Derivation of the Ellipsoid Algorithm

Mike Karr*, Srinivasan Krishnan† and John Reif†

June 24, 1991

Abstract

In the recent past a number of algorithms for linear programming have been published. In this paper we present an informal derivational framework for linear programming algorithms and derive the ellipsoid method using the ideas in this framework. The natural exposition circumvents the tedious posteriori correctness proof.

1 Introduction

In this paper, we will present our framework of derivation and derive the ellipsoid algorithm in this framework. Similar derivational work for efficient graph algorithms has been undertaken in the past [4], but we are not aware of such efforts for algebraic algorithms.

The ellipsoid algorithm, proposed by Khachian [2], is a good starting point for such efforts for it is the first polynomial time algorithm for linear programming. In addition, it has a long and tedious correctness proof based on an inductive assertion on the iteration number. We first consider only a simple special case of the problem and develop the work required for this special case. Then we show that using affine transforms, we may reconstruct the entire algorithm.

With the advent of the ellipsoid method, a number of efforts were made to attack the feasibility version of the linear programming problem. A recent effort, by Vaidya [7], pointed out that such efforts can be captured under one generic scheme:

Let $S \subset R^n$ be a convex set for which there is an oracle with the following property: The oracle accepts as input any point z in R^n. If $z \in S$ the oracle returns "yes". Otherwise it returns a "no" along with a separating hyperplane. A generic iterative algorithm is then as follows. We maintain a region R such that $S \subset R$. At each iteration we choose a test point $z \in R$ and call the oracle with z as input. We halt if $z \in S$. If $z \notin S$, the oracle returns a vector c such that $\forall x \in S, c^T x \geq c^T z$. Then $S \subset R \cap \{x : c^T x \geq \beta\}$, where $\beta \leq c^T z$. As the algorithm proceeds, R shrinks; its volume decreases at a constant rate. If S is non-empty, it contains a ball of radius 2^{-nL}, and the algorithm halts with a point in S before the volume of R falls below this value. Otherwise the algorithm halts the first time the volume of R falls below this value. During the course of the algorithm the description of R may become complicated and so we may need to approximate R by a simpler region that contains R. This trades volume for computational efficiency in a manner that the algorithm still converges.

0* -Software Options Inc. 22 Hilliard Street Cambridge, MA 02138

† Duke University, Dept. of Computer Science, Durham, NC 27705.

This work was supported by Air Force Contract AFOSR-87-0386, ONR Contract N00014-87-K-0310, DARPA/ISTO Contract N00014-88-K-0458, ARO Contract DAAL03-88-K-0195, and NASA/CESDIS Subcontract 550-63 NAS 5-30428 URSA.

Similarly, it has also been noted that the family of interior point algorithms follow a generic scheme. This approach considers the optimization of a linear function over a polytope using an interior point technique, i.e, by generating a sequence of interior points in the polytope, which converge to the optimum in polynomial time. In this model, each hyperplane exerts a force (away from it) and there is also an external force in the direction of the gradient of the objective function. All the algorithms obtainable from potential functions can then be obtained in this model by choosing the potential function as the gradient of the objective function. It is also possible to choose as gradients functions that are not potential functions. In general, let p denote a hyperplane of the polytope and d_p the distance of the current interior point to the plane p. Let w_p denote the weight associated with plane p and $\hat{e_p}$ denote the normal to this plane pointing towards the interior of the polytope. Let $\bar{F}$ denote the external force along the direction of the objective function. Then these forces balance when:

$\hat{F} + \sum \frac{w_p}{d_p} \hat{e_p} = 0$

(under the the potential function model)

This results in a trajectory of motion as the external force rises which converges to the optimum.

