ arithmetic operations in the worst case. However, if we solve the equations at each step by modifying the solution to the previous step, we can save a factor of $\sqrt{n}$. This method is described in Section 6.

For each arithmetic operation, we need a precision of $O(L)$ bits. Note that this much precision is required for any method that solves the equations $A\mathbf{x}=\mathbf{b}$ if $\det(A)=O(2^L)$, e.g., the complexity of gaussian elimination is $O(n^3 L \ln L \ln\ln L)$, although it is normally stated as $O(n^3)$. This factor L must appear in the complexity of any algorithm for linear programming, since representing the output can require $O(L)$ bits/variable in the worst case.

The overall complexity of our algorithm is $O(n^{3.5}L^2 \ln L \ln\ln L)$ as compared to $O(n^6 L^2 \ln L \ln\ln L)$ for the ellipsoid algorithm.

2. Main algorithm

2.1. Problem definition

In order to focus on the ideas which are new and the most important in our algorithm, we make several simplifying assumptions. In Section 5 we will remove all these restrictions by showing how a linear programming problem in standard form can be transformed into our canonical form.

Canonical form of the problem

Input: $A \in \mathbf{Z}^{m\times n}$, $\mathbf{c} \in \mathbf{Z}^n$.

Let $\Omega = \{\mathbf{x} \mid A\mathbf{x}=\mathbf{0}\}$, $\Delta = \{\mathbf{x} \mid \mathbf{x} \geqq \mathbf{0}, \ \sum x_i = 1\}$ and $a_0 = e/n$.

Assumptions: $\mathbf{c}^T\mathbf{x} \geqq 0$ for every $\mathbf{x} \in \Omega \cap \Delta$; $\mathbf{a}_0$ is a feasible starting point, i.e. $A\mathbf{a}_0 = 0$; $\mathbf{c}^T\mathbf{a}_0 \neq 0$.

Output: either $\mathbf{x}_0 \in \Omega \cap \Delta$ such that $\mathbf{c}^T\mathbf{x}_0 = 0$ or a proof that $\min\{\mathbf{c}^T\mathbf{x} \mid \mathbf{x} \in \Omega \cap \Delta\} > 0$.

2.2. Comments

1. Note that the feasible region is the intersection of an affine space with a simplex rather than the positive orthant. Initially it takes one projective transformation to bring a linear programming problem to this form.

2. The linear system of equations defining Ω is homogeneous i.e., the right hand side is zero. The projective transformation that transforms positive orthant into simplex also transforms inhomogeneous system of equations into an homogeneous one.

3. The target minimum value of the objective function is zero. This will be justified by combining primal and dual into a single problem.

2.3. Description of the Algorithm

The algorithm creates a sequence of points $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}$ by these steps:

Step 0. Initialize

$$\mathbf{x}^{(0)} = \text{center of the simplex}$$

Step 1. Compute the next point in the sequence

$$\mathbf{x}^{(k+1)} = \varphi(\mathbf{x}^{(k)}).$$

Step 2. Check for infeasibility

Step 3. Check for optimality

go to Step 1.

Now we describe the steps in more detail.

Step 1. The function $\mathbf{b}=\varphi(\mathbf{a})$ is defined by the following sequence of operations.

Let $D=\text{diag}\,\{a_1, a_2, \ldots, a_n\}$ be the diagonal matrix whose i^{th} diagonal entry is a_i.

1. Let $B=\begin{bmatrix} AD \\ \mathbf{e}^T \end{bmatrix}$

i.e., augment the matrix AD with a row of all 1's. This guarantees that $\text{Ker}\, B \subseteqq \Sigma$.

2. Compute the orthogonal projection of $D\mathbf{c}$ into the null space of B.

$$\mathbf{c}_p = [I - B^T(BB^T)^{-1}B]\, D\mathbf{c}.$$

3. $\hat{\mathbf{c}}=\dfrac{\mathbf{c}_p}{|\mathbf{c}_p|}$, i.e. $\hat{\mathbf{c}}$ is the unit vector in the direction of $\mathbf{c}_p$.

4. $\mathbf{b}'=\mathbf{a}_0-\alpha r\hat{\mathbf{c}}$ i.e. take a step of length αr in the direction $\hat{\mathbf{c}}$, where r is the radius of the largest inscribed sphere

$$r = \frac{1}{\sqrt{n(n-1)}}$$

and $\alpha \in (0, 1)$ is a parameter which can be set equal to 1/4.

5. Apply inverse projective transformation to $\mathbf{b}'$

$$\mathbf{b} = \frac{D\mathbf{b}'}{\mathbf{e}^T D\mathbf{b}'}.$$

Return $\mathbf{b}$.

Step 2. *Check for infeasibility.*

We define a "potential" function by

$$f(\mathbf{x}) = \sum_i \ln \frac{\mathbf{c}^T\mathbf{x}}{x_i}.$$

We expect certain improvement δ in the potential function at each step. The value of δ depends on the choice of parameter α in Step 1.4. For example if $\alpha = 1/4$ then $\delta = 1/8$. If we don't observe the expected improvement i.e. if $f(\mathbf{x}^{(k+1)}) > f(\mathbf{x}^{(k)}) - \delta$ then we stop and conclude that the minimum value of the objective function must be strictly positive. When the canonical form of the problem was obtained by transformation on the standard linear program, then this situation corresponds to the case that the original problem does not have a finite optimum i.e. it is either infeasible or unbounded.

Step 3. *Check for optimality.*

This check is carried out periodically. It involves going from the current interior point to an extreme point without increasing the value of the objective function and testing the extreme point for optimality. This is done only when the time spent since the last check exceeds the time required for checking.

3. Underlying conceptual algorithm

In this Section we show how the computations in step 2 represent an underlying conceptual sequence of operations. We will also introduce more notation and state theorems about the performance of the algorithm. All proofs will be given in Section 4. (A note on notation: In choosing variable names we use the same philosophy as in the design of a large program: Some variables are local to a theorem while others are global, some theorems are stated in terms of "formal" parameters and in order to apply them to the algorithm, one has to substitute actual values for the formal parameters e.g., a result about one step of the algorithm may be stated in terms of input point $\mathbf{a}$ and output point $\mathbf{b}$, and can be applied to the k^{th} step of the algorithm by substituting $\mathbf{a} = \mathbf{x}^{(k)}$ and $\mathbf{b} = \mathbf{x}^{(k+1)}$.)

3.1. The top level

Theorem 1. *In* $O(n(q + \log n))$ *steps the algorithm finds a feasible point* $\mathbf{x}$ *such that*

$$\frac{\mathbf{c}^T\mathbf{x}}{\mathbf{c}^T\mathbf{a}_0} \leqq 2^{-q}.$$

We associate the objective function $\mathbf{c}^T\mathbf{x}$ with a "potential function" $f(\mathbf{x})$

given by

$$f(\mathbf{x}) = \sum_j \ln\left(\frac{\mathbf{c}^T\mathbf{x}}{x_j}\right).$$

Theorem 2. *Either* (i) $f(\mathbf{x}^{(k+1)}) \leqq f(\mathbf{x}^{(k)}) - \delta$ *or*

(ii) *the minimum value of objective function is strictly positive, where* δ *is a constant and depends on the choice of the value of the parameter* α *of Step 1, Substep 4.*

A particular choice that works: If $\alpha = 1/4$, then $\delta \geqq 1/8$.

