A New Polynomial-Time Algorithm for Linear Programming-II

Article *in* Combinatorica · December 1984

DOI: 10.1007/BF02579150 · Source: DBLP

1 author:

Narendra Karmarkar

39 PUBLICATIONS **5,620** CITATIONS

SEE PROFILE

A New Polynomial-Time Algorithm for Linear Programming

N. Karmarkar

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

We present a new polynomial-time algorithm for linear programming. The running-time of this algorithm is $O(n^{3.5}L^2)$, as compared to $O(n^6L^2)$ for the ellipsoid algorithm. We prove that given a polytope P and a strictly interior point $a \in P$, there is a projective transformation of the space that maps P, a to P', a' having the following property. The ratio of the radius of the smallest sphere with center a', containing P' to the radius of the largest sphere with center 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.

0. Some Comments on the Significance of the Result

0.1 Worst-case Bounds on Linear Programming

The simplex algorithm for linear programming has been shown to require an exponential number of steps in the worst-case [1]. A polynomial-time algorithm for linear programming was published by Khachiyan in 1979 [2]. The complexity of this algorithm is $O(n^6L^2)$ where n is the dimension of the problem and L is the number of bits in the input [3]. In this paper we present a new polynomial-time algorithm for linear programming whose time-complexity is $O(n^{3.5}L^2)$.

0.2 Polytopes and Projective Geometry

We prove a theorem about polytopes which seems to be interesting in its own right. Given a polytope $P \subseteq \mathbf{R}^n$ and a strictly interior point $a \in P$, there is a projective transformation of the space that maps P, a to P', a' having the following property: The ratio of the radius of the smallest sphere with center a', containing P' to the radius of the largest sphere with center a', contained in P' in $O(n)$.

Our algorithm for linear programming is based on repeated application of such projective transformations followed by optimization over the inscribed sphere to create a sequence of points which converges to the optimal solution in polynomial time.

0.3 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 should the two functions be mixed and the right proportion depends on both the objective function and the feasible region.

0.4 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.

The factor L^2 in the complexity of our algorithm comes from two sources. The number of steps of the algorithm is $O(nL)$, each step requires $O(n^{2.5})$ arithmetic operations and each arithmetic operation requires a precision of $O(L)$ bits.

If we are interested in finding a solution whose objective function value is certain fixed fraction, say 99.99% of the optimum value, then the algorithm requires only $O(n)$ steps thus saving a factor of L. The complexity of finding an exact solution can be better expressed in terms of a parameter R which we call the "discreteness factor" of the polytope, defined as:

$$R = \frac{s_{\max}}{s_{\min}} \quad \text{where}$$

$s_{\max}$ = Difference between the best and worst values of the objective function achieved on the vertices of the polytope.

$s_{\min}$ = Difference between the best and next best value of the objective function on the vertices of the polytope.

The actual number of steps required by our algorithm is $O(n \ln(R))$ which can be bounded above by $O(nL)$ because

$$\ln(R) = O(L) .$$

If the sequence of distinct values taken by the objective function o the vertices of the polytope is uniformly spaced, then the performance of the simplex algorithm depends linearly on R. It is possible to create examples in which R is exponentially large. Indeed, the examples which cause the simplex algorithm to run exponentially long have this property. Since performance of our algorithm depends logarithmically on R, we still get a polynomial-time algorithm.

Regarding the second factor of L, any algorithm — such as the simplex algorithm — that requires representation of inverse of a submatrix of the constraint matrix during the course of computation requires at least as much precision in arithmetic operations as our algorithm, which is $O(L)$ bits in the worst case. As compared to this worst-case bound, the simplex algorithm seems to work with much less precision in practice and the same amount of precision is sufficient for our algorithm, thus saving the second factor of L.

0.5 Performance in Practice

Each step of the algorithm requires optimization of a linear function over an ellipsoid or equivalently, solution of a linear system of equations of the type $(ADA^T)x = b$, 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 x_0 be the exact solution and x an approximate solution and let $c^T x$ be the objective function. A strong approximation, (or approximation in the solution space) requires that x be close to x_0 in some suitable norm. A weak approximation (or approximation in objective function space) requires that $c^T x$ be close to $c^T x_0$. 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 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 "box" type of constraints and finally, the most difficult of all, obtaining better worst-case bound.

0.6 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 $x^{(k)}$ in each step, we can compensate for the errors by running the algorithm 2% more steps.

0.7 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.

0.8 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.

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 Optimization Over a Quadratic Region

Consider the linear programming problem of the form

$$\begin{aligned} &\text{minimize} && c^T x\ , \quad c, x \in R^n \\ &\text{subject to} && Ax = b\ , \quad x \geqslant 0\ . \end{aligned}$$

Let Ω denote the affine space $\{x \mid Ax = b\}$. Let P_+ denote the positive orthant $\{x \mid x \geqslant 0\}$.