A third crucial commonality that is prevalent in both the ellipsoid like methods and the interior point algorithms is that both use the idea of transforms extensively. The most common transforms that are used are affine transforms. The Karmarkar algorithm however [3] uses projective transforms (affine transforms are a special case of projective transforms). In general, the desirable properties which a transform used in this context must possess are:

- The transform must be invertible and must be closed under composition, i.e we require it to form a group under composition.
- The transform must preserve the incidence properties of the polytope.

It turns out that projective (and therefore affine) transforms satisfy this property. The intuition behind using transforms is to transform unfavorable instances of a problem (with respect to the algorithm) into relatively favorable ones.

The approach used by the randomized algorithms based on geometry is different. As noted in the previous chapter, the main idea here is that the optimum in d dimensions consists of d or fewer constraints and the goal is to drop the redundant constraints quickly. The "small" sized problems that result are solved by using the traditional simplex method.

2 Proposed framework of tools

We now present an informal toolkit of ideas that may be used for developing and experimenting with new algorithms. We do not claim any formal basis for completeness; our goal is to point out that these ideas have been crucial to the development of LP algorithms and furthermore, they fit in an overall framework of approaches. The set of ideas presented below are not limited to ellipsoid like methods, they have been set in a larger context citemythesis. In the next section we will use a subset of these ideas to derive the ellipsoid algorithm.

The first three items refer to two such approaches towards algorithm design. The fourth brings out the concept of transforms which is integral to the development of both ellipsoid-like and interior point algorithms. The fifth addresses the issue of computational efficiency involved in projection and solution of linear systems. Finally we point out analysis tools that offer a quick overview to estimating the complexity of algorithms.

- The generic feasibility approach: Let P denote the polytope under consideration. Consider a region R (s.t. $P \subset R$) of R^d that has a geometric characterization (such as a spheroid or a simplex). If we can approximate $R \cap H$ (where H is a halfspace) by R' such that $v(R') < v(R)$ in a suitable metric, (v refers to the volume of the region), this forms the basis for a polynomial algorithm for the feasibility problem.

- The generic interior point approach: Consider a force function $\hat{F}$ and the hyperplanes defining the polytope. Let $\hat{f_{p_i}}$ denote the force exerted by the i^th plane. Then when

  $\hat{F} + \hat{f_{p_i}}\hat{e_p} = 0$

  the forces balance. This results in a trajectory of motion as the external force rises. This trajectory is then approximated by a discrete sequence of interior points at each iteration of the algorithm.

- Random Sampling: This approach uses the idea that the solution of the linear programming problem with a large number of constraints can be obtained more readily by sampling a subset of the constraints and solving subproblems of a smaller size. When the number of constraints is large, we recommend this approach as a means of reducing the complexity of the algorithm.

- Transforms: The techniques mentioned above would be of limited use without the central technique of transforms. To reapply the the above techniques at each iteration, it is necessary to apply a transformation of space. There is a second major advantage to transforms. By using them we may restrict the algorithm to the solution of special cases of the problem. The generality of the solution automatically follows if any linear program can be reduced to this special form. We use this in the next section to develop the ellipsoid algorithm. These transforms must form a group under composition to ensure invertibility and closure under composition. Transforms probably capture the central ideas of our derivational framework best.

- Underlying Operations: LP algorithms usually involve only a few key operations, the most important being the projection of vectors and the solutions of linear systems. This technique of projecting a vector is usually accomplished by the least squares method [6] and it usually dominates the computational work required at each iteration of the algorithm. Therefore any practical scheme for LP must have efficient implementations of this method. An even more fundamental operation here is the solution of linear systems. Practical experimentation with algorithms therefore requires an advanced programming environment that provides these facilities with ease.

- Analysis tools: Now we point out certain facts that simplify the analysis of algorithms considerably. We restrict ourselves to analyzing the number of iterations in the algorithm. The work in each iteration is relatively simpler to analyze.

  - Feasibility problems:

    The key metric used here to judge the progress of the algorithm is the volume of the feasible region R. At each iteration of the algorithm, the volume of the R is reduced. The rate of reduction of R decides the speed of convergence.