3.2. The basic step

One step of the algorithm is of the form $\mathbf{b} = \varphi(\mathbf{a})$ which consists of a sequence of three steps:

1. Perform the projective transformation $T(\mathbf{a}, \mathbf{a}_0)$ of the simplex Δ that maps input point $\mathbf{a}$ to the center $\mathbf{a}_0$.

2. Optimize (approximately) the transformed objective function over an inscribed sphere to find a point $\mathbf{b}'$.

3. Apply the inverse of T to point $\mathbf{b}'$ to obtain output $\mathbf{b}$.

We describe each of the steps in detail.

3.3. Projective transformation $T(\mathbf{a}, \mathbf{a}_0)$

Let $\mathbf{a} = (a_1, a_2, \ldots, a_n)$

Let $D = \text{diag}\,\{a_1, a_2, \ldots, a_n\}$ be the diagonal matrix with diagonal entries $a_1, a_2, \ldots, a_n$. Then $T(\mathbf{a}, \mathbf{a}_0)$ is given by

$$\mathbf{x}' = \frac{D^{-1}\mathbf{x}}{\mathbf{e}^T D^{-1}\mathbf{x}} \quad \text{where} \quad \mathbf{e} = (1, 1, \ldots, 1).$$

Its inverse is given by

$$\mathbf{x} = \frac{D\mathbf{x}'}{\mathbf{e}^T D\mathbf{x}}.$$

Let f' be the transformed potential function defined by

$$f(\mathbf{x}) = f'(T(\mathbf{x})).$$

Then

$$f'(\mathbf{y}) = \sum_j \ln \frac{\mathbf{c}'^T\mathbf{y}}{y_j} - \sum_j \ln a_j$$

where $\mathbf{c}' = D\mathbf{c}$. Let Ω' be the transformed affine space Ω.

$$A\mathbf{x} = 0 \Leftrightarrow AD\,\mathbf{x}' = 0.$$

Thus Ω' is the null-space of AD.

3.4. Optimization over a ball

Let r be the radius of the largest ball with center $\mathbf{a}_0$ that can be inscribed in the simplex Δ. Then

$$r = \frac{1}{\sqrt{n(n-1)}}.$$

We optimize over a smaller ball $B(\mathbf{a}_0, \alpha r)$ $0<\alpha<1$ for two reasons:

1. It allows optimization of $f'(\mathbf{y})$ to be approximated very closely by optimization of a linear function.

2. If we wish to perform arithmetic operations approximately rather than by exact rational arithmetic, it gives us a margin to absorb round-off errors without going outside the simplex.

One choice of the value of α that works is $\alpha=1/4$ and corresponds to $\delta>1/8$ in Theorem 2. We restrict the affine space $\Omega'=\{\mathbf{y}\mid AD\mathbf{y}=\mathbf{0}\}$ by the equation $\sum y_i= =1$. Let Ω'' be the resulting affine space. Let B be the matrix obtained by adding a row of all 1's to AD. Then any displacement in Ω'' is in the null space of B,

$$\text{i.e., for } u\in\Omega'' \quad \Omega''=u+\operatorname{Ker} B.$$

We are interested in optimizing $f'(\mathbf{y})$ over $B(\mathbf{a}_0, \alpha r)\cap\Omega''$. First, we prove the existence of a point that achieves a constant reduction in the potential function.

Theorem 3. *There exists a point* $\mathbf{b}'\in B(\mathbf{a}_0, \alpha r)\cap\Omega''$ *such that* $f'(\mathbf{b}')\leqq f'(\mathbf{a}_0)-\delta$, *where* δ *is a constant.*

Then we prove that minimization of $f'(\mathbf{x})$ can be approximated by minimization of the linear function $\mathbf{c}'^T\mathbf{x}$.

Theorem 4. *Let* $\mathbf{b}'$ *be the point that minimizes* $\mathbf{c}'^T\mathbf{x}$ *over* $B(\mathbf{a}_0, \alpha r)\cap\Omega''$. *Then* $f'(\mathbf{b}')\leqq f'(\mathbf{a}_0)-\delta$ *where* δ *is a positive constant depending on* α. *For* $\alpha=1/4$ *we may take* $\delta=1/8$.

Finally we describe an algorithm to minimize $\mathbf{c}'^T\mathbf{x}$, over $B(\mathbf{a}_0, \alpha^* r)$.

Algorithm A.

1. Project $\mathbf{c}'$ *orthogonally into the null space of* B:

$$\mathbf{c}_P = [I-B^T(BB^T)^{-1}B]\mathbf{c}'.$$

2. Normalize $\mathbf{c}_P$:

$$\hat{\mathbf{c}}_P = \frac{\mathbf{c}_P}{|\mathbf{c}_P|}.$$

3. Take a step of length αr *along the direction* $-\hat{\mathbf{c}}_P$

$$\mathbf{b}' = \mathbf{a}_0-\alpha r\hat{\mathbf{c}}_P.$$

Theorem 5. *The point* $\mathbf{b}'$ *returned by algorithm A minimizes* $\mathbf{c}'^T\mathbf{x}$ *over* $B(\mathbf{a}_0, \alpha r)\cap\Omega''$.

4. Analysis of the algorithm

Proof of Theorem 5. Let $\mathbf{z}$ be any point such that $\mathbf{z}\in B(\mathbf{a}_0, \alpha r)\cap\Omega''$

$$\mathbf{b}', \mathbf{z}\in\Omega'' \Rightarrow B(\mathbf{b}'-\mathbf{z}) = 0$$

$$\Rightarrow B^T(BB^T)^{-1}B(\mathbf{b}'-\mathbf{z}) = 0$$

$$\Rightarrow (\mathbf{c}'-\mathbf{c}_P)^T(\mathbf{b}'-\mathbf{z}) = 0$$

$$\Rightarrow \mathbf{c}'^T(\mathbf{b}'-\mathbf{z}) = \mathbf{c}_P^T(\mathbf{b}'-\mathbf{z})$$

$$\mathbf{c}_P^T(\mathbf{b}'-\mathbf{z}) = |\mathbf{c}_P^T|\hat{\mathbf{c}}_P^T[\mathbf{a}_0-\alpha r\hat{\mathbf{c}}_P-\mathbf{z}]$$

$$= |\mathbf{c}_P^T|\{\hat{\mathbf{c}}_P^T(\mathbf{a}_0-\mathbf{z})-\alpha r\}$$

$$\hat{\mathbf{c}}_P^T(\mathbf{a}_0-\mathbf{z}) \leqq |\hat{\mathbf{c}}_P||\mathbf{a}_0-\mathbf{z}| \leqq \alpha r \quad \text{since} \quad \mathbf{z}\in B(\mathbf{a}_0, \alpha r)$$

$$\mathbf{c}_P^T(\mathbf{b}'-\mathbf{z}) \leqq 0$$

$$\mathbf{c}'^T(\mathbf{b}'-\mathbf{z}) \leqq 0$$

$\mathbf{c}'^T\mathbf{b}' \leqq \mathbf{c}'^T\mathbf{z}$, for all $\mathbf{z}\in B(\mathbf{a}_0, \alpha r)\cap\Omega''$. ∎

Proof of Theorem 3. Let $\mathbf{x}^*$ be the point in $\Omega''\cap S$ where the minimum of $\mathbf{c}'^T\mathbf{x}$ is achieved. Clearly $\mathbf{x}^*\notin B(\mathbf{a}_0, \alpha r)$.