If we replace the constraint $x \in P_+$ by a constraint of the form $x \in E$, where E is an ellipsoid, then the problem becomes easy and can be solved by solving a linear system of equations as follows:

Apply a linear transformation T that transforms the ellipsoid E into a sphere S.

$$E \xrightarrow{T} S$$

In doing so, we have also transformed the affine space Ω into another affine space Ω' and the objective function vector to c'.

$$\Omega \xrightarrow{T} \Omega'\ , \quad c \xrightarrow{T} c'\ .$$

The transformed problem is:

$$\begin{aligned} &\text{minimize} && c'^T x \\ &\text{subject to} && x \in S \cap \Omega'\ . \end{aligned}$$

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

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

where c'' was obtained by projecting c' orthogonally onto Ω'.

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

1.2 Bounds on Objective Function

Let P be the polytope defined by $Ax = b, x \geqslant 0$ and let $a_0 \in P$, be a strictly interior point. Suppose we draw an ellipsoid E with center 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 ν so that E' contains P.

$$E \subset P \subset E'\ , E' = \nu E\ .$$

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

$$f(a_0) - f_E \leqslant f(a_0) - f_P \leqslant f(a_0) - f_{E'} = \nu\,[f(a_0) - f_E]$$

The last equation follows from the linearity of $f(x)$.

$$\frac{f(a_0) - f_E}{f(a_0) - f_P} \geqslant \frac{1}{\nu}$$

$$\frac{f_E - f_P}{f(a_0) - f_P} \leqslant \left(1 - \frac{1}{\nu}\right)\ .$$

Thus by going from a_0 to the point, say a', that minimizes $f(x)$ over E we come closer to the minimum value of the objective function by a factor $1 - \frac{1}{\nu}$. We can repeat the same process with a' as the center. The rate of convergence of this method depends on ν, the smaller the value of ν, the faster the convergence.

1.3 Projective Transformation

We are going to show that $\nu = n$ can always be achieved by making a suitable projective transformation.

Let $a = (a_1, a_2, \ldots, a_n)$ be any strictly interior point in the polytope P, $a \in P_+$, $a_i > 0$, $a \in \Omega$.

Define an n-dimensional simplex $S \subseteq R^{n+1}$ by the set of equations:

$$x_i \geqslant 0 \quad i = 1, \ldots, n+1$$

$$\sum x_i = 1$$

The center of this simplex is given by $a_0 = \frac{1}{n+1}\,\mathbf{e}$ where $\mathbf{e}$ is the vector of all 1's.

Consider a transformation T that maps P_+ into the simplex S:

$$x_i' = \frac{x_i/a_i}{\sum_i (x_i/a_i) + 1} \quad i = 1, \ldots, n$$

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

This transformation has the following properties:

1. Since $a_i > 0$, $x_i \geqslant 0$, we have, $\sum_i \frac{x_i}{a_i} + 1 > 0$.

   Also $x_i \geqslant 0 \Rightarrow x_i' \geqslant 0$.

   $$\sum x_i' = \frac{\sum_i x_i/a_i}{\sum_i x_i/a_i + 1} \leqslant 1\ , \text{ hence } x_{n+1}' \geqslant 0\ .$$

Thus P_+ is mapped into S. T is also one-to-one and onto, and its inverse is given by:

$$x_i = \frac{a_i x_i'}{1 - \sum x_i'} \quad i=1,2,\ldots, n$$

furthermore, the image of point $a \in P_+$ is the center $a_0 \in S$.

2. Since T is a projective transformation, it maps the affine space Ω into another affine space Ω'. Since $a \in \Omega$, its image $a_0 \in \Omega'$.

3. Each face of P_+ given by $x_i = 0$ is mapped into $x_i' = 0$, a face of the simplex. This accounts for n out of $n+1$ faces of the n-dimensional simplex.

Now consider a straight-line segment in P_+ defined by

$$x_i(t) = \mu_i t, \quad \mu_i \geq 0, \quad t \in (0, \infty) .$$

Its image in the simplex S is given by

$$x_i'(t) = \frac{\frac{\mu_i}{a_i} t}{(\sum_i \frac{\mu_i}{a_i}) t + 1} \quad i=1,\ldots, n$$

$$x_{n+1}'(t) = 1 - \sum x_i'(t) = \frac{1}{(\sum_i \frac{\mu_i}{a_i}) t + 1}$$

(Convince yourself that this set of equations defines a straight line in S, despite the denominator, as it must, since the image of a straight-line under a projective transformation is a straight-line.)

If we take the limit as $t \to \infty$, we get

$$x_i' = \frac{\mu_i/a_i}{\sum \mu_i/a_i}, \quad i=1,\ldots, n$$

$$\sum_{i=1}^{n} x_i' = 1 \quad \text{hence } x_{n+1}' = 0 .$$

Thus the "points at ∞" are mapped onto the $(n+1)^{st}$ face of the simplex.

Let $B(a_0, r)$ be the largest sphere with center a_0 that can be inscribed into the simplex S and let $B(a_0, R)$ be the smallest circumscribing sphere. It is easy to show that $\frac{R}{r} = n$.

$$B(a_0, r) \subseteq S \subseteq B(a_0, R)$$

hence

$$B(a_0, r) \cap \Omega' \subseteq S \cap \Omega' \subseteq B(a_0, R) \cap \Omega' .$$

But $S \cap \Omega'$ is the image P' of the polytope $P = P_+ \cap \Omega$. The intersection of a sphere and an affine space is a sphere of lower dimension in that space and has the same radius if the center of the original sphere lies in the affine space, which it does in our case. Hence we get

$$B'(a_0, r) \subseteq P' \subseteq B'(a_0, R), \quad \frac{R}{r} = n$$

which proves that $\nu = n$ is achieved by this method.