    Fact 2.1 *If a sequence of numbers a_i starting at some positive value a_0 falls at the rate of $O(1 - \frac{1}{n})$ $(n > 1)$ then the numbers $a_0, a_n, a_{2n}, \ldots$ fall at a constant rate $\alpha < 1$.*

Proof follows from the following:
$(1 - \frac{1}{n}) \rightarrow \frac{1}{e}$ as $n \rightarrow \infty$.

* If the rate of reduction R is $O(1 - \frac{1}{n})$ every iteration then, the number of iterations required is $O(n^2 L)$.
* If the rate of reduction R is by a constant factor every iteration then, the number of iterations required is $O(nL)$. (follows from above and fact 2.1)

3 Deriving the Ellipsoid algorithm

3.1 Preliminaries

We consider linear programming in the following form:
Given $A \in R^{mxn}$,
$b \in R^m$,
find $x \in R^n$ such that:
$Ax < b$ ...(1)
There are various forms of the linear programming problems all of which are equivalent. (For further details, see [1],[5])

In our model, we will assume that the vectors in consideration have components that are rational numbers, both absolute values and denominators of which are bounded by 2^L. Observing that x is a n-vector of these numbers:
$\| x \| \leq n * 2^L$ ...(2)
The bound above is not really tight, but will suffice for our arguments.

For the derivation of the algorithm, we will use the generic feasibility approach that we have stated. The region R that we shall consider will be an ellipsoid. The initial volume of the ellipsoid will be large enough to enclose any possible solutions. Hyperplanes at the origin will then be used to derive lower volume ellipsoids. Note that in the following discussion, spheroids are special cases of ellipsoids.

Since we have restricted our solutions to satisfy inequality(2), any solutions to (1), if they exist must lie inside a sphere of radius $n2^L$. Thus we have:

Lemma 3.1 *If the system of linear strict inequalities (1) has a solution then, all solutions lie inside a sphere of radius $n2^L$ at the origin.*

Thus we have an upper bound on the volume of the spheroid at the origin that is guaranteed to contain the solution if one exists. Now, we derive the lower bound:

Lemma 3.2 *Consider the polytope: $P = x \in R^n : Ax < b$ If $P \neq \Phi$ then there must exist n+1 linearly independent vertices of P.*

Proof: Any set of n points in an n-dimensional space determines a hyperplane. If P lies on a hyperplane, then consider a point $x \in$ P. Let ϵ be the smallest distance of x from th facets of P. Since $x \in$ P, the sphere of radius ϵ at x must lie inside P. But this is absurd because no n-1 dimensional hyperplane can contain an n-dimensional space. Hence our assumption must be wrong, and P contains n+1 points that are linearly independent.

Now, we can establish the lower bound: If
$Ax < b$
has a solution then from lemma1 we know that:

$Ax < b$, $x_i < 2^L$ ...(3)

has a solution. Hence there must exist n+1 linearly independent points in this polytope(3). Let these points be $v_0, ..., v_n$. Then the volume of this polytope is at least the volume of the convex hull of these points, which is given by:

$$\left| det \begin{pmatrix} 1 & 1... & 1 \\ v_0 & v_1... & v_n \end{pmatrix} \right| /(n!)$$

Each v_i can be written as u_i/D_i where D_i is a determinant whose value is at most 2^L (follows from our model of the vectors). As a result, the volume of the convex hull is at least:

$2^{-nL}/n! > 2^{-(n+2)L}$

Therefore:

Lemma 3.3 *The system of inequalities (1) defines a polytope which has volume at least* $2^{-(n+2)L}$ *if a solution exists.*

Now we derive a systematic method for checking if a solution exists by considering successively smaller ellipsoid, starting with the "sufficiently large spheroid"(given by lemma1) and terminating with the "sufficiently small one" (given by lemma3).