Let $\mathbf{b}'$ be the point where the line segment $\mathbf{a}_0\mathbf{x}^*$ intersects the boundary of the sphere $B(\mathbf{a}_0, \alpha r)$. $\mathbf{b}'=(1-\lambda)\mathbf{a}_0+\lambda\mathbf{x}^*$ for some $\lambda\in[0, 1]$. By definition, $\mathbf{b}'\in B(\mathbf{a}_0, \alpha r)$ and $\mathbf{a}_0, \mathbf{x}^*\in\Omega''$ imply $\mathbf{b}'\in\Omega''$.

Hence $\mathbf{b}'\in B(\mathbf{a}_0, \alpha r)\cap\Omega''$

$$\mathbf{c}'^T\mathbf{b}' = (1-\lambda)\mathbf{c}'^T\mathbf{a}_0+\lambda\mathbf{c}'^Tx^* = (1-\lambda)\mathbf{c}'^T\mathbf{a}_0$$

$$\frac{\mathbf{c}'^T\mathbf{a}_0}{\mathbf{c}'^T\mathbf{b}'} = \frac{1}{1-\lambda}$$

and

$$b_j' = (1-\lambda)a_{0j}+\lambda x_j^*.$$

Hence

$$f'(\mathbf{a}_0)-f'(\mathbf{b}') = \sum_j \ln\frac{\mathbf{c}'^T\mathbf{a}_0}{\mathbf{c}'^T\mathbf{b}'}\frac{b_j}{a_{0j}} = \sum_j \ln\frac{(1-\lambda)a_{0j}+\lambda x_j^*}{(1-\lambda)a_{0j}}$$

$$= \sum_j \ln\left[1+\frac{\lambda}{1-\lambda}\frac{x_j^*}{a_{0j}}\right].$$

Now we use the following inequality: If $P_i \geqq 0$ then $\prod_i (1+P_i) \geqq 1+\sum P_i$. Therefore, $\sum_i \ln(1+P_i) \geqq \ln(1+\sum P_i)$.

$$f'(\mathbf{a}_0) - f'(\mathbf{b}') \geqq \ln\left[1+\frac{\lambda}{1-\lambda}\frac{1}{a_{0j}}\sum_j x_j^*\right] = \ln\left[1+\frac{n\lambda}{1-\lambda}\right]$$

$$\mathbf{b}' = (1-\lambda)\mathbf{a}_0 + \lambda\mathbf{x}^*$$

$$\mathbf{b}' - \mathbf{a}_0 = \lambda(\mathbf{x}^* - \mathbf{a}_0)$$

$$\alpha r = |\mathbf{b}' - \mathbf{a}_0| = \lambda|\mathbf{x}^* - \mathbf{a}_0| \leqq \lambda R$$

where R is the radius of the smallest circumscribing sphere and $R/r = n-1$.

$$\lambda \geqq \alpha\frac{r}{R} = \frac{\alpha}{n-1}$$

$$1+\frac{\lambda n}{1-\lambda} \geqq 1+\frac{n\alpha/(n-1)}{1-\alpha/(n-1)} = 1+\frac{n\alpha}{n-1-\alpha} \geqq 1+\alpha$$

$$f'(\mathbf{a}_0) - f'(\mathbf{b}') \geqq \ln(1+\alpha).$$

Taking $\delta = \ln(1+\alpha)$, $f'(\mathbf{b}') \leqq f'(\mathbf{a}_0) - \delta$. ∎

Proof of Theorem 4. First we prove a few lemmas.

Lemma 4.1. *If* $|x| \leqq \beta < 1$ *then* $|\ln(1+x) - x| \leqq \dfrac{x^2}{2(1-\beta)}$.

Proof. A simple exercise in calculus. ∎

Lemma 4.2. *Let* $\beta = \alpha\sqrt{\dfrac{n}{n-1}}$. *Then for any* $\mathbf{x} \in B(\mathbf{a}_0, \alpha r)$, $\left|\sum_j \ln\dfrac{x_j}{a_{0j}}\right| \leqq \dfrac{\beta^2}{2(1-\beta)}$.

Proof. $\mathbf{x} \in B(\mathbf{a}_0, \alpha r)$ implies $|\mathbf{x} - \mathbf{a}_0|^2 \leqq \alpha^2 r^2$. Therefore

$$\sum_j \left(\frac{x_j - a_{0j}}{a_{0j}}\right)^2 \leqq \frac{\alpha^2 r^2}{(1/n)^2} = \frac{\alpha^2 n^2}{n(n-1)} = \beta^2$$

$$\left|\frac{x_j - a_{0j}}{a_{0j}}\right| \leqq \beta \quad \text{for} \quad j = 1, \ldots, n$$

$$\left|\ln\left(1+\frac{x_j - a_{0j}}{a_{0j}}\right) - \frac{x_j - a_{0j}}{a_{0j}}\right| \leqq \frac{1}{2(1-\beta)}\left(\frac{x_j - a_{0j}}{a_{0j}}\right)^2 \quad \text{by Lemma 4.1}$$

$$\sum_j \ln\frac{x_j}{a_{0j}} = \sum_j \ln\left[1+\frac{x_j - a_{0j}}{a_{0j}}\right]$$

$$\sum_j \left|\ln\frac{x_j}{a_{0j}} - \frac{x_j - a_{0j}}{a_{0j}}\right| \leqq \frac{1}{2(1-\beta)}\sum_j \left(\frac{x_j - a_{0j}}{a_{0j}}\right)^2 = \frac{\beta^2}{2(1-\beta)}$$

but

$$\sum_j \frac{x_j - a_{0j}}{a_{0j}} = \frac{1}{a_{0j}}\left[\sum_j x_j - \sum_j a_{0j}\right] = \frac{1}{a_{0j}}[1-1] = 0$$

$$\left|\sum_j \ln\frac{x_j}{a_{0j}}\right| \leqq \frac{\beta^2}{2(1-\beta)}. \quad ∎$$

Now we prove Theorem 4.