1.4 Invariant Potential Function

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

$$g(x) = \sum_j \ell n \frac{\ell(x)}{x_j} + k \quad (k \text{ constant}) .$$

It has the following properties:

1. Any desired amount of reduction in the value of $\ell(x)$ can be achieved by sufficient reduction in the value of $g(x)$.
2. $g(x)$ is invariant under projective transformation, (i.e., is mapped into a function of the same form).
3. Optimization of $g(x)$ in each step can be done approximately by optimizing a linear function, (but a different linear function in different steps).

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

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

If we run the algorithm for $O(nL)$ steps, then we come within $2^{-O(L)}$ of the optimum, at which point we round the solution by the same method used in the ellipsoid algorithm to get an exact optimum 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 5.

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 $Ax = b$ if $\det(A) = O(2^L)$, e.g., the complexity of gaussian elimination is $O(n^3L)$, 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 $O(n^{3.5}L^2)$ as compared to $O(n^6L^2)$ 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 6 we will remove all these restrictions by applications of standard techniques like slack variables, artificial variables, binary search, elementary linear transformations, etc.

Formulation of the problem:

$$\begin{array}{ll} \text{minimize} & c^T x, \quad c, x \in R^n \\ \text{subject to} & x \in \Omega \cap S \quad \text{where} \end{array}$$

$$\Omega = \{x | Ax = 0\}$$

$$\text{and} \quad S = \{x | x \geq 0, \quad \sum x_i = 1\} .$$

2.2 Assumptions:

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. But there is an additional equation $\sum x_i = 1$, and with the help of this equation any non-homogeneous system can be made homogeneous.
3. The minimum value of the objective function is zero. If the minimum value, say c_m, is known beforehand but is non-zero we can take a modified objective function $c^T x - c_m$ and make it homogeneous.

   If the minimum value is not known, a "sliding objective function" variant of the algorithm can be used and has the same time-complexity.
4. The problem is feasible, and the center of simplex S given by $a_0 = \frac{1}{n}\mathbf{e}$, where $\mathbf{e}$ is the vector of all 1's, is a feasible starting point.

   The so-called "feasibility problem" i.e., deciding whether a linear system of inequalities is feasible, and the "optimization problem" which is not known to be feasible can both be solved in terms of an optimization problem with a known starting feasible solution.
5. A termination parameter q is given, and our objective is to find a feasible x such

$$\frac{c^T x}{c^T a_0} \leqslant 2^{-q} .$$

As in the ellipsoid method, if we take $q = O(L)$, the resulting approximately optimal solution can be converted to an exact optimal solution.

2.3 Description of the Algorithm

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

$$x^{(0)} = a_0,\ k = 0$$

while $c^T x^{(k)}$ is still too large do
$\{\ x^{(k+1)} = \phi(x^{(k)})$

$$k = k+1\ \} .$$

One step of the algorithm is a computation of the form $b = \phi(a)$ where the function ϕ 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, i^{th} entry is a_i.

1. Let $B = \left[\frac{AD}{e^T}\right]$
   i.e., Augment the matrix AD with a row of all 1's.
2. $c_p = [I - B^T(BB^T)^{-1}B]\ Dc$
3. $\hat{c} = \frac{c_p}{|c_p|}$, i.e. $\hat{c}$ is the unit vector in the direction of c_p
4. $b' = a_0 - \alpha r\ \hat{c}$
   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. $b = \frac{D\ b'}{e^T D\ b'}$

   Return b.

3. Underlying Conceptual Algorithm

In this section we show how the algorithm in section 2 represents 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 a and output point b, and can be applied to the k^{th} step of the algorithm by substituting $a = x^{(k)}$ and $b = x^{(k+1)}$.)

3.1 The Top Level

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

$$\text{either } (i)\ \ c^T x = 0$$

$$\text{or } (ii)\ \ \frac{c^T x}{c^T a_0} \leqslant 2^{-q} .$$

We associate the objective function $c^T x$ with a "potential function" $f(x)$ given by

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

Theorem 2. Either (i) $c^T x^{(k+1)} = 0$
or (ii) $f(x^{(k+1)}) \leqslant f(x^{(k)}) - \delta$
where δ is a constant and depends on the choice of the value of the parameter α.

A particular choice that works: If $\alpha = \frac{1}{4}$, then $\delta \geqslant \frac{1}{8}$.

3.2 The Basic Step

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

1. Perform the projective transformation $T(a, a_0)$ of the simplex S that maps input point a to the center a_0.
2. Optimize (approximately) the transformed objective function over an inscribed sphere to find a point b'.
3. Apply the inverse of T to point b' to obtain output b.

We describe each of the steps in detail.