3.2 The work in each iteration

To recall, our overall approach is as follows:
At each iteration we will check if the center of the current ellipsoid is a solution. If it is not, then we consider any violated inequality and construct an ellipsoid of lower volume that must also contain any possible solution. In this manner, we consider successively smaller ellipsoids in every successive iteration, until the ellipsoids become so small that they cannot contain a solution anymore.

First, we check if the origin is a solution to the given system. If it is, then we have obtained a solution to the problem. If not, then there must be at least one inequality that is violated. For simplicity let us consider that the constraint violated is of the form

$-x_1 < b_i$ ...(4)

Hence the solution if one exists must lie in the intersection of the spheroid and the halfspace $-x_1 < b_i$ (This situation is depicted in figure 1.)

Following our derivational framework, we now encapsulate this spheroid S and the half-space $x_1 \geq 0$ by an ellipsoid E of lower volume.

From considerations of symmetry, one possible approach is to restrict the center of this new ellipsoid to lie on the x_1 axis between 0 and 1. Our attempted goal is depicted in figure 2.

The major axes (n-1 of them) can then be determined by the requirement that the ellipsoid intersect the n -1 sphere (defined by the intersection of the original sphere and the hyperplane $x_1 = 0$) as well as the point (1, 0, ..., 0) on the x_1 axis. Let the the major axes be of radius b and the minor axis radius a as shown. The equation for this new ellipsoid may be now written as follows:

$(x_1 - (1-a)/a)^2 + \sum_{i=2}^{n}(x_i/b)^2 = 1$

Now we need to determine a and b. For this we plug in the constraint that when

$x_1 = 0$, $\sum_{i=2}^{n} x_i^2 = 1$.

Hence we obtain the relation:

$((1-a)/a)^2 + (1/b)^2 = 1$

$\Rightarrow b = a/(\sqrt{(2a-1)})$

The volume of this new ellipsoid (equal to the product of its axes) is given by

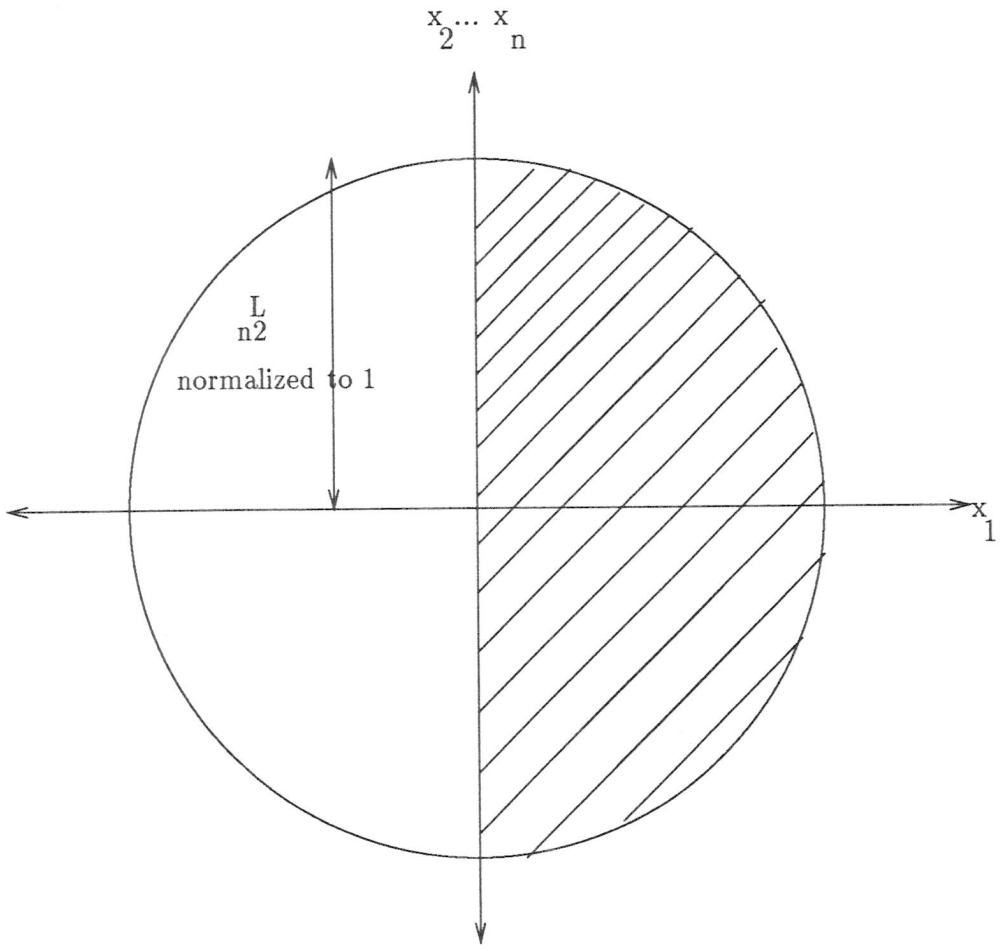

Figure 1: Intersection of the sphere and the halfspace

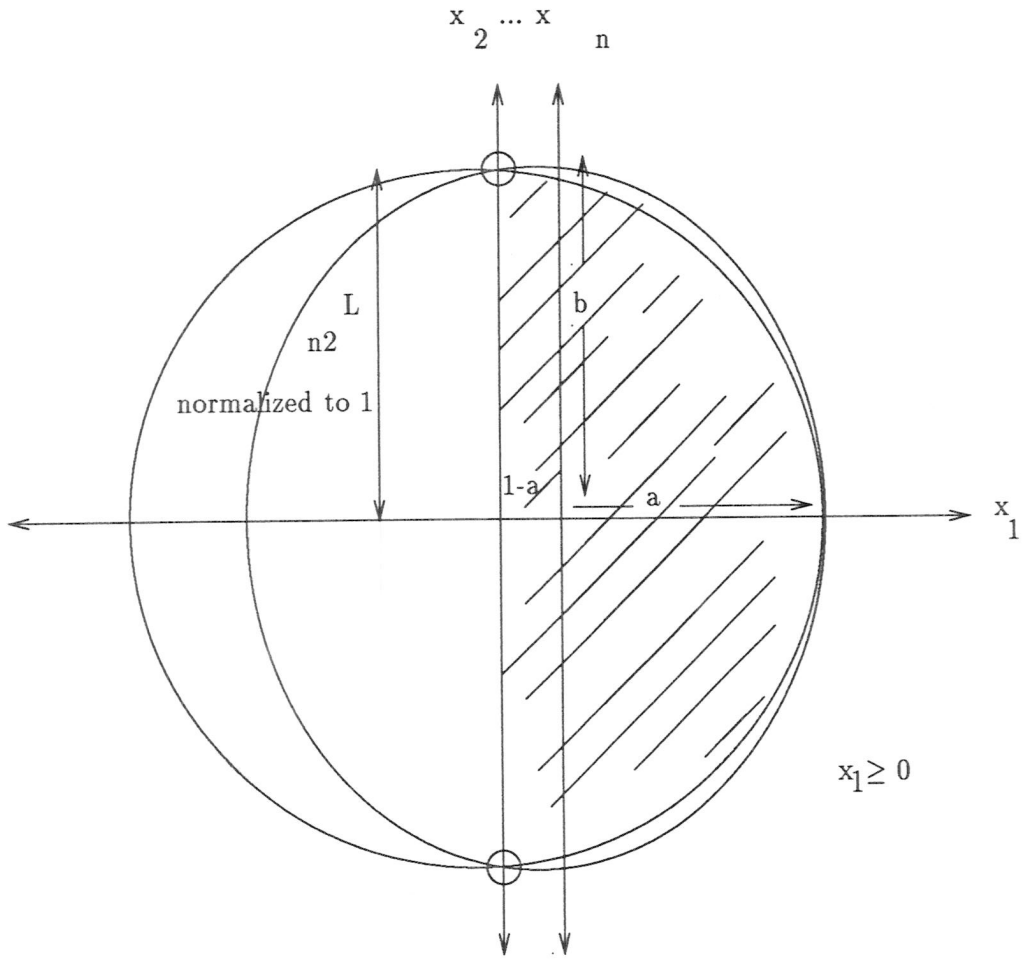

Figure 2: encapsulating the region by an ellipsoid

$v = ab^{n-1} = a^n(2a-1)^{-(n-1)/2}$

The conditions that we have imposed on our ellipsoid still do not uniquely determine it; we have one more degree of freedom. We use this to minimize the volume of the ellipsoid. We can determine a by setting the logarithmic derivative of v w.r.t. a equal to 0. This gives us,

$n/a - (n-1/2)(2/(2a-1)) = 0$

$\Rightarrow a(n+1) = n \Rightarrow a = 1/(n+1)$

We can now determine the the ratio of the new ellipsoid to the original spheroid which had unit volume.

The ratio is given by

$R = (n/(n+1))^n((2n/n+1)-1)^{-(n-1)/2}$

$= (n/n+1)(n^2/(n^2-1)^{(n-1)/2}$

$= n^{2n-1}/((n+1)^{(3n-1)/2}(n-1)^{(n-1)/2})$ ...(5)

Now, for all $n > 0, 1+n \leq e^n, 1-n \leq e^{-n}$ Using this in (4), we obtain:

$R < e^{-1/n+1}e^{-1/(n^2-1} < 2^{-1/2(n+1)}$

We have now obtained an ellipsoid of lower volume. Since we have both an upper and lower bound on the volume of the polytope such that a solution exists. The natural question to ask is how many recursive shrinkings of size $2^{-(n+2)L}$ are needed to shrink the polytope from its initial maximum size to the minimum allowable size. If k is the desired number, then we require:

$(2n^2 2^{2L})2^{-k/2(n+1)} < 2^{-(n+2)L}$

Using $k = 16n(n+1)L$, it can be verified that the inequality above can be satisfied. Note that this analysis is only a simpler case of the analysis tools that we had stated in the context of the feasibility problem.