Define $\tilde{f}(\mathbf{x}) = n \ln \dfrac{\mathbf{c}'^T\mathbf{x}}{\mathbf{c}'^T\mathbf{a}_0}$. Let $\mathbf{b}_m$ be the point in $B(\mathbf{a}_0, \alpha r) \cap \Omega''$ where $f(\mathbf{x})$ achieves its minimum value.

$$\begin{aligned} f(\mathbf{a}_0) - f(\mathbf{b}') &= f(\mathbf{a}_0) - f(\mathbf{b}_m) + f(\mathbf{b}_m) - f(\mathbf{b}') \\ &= [f(\mathbf{a}_0) - f(\mathbf{b}_m)] \\ &\quad + [f(\mathbf{b}_m) - (f(\mathbf{a}_0) + \tilde{f}(\mathbf{b}_m))] \\ &\quad - [f(\mathbf{b}') - (f(\mathbf{a}_0) + \tilde{f}(\mathbf{b}'))] \\ &\quad + [\tilde{f}(\mathbf{b}_m) - \tilde{f}(\mathbf{b}')] \end{aligned}$$

By Theorem 3,

$$f(\mathbf{a}_0) - f(\mathbf{b}_m) \geqq \ln(1+\alpha). \tag{1}$$

For $\mathbf{x} \in B(\mathbf{a}_0, \alpha r) \cap \Omega'$,

$$\begin{aligned} f(\mathbf{x}) - (f(\mathbf{a}_0) + \tilde{f}(\mathbf{x})) &= \sum_j \ln \frac{\mathbf{c}'^T\mathbf{x}}{x_j} - \sum_j \ln \frac{\mathbf{c}'^T\mathbf{a}_0}{a_{0j}} - n \sum \ln \frac{\mathbf{c}'^T\mathbf{x}}{\mathbf{c}'^T a_0} \\ &= -\sum_j \ln \frac{x_j}{a_{0j}} \end{aligned}$$

$$|f(\mathbf{x}) - (f(\mathbf{a}_0) + \tilde{f}(\mathbf{x}))| = \left| \sum_j \ln \frac{x_j}{a_{0j}} \right| \leqq \frac{\beta^2}{2(1-\beta)} \tag{2}$$

by Lemma 4.2.

Since $\tilde{f}(\mathbf{x})$ depends on $\mathbf{c}'^T\mathbf{x}$ in a monotonically increasing manner, $\tilde{f}(\mathbf{x})$ and $\mathbf{c}'^T\mathbf{x}$ achieve their minimum values over $B(\mathbf{a}_0, \alpha r) \cap \Omega''$ at the same point, viz $\mathbf{b}'$,

$$\tilde{f}(\mathbf{b}_m) \geqq \tilde{f}(\mathbf{b}'). \tag{3}$$

By (1), (2) and (3),

$$f(\mathbf{a}_0) - f(\mathbf{b}') \geqq \ln(1+\alpha) - \frac{\beta^2}{(1-\beta)} \geqq \alpha - \frac{\alpha^2}{2} - \frac{\alpha^2 n^2}{n(n-1)\left[1-\alpha\sqrt{\dfrac{n}{n-1}}\right]}.$$

Define

$$\delta = \alpha - \frac{\alpha^2}{2} - \frac{\alpha^2 n}{(n-1)\left[1-\alpha\sqrt{\dfrac{n}{n-1}}\right]}$$

$$f(\mathbf{a}_0) - f(\mathbf{b}') \geqq \delta.$$

Observe that

1. $\lim\limits_{n\to\infty} \delta(n) = \alpha - \dfrac{\alpha^2}{2} - \dfrac{\alpha^2}{1-\alpha}$. If $\alpha = \dfrac{1}{4}$, then $\lim\limits_{n\to\infty} \delta(n) \geqq \dfrac{1}{8}$.

2. If $n \geqq 4$, $\alpha = \dfrac{1}{5}$ then $\delta \geqq \dfrac{1}{10}$. ∎

Proof of Theorem 2. By Theorem 4, $f'(\mathbf{b}') \leqq f'(\mathbf{a}_0) - \delta$. Applying the inverse transform $T^{-1}(\mathbf{a}, \mathbf{a}_0)$ and substituting $\mathbf{x}^{(k)} = T^{-1}(\mathbf{a}_0)$, we obtain $\mathbf{x}^{(k+1)} = T^{-1}(\mathbf{b}')$, $f(\mathbf{x}^{(k+1)}) \leqq f(\mathbf{x}^{(k)}) - \delta$. ∎

Proof of Theorem 1. For each k we have

$$f(\mathbf{x}^{(k+1)}) \leqq f(\mathbf{x}^{(k)}) - \delta$$

$$f(\mathbf{x}^{(k)}) \leqq f(\mathbf{x}^{(0)}) - k\delta$$

$$\sum_j \ln \frac{\mathbf{c}^T \mathbf{x}^{(k)}}{x_j^{(k)}} \leqq \sum_j \ln \frac{\mathbf{c}^T \mathbf{a}_0}{a_{0j}} - k\delta$$

$$n \ln \frac{\mathbf{c}^T \mathbf{x}^{(k)}}{\mathbf{c}^T \mathbf{a}_0} \leqq \sum_j \ln (\mathbf{x}_j^{(k)}) - \sum_j \ln a_{0j} - k\delta.$$

But $x_j^{(k)} \leqq 1$, $a_{0j} = \frac{1}{n}$, so $\ln \frac{\mathbf{c}^T \mathbf{x}^{(k)}}{\mathbf{c}^T \mathbf{a}_0} \leqq \ln (n) - \frac{k}{n}\delta$. Therefore there exists a constant k' such that after $m = k'(n(q + \ln (n)))$ steps.

$$\frac{\mathbf{c}^T \mathbf{x}}{\mathbf{c}^T \mathbf{a}_0} \leqq 2^{-q}. \quad \blacksquare$$

5. Transformation of a linear program to canonical form

Consider a linear programming problem in the standard form:

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

$$\text{subject to} \begin{cases} A\mathbf{x} \geqq \mathbf{b} \\ \mathbf{x} \geqq \mathbf{0}. \end{cases}$$

There are many ways by which this problem can be converted to our canonical form. We describe one such method.

Step 1. *Combine primal and dual problems.*

$$A\mathbf{x} \geqq \mathbf{b}$$

$$A^T \mathbf{u} \leqq \mathbf{c}$$

$$\mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{u} = 0$$

$$\mathbf{x} \geqq \mathbf{0} \quad \mathbf{u} \geqq \mathbf{0}.$$

By duality theory, this combined problem is feasible if and only the original problem has finite optimum solution.

Step 2. *Introduce slack variables.*

$$A\mathbf{x} - \mathbf{y} = \mathbf{b}$$

$$A^T \mathbf{u} + \mathbf{v} = \mathbf{c}$$

$$\mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{u} = 0$$

$$\mathbf{x} \geqq \mathbf{0} \quad \mathbf{u} \geqq \mathbf{0} \quad \mathbf{y} \geqq \mathbf{0} \quad \mathbf{v} \geqq \mathbf{0}.$$

Step 3. *Introduce an artificial variable to create an interior starting point.* Let $\mathbf{x}_0$, $\mathbf{y}_0$, $\mathbf{u}_0$, $\mathbf{v}_0$ be strictly interior points in the positive orthant

$$\text{minimize} \quad \lambda$$
$$\text{subject to} \begin{cases} A\mathbf{x}-\mathbf{y}+(\mathbf{b}-A\mathbf{x}_0+\mathbf{y}_0)\lambda = \mathbf{b} \\ A^T\mathbf{u}+\mathbf{v}+(\mathbf{c}-A^T\mathbf{u}_0)\lambda = \mathbf{c} \\ \mathbf{c}^T\mathbf{x}-\mathbf{b}^T\mathbf{u}+(-\mathbf{c}^T\mathbf{x}_0+\mathbf{b}^T\mathbf{v}_0)\lambda = 0 \\ \mathbf{x} \geqq \mathbf{0} \quad \mathbf{y} \geqq \mathbf{0}, \ \mathbf{u} \geqq \mathbf{0}, \ \mathbf{v} \geqq \mathbf{0}, \ \lambda \geqq 0. \end{cases}$$

Note that $\mathbf{x}=\mathbf{x}_0$, $\mathbf{y}=\mathbf{y}_0$, $\mathbf{u}=\mathbf{u}_0$, $\mathbf{v}=\mathbf{v}_0$, $\lambda=1$ is a strictly interior feasible solution which we can take as a starting point. The minimum value of λ is zero if and only if the problem in step 2 is feasible.