3.3 Projective Transformation $T(a, a_0)$

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

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 $T(a, a_0)$ is given by

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

Its inverse is given by

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

Let f' be the transformed potential function defined by

$$f(x) = f'(T(x)) .$$

Then

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

where $c' = Dc$.

Let Ω' be the transformed affine space Ω.

$$Ax = 0 \Leftrightarrow AD\ x' = 0 .$$

Thus Ω' is the null-space of AD.

3.4 Optimization Over Sphere

Let r be the radius of the largest sphere with center a_0 that can be inscribed in the simplex S. Then

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

We optimize over a smaller sphere $B(a_0, \alpha r)$ $0 < \alpha < 1$ for two reasons:

1. It allows optimization of $f'(y)$ 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 augment the affine space $\Omega' = \{y \mid AD\ y = 0\}$ 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.

$$\text{i.e., } u, v \in \Omega'' \Leftrightarrow B(u-v) = 0 .$$

We are interested in optimizing $f'(y)$ over $B(a_0, \alpha r) \cap \Omega''$. First, we prove existence of a point that achieves a constant reduction in the potential function.

Theorem 3. $\exists$ a point $b' \in B(a_0, \alpha r) \cap \Omega''$ such that

either (i) $c'^T b' = 0$
or (ii) $f'(b') \leqslant f'(a_0) - \delta$, where δ is a constant.

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

Theorem 4. Let b' be the point that minimizes $c'^T x$ over $B(a_0, \alpha r) \cap \Omega''$. Then

either (i) $c'^T b' = 0$
or (ii) $f'(b') \leqslant f'(a_0) - \delta$

where δ is a constant and $\alpha = \frac{1}{4} \Rightarrow \delta \geq 1/8$.

Finally we describe an algorithm to minimize $c'^T x$.

Algorithm A

1. Project c' orthogonally onto null space of B:
$$c_P = [I - B^T(BB^T)^{-1}B]\, c' .$$
2. Normalize c_P:
$$\hat{c}_P = \frac{c_P}{|c_P|} .$$
3. Take a step of length $\alpha \cdot r$ along the direction $-\hat{c}_P$
$$b' = a_0 - \alpha r \hat{c}_P .$$

Theorem 5. The point b' returned by algorithm A minimizes $c'^T x$ over $B(a_0, \alpha r) \cap \Omega''$.

3.5 Derivation of the Main Algorithm

Now it is easy to see how the main algorithm was derived:

The first step simply augments AD with row e^T. The second step follows from substituting $c' = Dc$ in the first step of algorithm A. The next two steps are identical to those in algorithm A. The last step applies the inverse transform $T^{-1}(a, a_0)$ to point b'.

4. Analysis of the Algorithm

Proof of Theorem 5:

Let z be any point such that $z \in B(a_0, \alpha r) \cap \Omega''$

$$b', z \in \Omega'' \Rightarrow B(b'-z) = 0$$
$$\Rightarrow B^T (BB^T)^{-1} B(b'-z) = 0$$
$$\Rightarrow (c'-c_P)^T (b'-z) = 0$$
$$\Rightarrow c'^T(b'-z) = c_P^T(b'-z)$$
$$c_P^T (b'-z) = |c_P^T| . \hat{c}_P^T [a_0 - \alpha r \hat{c}_P - z]$$
$$= |c_P^T| \{\hat{c}_P^T (a_0 - z) - \alpha r\}$$
$$\hat{c}_P^T (a_0 - z) \leq |\hat{c}_P| . |a_0 - z| \leq \alpha r \text{ since } z \in B(a_0, \alpha r)$$
$$c_P^T (b'-z) \leq 0$$
$$c'^T (b'-z) \leq 0$$

$$c'^T b' \leq c'^T z, \text{ for all } z \in B(a_0, \alpha r) \cap \Omega''$$

Proof of Theorem 3:

Let x^* be the point in $\Omega'' \cap S$ where the minimum of $c'^T x$ is achieved.

Case 1. $x^* \in B(a_0, \alpha r)$
putting $b' = x^*$ proves the theorem.

Case 2. $x^* \notin B(a_0, \alpha r)$
Let b' be the point where the line segment $a_0 x^*$ intersects the boundary of the sphere $B(a_0, \alpha r)$.

$$b' = (1-\lambda)\, a_0 + \lambda x^* \text{ for some } \lambda \in [0,1] .$$

By definition, $b' \in B(a_0, \alpha r)$ and $a_0, x^* \in \Omega'' \Rightarrow b' \in \Omega''$
Hence $b' \in B(a_0, \alpha r) \cap \Omega''$

$$c'^T b' = (1-\lambda)\, c'^T a_0 + \lambda\, c'^T x^* = (1-\lambda)\, c'^T a_0$$
$$\frac{c'^T a_0}{c'^T b'} = \frac{1}{1-\lambda}$$

and

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

Hence

$$f'(a_0) - f'(b') = \sum_j \ln \frac{c'^T a_0}{c'^T b'} \cdot \frac{b_j}{a_{0j}} = \sum_j \ln \frac{(1-\lambda)a_{0j} + \lambda x_j^*}{(1-\lambda)a_{0j}}$$