Step 3. *Change of Notation.* For convenience of description we rewrite the problem in Step 3 as

$$\text{minimize} \quad \mathbf{c}^T\mathbf{x}, \quad \mathbf{c}, \mathbf{x} \in \mathbf{R}^n$$
$$\text{subject to} \quad A\mathbf{x} = \mathbf{b}, \ \mathbf{x} \geqq \mathbf{0}$$

$\mathbf{x}=\mathbf{a}$ is a known strictly interior starting point, and the target value of the objective function we are interested in is zero.

Step 4. *A projective transformation of the positive orthant into a simplex.* Consider the transformation $\mathbf{x}'=T(\mathbf{x})$ where $\mathbf{x}\in\mathbf{R}^n$, $\mathbf{x}'\in\mathbf{R}^{n+1}$ are defined by

$$x_i' = \frac{x_i/a_i}{\sum_j (x_j/a_j)+1} \quad i = 1, 2, \ldots, n$$

$$x_{n+1}' = 1 - \sum_{i=1}^{n} x_i'.$$

Let P_+ denote the positive orthant

$$P_+ = \{\mathbf{x}\in\mathbf{R}^n | \mathbf{x} \geqq 0\}$$

and Δ denote the simplex

$$\Delta = \left\{\mathbf{x}\in\mathbf{R}^{n+1} | \mathbf{x} \geqq 0, \quad \sum_{i=1}^{n+1} x_i = 1\right\}.$$

The transformation $T(\mathbf{x})$ has the following properties:

1. It maps P_+ onto Δ.
2. It is one-to-one and the inverse transformation is given by

$$x_i = \frac{a_i x_i'}{x_{n+1}'} \quad i = 1, 2, \ldots, n.$$

3. The image of point $\mathbf{a}$ is the center of the simplex.
4. Let A_i denote ith column of A. If

$$\sum_{i=1}^{n} A_i x_i = \mathbf{b}$$

then

$$\frac{\sum A_i a_i x_i'}{x_{n+1}'} = \mathbf{b}$$

or

$$\sum_{i=1}^{n} A_i a_i x_i' - \mathbf{b} x_{n+1}' = \mathbf{0}.$$

We define the columns of a matrix A' by these equations:

$$A_i' = a_i A_i, \quad i = 1, \ldots, n$$

$$A_{n+1}' = -\mathbf{b}.$$

Then

$$\sum_{i=1}^{n+1} A_i' x_i' = \mathbf{0}.$$

Let Ω' denote this affine subspace, i.e. $\Omega' = \{\mathbf{x}' \in \mathbf{R}^{n+1} | A'\mathbf{x}' = \mathbf{0}\}$. Thus we get a system of homogeneous equations in the transformed space.

5. Since we are interested in target value of zero, we define Z to be the affine subspace corresponding to the zero set of the objective function in the original space

$$Z = \{\mathbf{x} \in \mathbf{R}^n | \mathbf{c}^T \mathbf{x} = 0\}.$$

Substituting

$$x_i = \frac{a_i x_i'}{x_{n+1}'}$$

we get

$$\sum_{i=1}^{n+1} \frac{c_i a_i x_i'}{x_{n+1}'} = 0.$$

Define $\mathbf{c}' \in \mathbf{R}^{n+1}$ by

$$c_i' = a_i c_i \quad i = 1, 2, \ldots, n$$

$$c_{n+1}' = 0.$$

Then $\mathbf{c}^T\mathbf{x} = 0$ implies $\mathbf{c}'^T\mathbf{x}' = 0$.

Therefore the image of Z is given by $Z' = \{\mathbf{x}' \in \mathbf{R}^{n+1} | \mathbf{c}'^T\mathbf{x}' = 0\}$.
The transformed problem is

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}'^T\mathbf{x}' \\ & A'\mathbf{x}' = \mathbf{0} \\ \text{subject to} \quad & \mathbf{x}' \geqq \mathbf{0}, \ \sum_{i=1}^{n+1} x_i' = 1. \end{aligned}$$

The center of the simplex is a feasible starting point. Thus we get a problem in the canonical form.

Application of our main algorithm to this problem either gives a solution $\mathbf{x}'$ such that $\mathbf{c}'^T\mathbf{x}' = 0$ or we conclude that minimum value of $\mathbf{c}'^T\mathbf{x}'$ is strictly positive. In the first case, the inverse image of $\mathbf{x}'$ gives an optimal solution to the original problem. In the latter case we conclude that the original problem does not have a finite optimum, i.e. it is either infeasible or unbounded.

6. Incremental computation of inverse

6.1. A modification to the main algorithm

The computational effort in each step of the main algorithm is dominated by the computation of $\mathbf{c}_p$:

$$\mathbf{c}_p = [I - B^T(BB^T)^{-1}B]D\mathbf{c}.$$

We have to find the inverse of (BB^T) or solve a linear system of equations of the form $(BB^T)\mathbf{u}=\mathbf{v}$.

Substituting $B=\begin{bmatrix} AD \\ \mathbf{e}^T \end{bmatrix}$

$$BB^T = \begin{bmatrix} AD^2A^T & AD\mathbf{e} \\ (AD\mathbf{e})^T & \mathbf{e}^T\mathbf{e} \end{bmatrix} = \begin{bmatrix} AD^2A^T & \mathbf{0} \\ \mathbf{0} & n \end{bmatrix}. \tag{1}$$

The only quantity that changes from step to step is the diagonal matrix D, since $D_{ii}^{(k)} = x_i^{(k)}$. In order to take advantage of the computations done in previous steps, we exploit the following facts about matrix inserve:

Let D, D' be $n \times n$ diagonal matrices.

1. If D and D' are "close" in some suitable norm, the inverse of AD'^2A^T can be used in place of the inverse of AD^2A^T.

2. If D and D' differ in only one entry, the inverse of AD'^2A^T can be computed in $O(n^2)$ arithmetic operations, given the inverse of AD^2A^T.

(Note: Instead of finding the inverse, one could also devise an algorithm based on other techniques such as LU decomposition. In practice, it may be better to update the LU decomposition directly, but this will not change the theoretical worst-case complexity of the algorithm.)

We define a diagonal matrix $D'^{(k)}$, a "working approximation" to $D^{(k)}$ in step k, such that

$$\frac{1}{2} \leqq \left[\frac{D_{ii}'^{(k)}}{D_{ii}^{(k)}}\right]^2 \leqq 2 \quad \text{for} \quad i = 1, 2, \ldots, n. \tag{2}$$

We use $(A(D'^{(k)})^2A^T)^{-1}$ in place of $(A(D^{(k)})^2A^T)^{-1}$. Initially, $D'^{(0)} = D^{(0)} = \frac{1}{n} I$.

We update D' by the following strategy:

1. $D'^{(k+1)} = \sigma^{(k)} D'^{(k)}$, where $\sigma^{(k)}$ is a scaling factor whose significance will be explained in Section 6.3. For the purpose of this section take $\sigma^{(k)} = 1$.

2. For $i=1$ to n do

if

$$\left[\frac{D_{ii}'^{(k+1)}}{D_{ii}^{(k+1)}}\right]^2 \notin \left[\frac{1}{2}, 2\right],$$

then let $D_{ii}'^{(k+1)} = D_{ii}^{(k+1)}$; and update $(A(D'^{(k+1)})^2A^T)^{-1}$ by rank-one modification.

Thus, this strategy maintains the property in equation (2). Now we describe the rank-one modification procedure. Suppose a symmetric square matrix M is

modified by a rank-one matrix expressed as the outer product of two vectors $\mathbf{u}$ and $\mathbf{v}$. Then its inverse can be modified by the equation

$$(M+\mathbf{u}\mathbf{v}^T)^{-1} = M^{-1} - \frac{(M^{-1}\mathbf{u})(M^{-1}\mathbf{v})^T}{1+\mathbf{u}^T M^{-1}\mathbf{v}}. \tag{3}$$

Given M^{-1}, computation of $(M+\mathbf{u}\mathbf{v}^T)^{-1}$ can be done in $O(n^2)$ steps. In order to apply equation (3), note that if D' and D'' differ only in the ith entry then

$$AD'^2A^T = AD''^2A^T + [D'^2_{ii} - D''^2_{ii}]\mathbf{a}_i\mathbf{a}_i^T \tag{4}$$

where $\mathbf{a}_i$ is the ith column of A. This is clearly a rank-one modification. If D' and D'' differ in l entries we can perform l successive rank-one updates in $O(n^2 l)$ time to obtain the new inverse. All other operations required for computing $\mathbf{c}_p$, the orthogonal projection of $\mathbf{c}$, involve either multiplication of a vector by a matrix, or linear combination of two vectors, hence can be performed in $O(n^2)$ time.

6.2. Equivalence of the modified and original problems

In this Section we analyze the effect of the replacement of $D^{(k)}$ by $D'^{(k)}$. Consider the following generalization of the optimization problem over an inscribed sphere in the transformed space:

Problem P':

$$\left.\begin{array}{l} \text{Minimize } \mathbf{c}'^T\mathbf{x} \\ \text{subject to } B\mathbf{x} = \mathbf{0} \\ \quad\text{and } \mathbf{x}^T Q\mathbf{x} \leqq \alpha' r \end{array}\right\}. \tag{5}$$

(Taking $Q=I$ and $\alpha'=\alpha$ corresponds to the original problem.)

We are going to show that replacing $D^{(k)}$ by $D'^{(k)}$ corresponds to solving the problem P', for appropriate choice of Q. Let $h(\mathbf{x})=\mathbf{x}^T Q\mathbf{x}$. Its gradient is given by $\nabla h(\mathbf{x})=2Q\mathbf{x}$.

At the optimal solution $\mathbf{x}$ to the problem P',

$$\mathbf{c}' = B^T\boldsymbol{\lambda} + \mu \cdot 2Q\mathbf{x} \tag{6}$$

where the vector $\boldsymbol{\lambda}$ and the scalar μ are undetermined Lagrange multipliers

$$BQ^{-1}\mathbf{c}' = BQ^{-1}B^T\boldsymbol{\lambda} + 2\mu B\mathbf{x} = BQ^{-1}B^T\boldsymbol{\lambda}$$

$$\boldsymbol{\lambda} = (BQ^{-1}B^T)^{-1}BQ^{-1}\mathbf{c}'. \tag{7}$$

Substituting for $\boldsymbol{\lambda}$ in equation (6), $\mathbf{x}$, which we now denote by $\mathbf{c}_p$, is given (up to a scale factor) by

$$\mathbf{x} = \mathbf{c}_p = [I - Q^{-1}B^T(BQ^{-1}B^T)^{-1}B]Q^{-1}\mathbf{c}'. \tag{8}$$

This requires the inverse of $BQ^{-1}B^T$. Substituting $B=\begin{bmatrix} AD \\ \mathbf{e}^T \end{bmatrix}$ we get

$$BQ^{-1}B^T = \begin{bmatrix} ADQ^{-1}DA^T & ADQ^{-1}\mathbf{e} \\ (ADQ^{-1}\mathbf{e})^T & \mathbf{e}^T Q^{-1}\mathbf{e} \end{bmatrix}. \tag{9}$$

It is enough to know the inverse of the submatrix $ADQ^{-1}DA^T$, because then $(BQ^{-1}B^T)^{-1}$ can be computed in $O(n^2)$ additional steps by the formula

$$\text{(10)}\quad \begin{bmatrix} M & \mathbf{a} \\ \mathbf{a}^T & c \end{bmatrix}^{-1} = \frac{1}{c-\mathbf{a}^T M^{-1}\mathbf{a}} \begin{bmatrix} (c-\mathbf{a}^T M^{-1}\mathbf{a})M^{-1}+(M^{-1}\mathbf{a})(M^{-1}\mathbf{a})^T & -M^{-1}\mathbf{a} \\ -(M^{-1}\mathbf{a})^T & 1 \end{bmatrix}$$

where M, $\mathbf{a}$ and c are $m\times m$, $m\times 1$, 1×1 matrices, respectively. But we know $(AD'^2A^T)^{-1}$ where $D'=D\cdot E$, and E is a diagonal "error matrix" such that $E_{ii}^2\in \in[1/2, 2]$. Setting $Q=E^{-2}$ we get

$$AD'^2A^T = ADE^2DA^T = ADQ^{-1}DA^T.$$

In the modified algorithm we keep $(AD'^2A^T)^{-1}$ around, update it whenever any entry in D' changes and use it in equation (8) to compute $\mathbf{c}_p$ and thereby solve the modified problem P'.