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

Now we use the following inequality:

$$P_i \geq 0 \Rightarrow \prod_i (1+P_i) \geq 1 + \sum P_i$$
$$\Rightarrow \sum_i \ln(1+P_i) \geq \ln(1 + \sum P_i)$$

$$f'(a_0) - f'(b') \geq \ln \left[1 + \frac{\lambda}{1-\lambda} \cdot \frac{1}{a_{0j}} \cdot \sum_j x_j^*\right] = \ln \left[1 + \frac{n\,\lambda}{1-\lambda}\right]$$

$$b' = (1-\lambda)\, a_0 + \lambda x^*$$
$$b' - a_0 = \lambda(x^* - a_0)$$
$$\alpha r = |b' - a_0| = \lambda |x^* - a_0| \leq \lambda R$$

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

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

$$1 + \frac{\lambda \cdot n}{1-\lambda} \geq 1 + \frac{n \cdot \alpha/n-1}{1-\alpha/n-1} = 1 + \frac{n \cdot \alpha}{n-1-\alpha} \geq 1 + \alpha$$

$$f'(a_0) - f'(b') \geq \ln(1+\alpha)$$

Taking $\delta = \ln(1+\alpha)$,

$$f'(b') \leq f'(a_0) - \delta$$

Proof of Theorem 4:

First we prove a few lemmas.

Lemma 4.1

$$|x| \leq \beta < 1 \Rightarrow |\ln(1+x) - x| \leq \frac{x^2}{2(1-\beta)}$$

Proof:

Let $h(x) = \ln(1+x)$. Then $h'(x) = \frac{1}{1+x}$, $h''(x) = \frac{-1}{(1+x)^2}$.

By the mean value theorem,

$$\ln(1+x) = x + \frac{-1}{(1+\bar{x})^2} \cdot \frac{x^2}{2} \text{ where } |\bar{x}| \leq |x| < \beta$$

$$|\ln(1+x) - x| = \frac{x^2}{2} \cdot \frac{1}{(1+\bar{x})^2} \leq \frac{x^2}{2(1-\beta)}$$

Lemma 4.2 Let $\beta = \alpha \sqrt{\frac{n}{n-1}}$ then

$$x \in B(a_0, \alpha r) \Rightarrow \left| \sum_j \ln \frac{x_j}{a_{0j}} \right| \leq \frac{\beta^2}{2(1-\beta)}$$

Proof:

$$x \in B(a_0,\alpha r) \Rightarrow |x-a_0|^2 \leq \alpha^2 r^2$$

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

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

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

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

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

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

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

Now we prove theorem 4.

Define $\tilde{f}(x) = n \ln \frac{c'^T x}{c'^T a_0}$

Let b_m be the point in $B(a_0, \alpha r) \cap \Omega''$ where $f(x)$ achieves its minimum value.

$$\begin{aligned} f(a_0) - f(b') &= f(a_0) - f(b_m) + f(b_m) - f(b') \\ &= [f(a_0) - f(b_m)] \\ &\quad + [f(b_m) - (f(a_0) + \tilde{f}(b_m))] \\ &\quad - [f(b') - (f(a_o) + \tilde{f}(b'))] \\ &\quad + [\tilde{f}(b_m) - \tilde{f}(b')] \end{aligned}$$

If the minimum value of $c'^T x$ over $B(a_0, \alpha r) \cap \Omega''$ is zero then the theorem is trivially true. Otherwise, by theorem 3,

$$f(a_0) - f(b_m) \geq \ln(1+\alpha) \tag{1}$$

For $x \in B(a_0, \alpha r) \cap \Omega'$,

$$f(x) - f(a_0) + \tilde{f}(x)) = \sum_j \ln \frac{c'^T x}{x_j} - \sum_j \ln \frac{c'^T a_0}{a_{0j}} - n \sum \ln \frac{c'^T x}{c'^T a_0}$$

$$= -\sum_j \ln \frac{x_j}{a_{0j}}$$

$$| f(x) - (f(a_0) + \tilde{f}(x)) | = \left| \sum_j \ln \frac{x_j}{a_{0j}} \right| \leq \frac{\beta^2}{2(1-\beta)} \tag{2}$$

by lemma 4.2.

Since $\tilde{f}(x)$ depends on $c'^T x$ in a monotonically increasing manner, $\tilde{f}(x)$ and $c'^T x$ achieve their minimum values over $B(a_0, \alpha r) \cap \Omega''$ at the same point, viz b',

$$\tilde{f}(b_m) \geq \tilde{f}(b') \tag{3}$$

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

$$f(a_0) - f(b') \geq \ln(1+\alpha) - \frac{\beta^2}{(1-\beta)}$$

$$\geq \alpha - \frac{\alpha^2}{2} - \frac{\alpha^2 n^2}{n(n-1)[1 - \alpha\sqrt{\frac{n}{n-1}}]}$$