Now we relate the solution of problem P' to the main optimization problem. Since $Q_{ii}=E_{ii}^{-2}\in[1/2, 2]$

$$\text{(11)}\quad \frac{1}{2}\mathbf{x}^T\mathbf{x} \leqq \mathbf{x}^T Q\mathbf{x} \leqq 2\mathbf{x}^T\mathbf{x}.$$

Hence

$$B\left(\mathbf{a}_0, \frac{\alpha'}{2}r\right) \leqq \{\mathbf{x}|\mathbf{x}^T Q\mathbf{x} \leqq \alpha' r\} \subseteq B(\mathbf{a}_0, 2\alpha' r).$$

Taking $\alpha'=\alpha/2$, and taking intersection with Ω''

$$\text{(12)}\quad B(\mathbf{a}_0, \alpha r/4)\cap\Omega'' \subseteq \{\mathbf{x}|\mathbf{x}^T Q\mathbf{x} \leqq \alpha' r, B\mathbf{x}=0\} \subseteq B(\mathbf{a}_0, \alpha r)\cap\Omega''.$$

Because of the first inclusion,

$$\text{(13)}\quad \begin{aligned} &\min\{f'(\mathbf{x})|\mathbf{x}^T Q\mathbf{x} \leqq \alpha' r, B\mathbf{x}=\mathbf{0}\} \\ &\leqq \min\{f'(\mathbf{x})|\mathbf{x}\in B(\mathbf{a}_0, \alpha r/4)\cap\Omega''\} \leqq f'(\mathbf{a}_0)-\ln(1+\alpha/4). \end{aligned}$$

The last inequality follows from Theorem 3. Thus for the modified algorithm, the claim of Theorem 3 is modified as follows

$$\text{(14)}\quad f'(\mathbf{b}') \leqq f'(\mathbf{a}_0)-\ln(1+\alpha/4).$$

Because of the second inclusion, Lemma 4.2 continues to be valid and we can approximate minimization of $f'(\mathbf{x})$ by minimization of a linear function. The claim (as in Theorem 2) about one step of the algorithm $f(\mathbf{x}^{(k+1)})\leqq f(\mathbf{x}^{(k)})-\delta$ also remains valid where δ is redefined as

$$\text{(15)}\quad \delta = \ln(1+\alpha/4)-\frac{\beta^2}{2(1-\beta)}.$$

This affects the number of steps by only a constant factor and the algorithm still works correctly.

6.3. Performance of the modified algorithm

In this Subsection we show that the total number of rank-one updating operations in m steps of the modified algorithm is $O(m\sqrt{n})$. Since each rank-one modification requires $O(n^2)$ arithmetic operations, the average work per step is $O(n^{2.5})$ as compared to $O(n^3)$ in the simpler form of the algorithm.

In each step $|\mathbf{b}'-\mathbf{a}_0| \leqq \alpha r$. Substituting $\mathbf{b}'=T(\mathbf{x}^{(k+1)})$, $\mathbf{a}_0=n^{-1}\mathbf{e}$ and $T(\mathbf{x})=\frac{D^{-1}\mathbf{x}}{\mathbf{e}^T D^{-1}\mathbf{x}}$, $r=\frac{1}{\sqrt{n(n-1)}}$

$$\sum_i \left[\frac{x_i^{(k+1)}/x_i^{(k)}}{\sum_j x_j^{(k+1)}/x_j^{(k)}} - \frac{1}{n}\right]^2 \leqq \frac{\alpha^2}{n(n-1)}.$$

Let

$$\sigma^{(k)} = \frac{1}{n}\sum_j x_j^{(k+1)}/x_j^{(k)}. \tag{16}$$

Then

$$\sum_i \left[\frac{x_i^{(k+1)}}{x_i^{(k)}\sigma^{(k)}} - 1\right]^2 \leqq \beta^2.$$

Let

$$\Delta_i^{(k)} = \frac{x_i^{(k+1)}}{x_i^{(k)}\sigma^{(k)}}. \tag{17}$$

Therefore $\sum_i [\Delta_i^{(k)}-1]^2 \leqq \beta^2$. Recall that $D^{(k)}=\mathrm{Diag}\{x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\}$, $D^{(k+1)}=\mathrm{Diag}\{x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}\}$. $D'^{(k)}$ is updated in two stages.

First we scale $D'^{(k)}$ by a factor $\sigma^{(k)}$ defined in equation (16)

$$D'^{(k+1)} = \sigma^{(k)} D'^{(k)}. \tag{18}$$

Then for each entry i such that

$$\left[\frac{D_{ii}'^{(k+1)}}{D_{ii}^{(k+1)}}\right]^2 \notin \left[\frac{1}{2}, 2\right]$$

we reset $D_{ii}'^{(k+1)}=D_{ii}^{(k+1)}$.

Define discrepancy between $D^{(k)}$ and $D'^{(k)}$ as

$$\delta_i^{(k)} = \ln\left(\frac{D_{ii}'^{(k)}}{D_{ii}^{(k)}}\right). \tag{19}$$

Just after an updating operation for index i,

$$\delta_i^{(k)} = 0.$$

Just before an updating operation for index i,

$$|\delta_i^{(k)}| \geqq \ln\sqrt{2}.$$

Between two successive updates, say at steps k_1 and k_2,

$$\delta_i^{(k_2)} - \delta_i^{(k_1)} = \sum_{k=k_1}^{k_2-1} [\delta_i^{(k+1)} - \delta_i^{(k)}].$$

Let $\Delta\delta_i^{(k)}$ be the change in the discrepancy at step k.

$$\Delta\delta_i^{(k)} = \delta_i^{(k+1)} - \delta_i^{(k)}. \tag{20}$$

Then

$$\ln \sqrt{2} \leqq |\delta_i^{(k_2)}| = |\delta_i^{(k_2)} - \delta_i^{(k_1)}| \leqq \sum_{k=k_1}^{k_2-1} |\Delta\delta_i^{(k)}|. \tag{21}$$

Now we analyze the change in discrepancy in step k, assuming that no updating operation was performed for index i.

$$\Delta\delta_i^{(k)} = \delta_i^{(k+1)} - \delta_i^{(k)} = \ln \frac{D_{ii}'^{(k+1)}}{D_{ii}^{(k+1)}} \frac{D_{ii}^{(k)}}{D_{ii}'^{(k)}} = \ln \frac{\sigma^{(k)} x_i^{(k)}}{x_i^{(k+1)}} = -\ln \Delta_i^{(k)}$$

$$|\Delta\delta_i^{(k)}| = |\ln \Delta_i^{(k)}|. \tag{22}$$

Let $n_i =$ the number of updating operations corresponding to index i in m steps of the algorithm and $N = \sum_{i=1}^{n} n_i$, be the total number of updating operations in m steps.

By equations (20) and (21),

$$\ln \sqrt{2}\, n_i \leqq \sum_{k=1}^{m} |\ln \Delta_i^{(k)}|$$

$$\ln \sqrt{2}\, N \leqq \sum_{i=1}^{n} \sum_{k=1}^{m} |\ln \Delta_i^{(k)}|. \tag{23}$$

We use equation (17) to bound the R.H.S. of equation (23)

$$\sum_i [\Delta_i^{(k)} - 1]^2 \leqq \beta^2 \Rightarrow |\Delta_i^{(k)} - 1| \leqq \beta$$

$$\Rightarrow |\ln \Delta_i^{(k)} - (\Delta_i^{(k)} - 1)| \leqq \frac{(\Delta_i^{(k)} - 1)^2}{2(1-\beta)} \quad \text{by Lemma 4.1}$$

$$\sum_i [\Delta_i^{(k)} - 1]^2 \leqq \beta^2 \Rightarrow \sum_i |\Delta_i^{(k)} - 1| \leqq \sqrt{n}\beta.$$

Hence

$$\sum_{i=1}^{n} |\ln \Delta_i^{(k)}| = \sum_{i=1}^{n} |\ln \Delta_i^{(k)} - (\Delta_i^{(k)} - 1) + (\Delta_i^{(k)} - 1)|$$

$$\leqq \sum_{i=1}^{n} |\ln \Delta_i^{(k)} - (\Delta_i^{(k)} - 1)| + \sum_{i=1}^{n} |\Delta_i^{(k)} - 1|$$

$$\leqq \sum_{i} \frac{[\Delta_i^{(k)} - 1]^2}{2(1-\beta)} + \sqrt{n}\beta$$

$$\leqq \frac{\beta^2}{2(1-\beta)} + \sqrt{n}\beta.$$

$$\sum_{i=1}^{n} \sum_{k=1}^{m} |\ln \Delta_i^{(k)}| \leqq m\left[\frac{\beta^2}{2(1-\beta)} + \sqrt{n}\beta\right] = O(m\sqrt{n}). \tag{24}$$

From equations (22) and (24)

$$N = O(m\sqrt{n}). \tag{25}$$

7. Concluding remarks

7.1. Global analysis of optimization algorithms

While theoretical methods for analyzing local convergence of non-linear programming and other geometric algorithms are well-developed the state-of-the-art of global convergence analysis is rather unsatisfactory. The algorithmic and analytical techniques introduced in this paper may turn out to be valuable in designing geometric algorithms with provably good global convergence properties. Our method can be thought of as a steepest descent method with respect to a particular metric space over a simplex defined in terms of "cross-ratio", a projective invariant. The global nature of our result was made possible because any (strictly interior) point in the feasible region can be mapped to any other such point by a transformation that preserves affine spaces as well as the metric. This metric can be easily generalized to arbitrary convex sets with "well-behaved" boundary and to intersections of such convex sets. This metric effectively transforms the feasible region so that the boundary of the region is at infinite distance from the interior points. Furthermore, this transformation is independent of the objective function being optimized. Contrast this with the penalty function methods which require an ad hoc mixture of objective function and penalty function. It is not clear a priori in what proportion the two functions should be mixed and the right proportion depends on both the objective function and the feasible region.

7.2. Comments on the factor "L" in running time

Since representing the output of a linear programming problem requires $O(L)$ bits per variable in the worst case, the factor L must appear in the worst-case complexity of any algorithm for linear programming. In practice L is much smaller than n. Therefore, whenever it is possible to replace a factor of n by L, one should always do so if one is interested in an algorithm which is efficient in practice.

7.3. Performance in practice

Each step of the algorithm requires optimization of a linear function over an ellipsoid or equivalently, solution of a system of linear equations of the type $(ADA^T)\mathbf{x}=\mathbf{b}$, where A is a fixed matrix and D is a diagonal matrix with positive entries which changes by small amount from step to step. We devise a method based on successive rank-one modifications and prove a worst-case bound of $O(n^{2.5})$ arithmetic operations per step.

In practice, the matrices encountered are sparse, hence more efficient methods for solving the above problem are possible. Another feature of the algorithm which can lead to computational saving is that it is not necessary to find the exact solution to the optimization problem stated above. Here it is useful to distinguish between two types of approximate solutions. Let $\mathbf{x}_0$ be the exact solution and $\mathbf{x}$ an approximate solution and let $\mathbf{c}^T\mathbf{x}$ be the objective function. A strong approximation (or approximation in the solution space) requires that $\mathbf{x}$ be close to $\mathbf{x}_0$ in some suitable norm. A weak approximation (or approximation in objective function space) requires that $\mathbf{c}^T\mathbf{x}$ be close to $\mathbf{c}^T\mathbf{x}_0$. A weak approximation is sufficient for our method, and is easier to achieve numerically. The optimization problem in each step is a good candidate for applying iterative methods to obtain approximate solutions. The worst-case bound is based on finding an exact solution.

The number of steps of the algorithm depends on the "R/r" ratio, i.e. the ratio of radius of the sphere circumscribing the polytope to the radius of the inscribed sphere. We prove an $O(n)$ upper bound on this ratio. However it is likely to be smaller than n in typical problems of practical interest and also in the worst-case if the circumscribing sphere is chosen to include only that portion of the polytope having values of objective function better than the current value, and after computing the direction of one step (as described in section 2) its length is chosen to be $\alpha r'$ rather than αr where r' is the largest possible step length without going outside the feasible region. The possibility of improvement in R/r ratio gives rise to a number of research problems for further investigation, such as probabilistic analysis of the problem, experimental measurements on problems arising in practical applications, analysis of special cases of practical interest such as "box" type of constraints and finally, the most difficult of all, obtaining a better worst-case bound.

7.4. Numerical stability and round-off errors

In the ellipsoid algorithm, the round-off errors in numerical computation accumulate from step to step. The amount of precision required in arithmetic operations grows with the number of steps. In our algorithm, errors do not accumulate. On the contrary, the algorithm is self-correcting in the sense that the round-off error made in one step gets compensated for by future steps. If we allow 1% error in computation of the vector $\mathbf{x}^{(k)}$ in each step, we can compensate for the errors by running the algorithm 2% more steps.

7.5. Multiple columns and parallel computation

In the simplex method, the current solution is modified by introducing a non-zero coefficient for one of the columns in the constraint matrix. Our method allows the current solution to be modified by introducing several columns at once. In the algorithm described in this paper, all columns are "active", but a variation in which a suitably chosen subset of columns is active at one time is possible. It seems that on a highly parallel architecture a method that uses many columns at once may be preferable to the simplex method.

7.6. Other areas for further research

Since many polynomial-time solvable combinatorial problems are special cases of linear programming, this work raises the possibility of applying the new techniques to such special cases. Problems solved by the column generation method require a special variant of the algorithm.

Acknowledgement. The author wishes to thank Arjen K. Lenstra, Mark Manasse, Joan Plumstead, Alon Itai, László Babai, Mike Garey and Ron Graham for pointing out errors and making suggestions for improvement.

References

[1] H. S. M. COXETER, *Introduction to Geometry*, Wiley (1961).
[2] G. B. DANTZIG, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ (1963).
[3] M. GRÖTSCHEL, L. LOVÁSZ and A. SCHRIJVER, The Ellipsoid Method and its Consequences in Combinatorial Optimization, *Combinatorica* **1** (1981), 169—197.
[4] L. G. KHACHIYAN, A polynomial Algorithm in Linear Programming, *Doklady Akademii Nauk SSSR* **244**:S (1979), 1093—1096, translated in *Soviet Mathematics Doklady* **20**:1 (1979), 191—194.
[5] V. KLEE and G. L. MINTY, How good is the simplex algorithm? in *Inequalities III*, (ed. O. Shisha) Academic Press, New York, 1972, 159-179.
[6] O. VEBLEN and J. W. YOUNG, *Projective Geometry, 1—2*, Blaisdell, New York, (1938).
[7] R. J. WALKER, *Algebraic Curves*, Princeton University Press (1950).

N. Karmarkar
AT&T Bell Laboratories
Murray Hill, NJ 07974
U.S.A.