Define

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

$$f(a_0) - f(b') \geq \delta$$

Observe that

1. $\lim_{n\to\infty} \delta(n) = \alpha - \frac{\alpha^2}{2} - \frac{\alpha^2}{1-\alpha}$
   If $\alpha = \frac{1}{4}$, then $\lim_{n\to\infty} \delta(n) \geq \frac{1}{8}$
2. If $n \geq 4$, $\alpha = \frac{1}{5}$ then $\delta \geq \frac{1}{10}$

Proof of Theorem 2:

By Theorem 3,

$$f'(b') \leq f'(a_0 - \delta) \quad or \quad c'^T b' = 0$$

Applying the inverse transform $T^{-1}(a, a_0)$ and substituting $x^{(k)} = T^{-1}(a_0)$, we obtain $x^{(k+1)} = T^{-1}(b')$,

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \delta \quad \text{or } c^T x^{(k+1)} = 0$$

Proof of Theorem 1:

Suppose the condition $c^T x^{(k)} = 0$ did not occur in the first m steps. Then

$$f(x^{(k+1)}) \leq f(x^{(k)}) - \delta \quad \text{for } k = 0, 1, \ldots, m-1$$

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

$$\sum_j \ln \frac{c^T x^{(k)}}{x_j^{(k)}} \leq \sum_j \ln \frac{c^T a_0}{a_{0j}} - k\delta$$

$$n \ln \frac{c^T x^{(k)}}{c^T a_0} \leq \sum_j \ln (x_j^{(k)}) - \sum_j \ln a_{0j} - k\delta$$

But $x_j^{(k)} \leq 1$, $a_{0j} = \frac{1}{n}$, so

$$\ln \frac{c^T x^{(k)}}{c^T a_0} \leq \ln(n) - \frac{k}{n}\delta$$

After $m = (O(n(q+\ln(n)))$ steps,

$$\ln \frac{c^T x^{(m)}}{c^T a_0} \leq -O(q).$$

Then $x = x^{(m)}$ is the output of the algorithm, and

$$\frac{c^T x}{c^T a_0} \leq 2^{-q}.$$

If the condition $c^T x^{(k)} = 0$ occurs before m steps, we terminate the algorithm with $x = x^{(k)}$ as the output.

$$c^T x = 0.$$

5. Incremental Computation of Inverse

5.1 A Modification to the Main Algorithm

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

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

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

Substituting $B = \left[\frac{AD}{e^T}\right]$

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

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 inverse:

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.)

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

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

We use $(AD'^{(k)}{}^2A^T)^{-1}$ in place of $(AD^{(k)}{}^2A^T)^{-1}$. 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 5.3. For the purpose of this section take $\sigma^{(k)} = 1$.
2. For $i = 1$ to n do

   if $\left[\frac{D'^{(k+1)}_{ii}}{D^{(k+1)}_{ii}}\right]^2 \notin [\frac{1}{2}, 2]$, then let $D'^{(k+1)}_{ii} = D^{(k+1)}_{ii}$; and update $(AD'^{(k+1)}{}^2A^T)^{-1}$ 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 u and v. Then its inverse can be modified by the equation

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

Given M^{-1}, computation of $(M + uv^T)^{-1}$ 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

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

where 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 $O(n^2\ell)$ time to obtain the new inverse.

5.2 Correctness of the Modified Algorithm

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':

$$\left.\begin{array}{l} \text{Minimize } c'^Tx \\ \text{subject to } Bx = 0 \\ \text{and } x^TQx \leqslant \alpha' r \end{array}\right\} . \tag{5}$$

(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 $h(x) = x^TQx$.
It's gradient is given by $\nabla h(x) = 2Qx$.
At the optimal solution 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

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

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

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

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

This requires inverse the of $BQ^{-1}B^T$. Substituting $B = \left[\frac{AD}{e^T}\right]$ we get

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

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

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

where M, 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 $E_{ii}^2 \in [\frac{1}{2}, 2]$. 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 $Q_{ii} = E_{ii}^{-2} \in [\frac{1}{2}, 2]$

$$\frac{1}{2} x^Tx \leqslant x^TQx \leqslant 2x^Tx . \tag{11}$$

Hence

$$B(a_0, \frac{\alpha'}{2} r) \leqslant \{x | x^TQx \leqslant \alpha' r\} \subseteq B(a_0, 2\alpha' r) .$$

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

$$B(a_0, \frac{\alpha}{4} r) \cap \Omega'' \subseteq \{x | \begin{array}{l} x^TQx \leqslant \alpha' r \\ Bx = 0 \end{array}\} \subseteq B(a_0, \alpha r) \cap \Omega'' . \tag{12}$$

Because of the first inclusion,

$$\begin{array}{lcll} \min f'(x) & & \min f'(x) & \\ \text{subject to} & \leqslant & \text{subject to} & \leqslant f'(a_0) - \ln(1 + \alpha/4) \\ x \in \{x | \begin{array}{l} x^TQx \leqslant \alpha' r \\ Bx = 0 \end{array}\} & & x \in B(a_0, \alpha/4\, r) \cap \Omega'' & \end{array} \tag{13}$$

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

$$f'(b') \leqslant f'(a_0) - \ln(1+\alpha/4) . \tag{14}$$

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

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

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

5.3 Performance of the Modified Algorithm

In this section 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 $|\, b'-a_0 \,| \leqslant \alpha r$

Substituting $b' = T(x^{(k+1)})$, $a_0 = \frac{1}{n} e$ and

$$T(x) = \frac{D^{-1}x}{e^T D^{-1} x}, \quad r = \frac{1}{\sqrt{n(n-1)}}$$

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

Let

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

Then

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

Let

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

Therefore

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

Recall

$$D^{(k)} = \text{Diag}\,\{x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\}$$

$$D^{(k+1)} = \text{Diag}\,\{x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}\}$$

$D'^{(k)}$ 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 $\left[\frac{D'^{(k+1)}_{ii}}{D_{ii}^{(k+1)}} \right]^2 \notin [\frac{1}{2}, 2]$ we reset $D'^{(k+1)}_{ii} = D_{ii}^{(k+1)}$.

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

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

Just after an updating operation for index i,

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

Just before an updating operation for index i,

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

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

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

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

$$\Delta\delta_i^{(k)} = \delta_i^{(k+1)} - \delta_i^{(k)} = \ln \frac{D_{ii}^{\text{new}}}{D_{ii}^{(k+1)}} \cdot \frac{D_{ii}^{(k)}}{D_{ii}^{\text{old}}} = \ln \frac{\sigma^{(k)} x_i^{(k)}}{x_i^{(k+1)}}$$

$$= -\ln \Delta_i^{(k)}$$

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

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),

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

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

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

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

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

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

Hence

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

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

$$\leqslant \sum_i \frac{[\Delta_i^{(k)}-1]^2}{2(1-\beta)} + \sqrt{n}\, \beta$$

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

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

From equations (22) and (24)

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

6. Extensions to the Main Algorithm

In this section we show how the restrictions made in Section 2 can be removed.

6.1 Feasibility Problem and Optimization Problem

Consider the problem of deciding if a system of linear inequalities has a solution. After introducing slack variables and eliminating variables unconstrained in sign we get the following problem.

Problem P1: $Ax = b, x \geqslant 0.$

It is a well-known fact used in the ellipsoid algorithm that problem P1 is feasible if and only if the following "perturbed" problem P2 is feasible.

Problem P2: $Ax = b + 2^{-O(L)} \cdot e, x \geqslant 0,$ e is vector of all 1's.

Now we show that P1 can be solved in terms of the simplied problem defined in Section 2.

Let $x_0 > 0$ be an arbitrary point, strictly interior to the positive orthant. We introduce a problem P3 by defining an artificial variable λ.

Problem P3: minimize λ

subject to $(Ax-b) = \lambda(Ax_{0-b})$

$$x \geqslant 0, \quad \lambda \geqslant 0 .$$

Note that $x = x_0, \lambda_0 = 1$ is a feasible solution to P3, which we take as the starting point. Let λ_m be the minimum value of the objective function. $\lambda_m = 0$ corresponds to a feasible solution to problem P1. But it turns out that finding a feasible solution with $\lambda' < 2^{-O(L)}$ is enough. More precisely, we have the following theorem.

Theorem 6.1.

The following statements are equivalent:

(1) Problem P1 is feasible

(2) Problem P2 is feasible

(3) There exists a feasible solution to Problem P3 having objective function value λ' such that

$$\frac{\lambda'}{\lambda_0} \leqslant 2^{-O(L)} .$$

Proof:

(1) $\Rightarrow$ (3) If $x = x^{(0)}$ is a feasible solution to P1, then $x = x^{(0)}, \lambda' = 0$ is a feasible solution to P3, with $\frac{\lambda'}{\lambda} \leqslant 2^{-O(L)}$.

(3) $\Rightarrow$ (2) Let $x = x', \lambda = \lambda'$ be a feasible solution to P3 with

$$\frac{\lambda'}{\lambda_0} \leqslant 2^{-kL}$$

$$(Ax-b) = \lambda'(Ax_0-b)$$

$$|(Ax-b)_i| \leqslant |\lambda'| \cdot |(Ax_0-b)_i|$$

$$\leqslant 2^{-kL} \cdot 2^{O(L)} \leqslant 2^{k'L} ,$$

by proper choice of k given a desired value for k'.

$\therefore \quad Ax = b + 2^{-O(L)} \cdot e$

(2) $\Rightarrow$ (1) Proof is same as in the ellipsoid algorithm.

To further transform P3, we can introduce a bound $M = 2^{O(L)}$ without loss of generality.

$$\sum_i x_i \leqslant M .$$

After introducing a slack variable and scaling by M, we get equations defining a simplex:

$$\sum x_i' = 1, \quad x_i' \geqslant 0 .$$

The choice of the initial starting point x_0, which was unspecified so far, can be made so that it corresponds to the center of the simplex.

Alternatively, we could also have used a projective transformation to transform P_+ into a simplex. In this case, the bound $\sum x_i \leqslant M$ can be used to conclude that

$$c_m = 2^{-O(L)} \Rightarrow \lambda' = 2^{-O(L)} ,$$

where c_m is the minimum value of the objective function in the transformed space. Now the problem is in a form suitable for applying the main algorithm.

Finally consider the following optimization problem, where a feasible starting point is not known.

Problem P4: $\min \ c^T x$

subject to $Ax = b, x \geqslant 0$

This problem can be solved by the "two-phase" method. First we solve the feasibility problem $Ax = b, x \geqslant 0$ by the method described above. If it is found to be feasible we also get a feasible point as a by-product. We bring it into the center of the simplex by a projective transformation and solve the resulting problem.

6.2 Inhomogeneous system of equations

Consider the system of equation $Ax = b, \sum x_i = 1$. We can transform the jth equation $a_j^T x = b_j$ to

$$a_j^T x - b_j \sum_i x_i = 0$$

i.e.

$$\sum_i (a_{ji} - b_j) x_i = 0 .$$

Thus we get an equivalent system of equations

$$A'x = 0 , \quad \sum x_i = 1 .$$

Suppose the minimum value of the objective function is c_m. Define a new objective function

$$\sum_i c_i x_i - c_m = \sum_i c_i x_i - c_m \sum_i x_i = \sum_i (c_i - c_m) x_i ,$$

which is homogenous, linear and has minimum value zero. Note that in order to carry out this transformation it is necessary to know the value of c_m in advance.

6.3 Sliding objective function method

This method works without the knowledge of the minimum value of the objective function. We assume that initially we are given lower and upper bounds l_0, u_0 on the objective function. (Otherwise we can take $l_0 = -2^{O(L)}$ and $u_0 = 2^{O(L)}$.) During the course of the algorithm we update the values of lower and upper bounds. The difference between the two is called the "range". The algorithm is divided into phases. Each phase reduces the range by 2/3 and requires no more than $n(k + \ln(n))$ steps where the constant k is such that

$$(1 - \frac{\delta}{n})^{kn} \leqslant \frac{1}{2} .$$

Let c_m be the minimum value of the objective function, unknown to us. Suppose we run the algorithm pretending that c_m' is the minimum value of the objective function i.e. we try to minimize $c^T x - c_m'$. We also modify algorithm A as follows: after finding b' we check if $c^T b' < c_m'$. If so, we choose a point b'' on the line segment $a_0 b'$ such that $c^T b'' = c_m'$ and return b'' instead of b' as the output of algorithm A.

Case 1: $c_m \leqslant c_m'$.

In this case all claims about the performance of the algorithm continue to be true. Proofs of theorems 3 and 4 require the following modification.

Let x_m be the point where the objective function $c'^T x$ (in its homogeneous form) achieves its minimum over $B(a_0, \alpha r) \cap \Omega''$.

Suppose $c'^T x_m < 0$.

In the proof of theorem 3, we redefine x^* to be the point on the line segment $a_0 x_m$ such that $c'^T x^* = 0$. The rest of the proof remains the same. Theorem 4 is trivially true because the point b'' returned by the modified algorithm A is such that $c'^T b'' = 0$.

If $c'^T x_m \geqslant 0$, the proofs of theorems 3 and 4 require no modification.

In either case, theorem 4 assures that we still get a reduction of δ in the potential function or find a point that achieves the assumed minimum c_m'.

Case 2: $c_m' < c_m$.

If we are lucky we get a reduction of δ or more in the potential function. Otherwise, we get a proof that the assumed minimum is lower than the actual minimum.

Now we describe a phase of the algorithm. Let l, u be the lower and upper bounds at the beginning of a phase. We set up tentative lower and upper bounds l', u' given by

$$l' = l + \frac{1}{3}(u-l)$$

$$u' = l + \frac{2}{3}(u-l)$$

We pretend that l' is the minimum value of the objective function and run the algorithm. We say that a tentative lower bound l' becomes invalid when for the first time we get a proof that it is lower than the true minimum. We say that a tentative upper bound u' becomes invalid when for the first time we discover a feasible point with objective function value less than u'. One of these events must occur in $n(k + \ln(n))$ or fewer steps, for suppose the lower bound did not become invalid in these many steps, then we must have

achieved a reduction of δ in the associated potential function in each step, forcing $c^T x$ to become less than u'.

When a tentative upper or lower bound becomes invalid, the phase terminates and we make that tentative bound a firm bound for the next phase. Thus we reduce the range by 2/3 in each phase. In $O(nL)$ steps we can reduce the range from $2^{O(L)}$ to $2^{-O(L)}$ and then round the solution to the exact optimum as in the ellipsoid method.

REFERENCES

1. V. Klee and G. L. Minty, "How good is the simplex algorithm?" in O. Shisha (ed.) Inequalities III, Academic Press, New York, 1972, p. 159-179.
2. L. G. Khachiyan, "A polynomial Algorithm in Linear Programming," Doklady Akademiia Nauk SSSR 244:S (1979), p. 1093-1096, translated in Soviet Mathematics Doklady 20:1 (1979), p. 191-194.
3. M. Grotschel, L. Lovasz, A. Schrijver, Then Elipsoid Method and its Consequences in Combinatorial Optimization, Combinatorica 1(2) 1981, p. 169-197.
4. Dantzig, G. B., Linear Programming and Extensions, Princeton University Press, Princeton, NJ (1963).