In matrix form this ellipsoid would be described by the inequality:

$(x-c)^T A^{-1}(x-c) \leq 1$

where c is the center of the ellipsoid.

Thus $A^{-1}$is a diagonal matrix given by $diag(1/a^2, 1/b^2, ..., 1/b^2)$. Or,

$A = diag(a^2, b^2, ..., b^2)$

$= b^2 diag((a/b)^2, 1, ..., 1)$

$= b^2(I + diag((a/b)^2 - 1, 0, ..., 0))$

Putting the values of a and b obtained and simplifying we get,

$A = (n^2/(n^2-1)(I - (2/(n+1))diag(1, 0, ...0))$ ...(6)

The above is the required lower volume ellipsoid for a specific type of inequality. In general the inequality not satisfied by the origin will be of the form

$a_i x < b_i$

In this case we wish to encapsulate the region given by the part of the sphere that lies in the halfspace $a_i x < 0$ This situation is depicted in figure 2.3.

Observe however that in order to be able to apply the techniques that we have derived, all that we need to do is to apply a rotation to this space such that the violated inequality again assumes the form of inequality (4). To this end, we study the effects of such a transform on the matrix A. Recall rotational and in general affine transforms form a group under composition. We will use them to convert the case of violating any equality to the special case whose solution we have developed.

Suppose we have an rotational transform T from an original domain to a transformed domain. Suppose also that we have a positive definite matrix B, which we use in the the transformed domain in the form:

$y^T B^{-1} y$

We wish to determine the corresponding matrix A in the original domain. This matrix may be determined as follows:

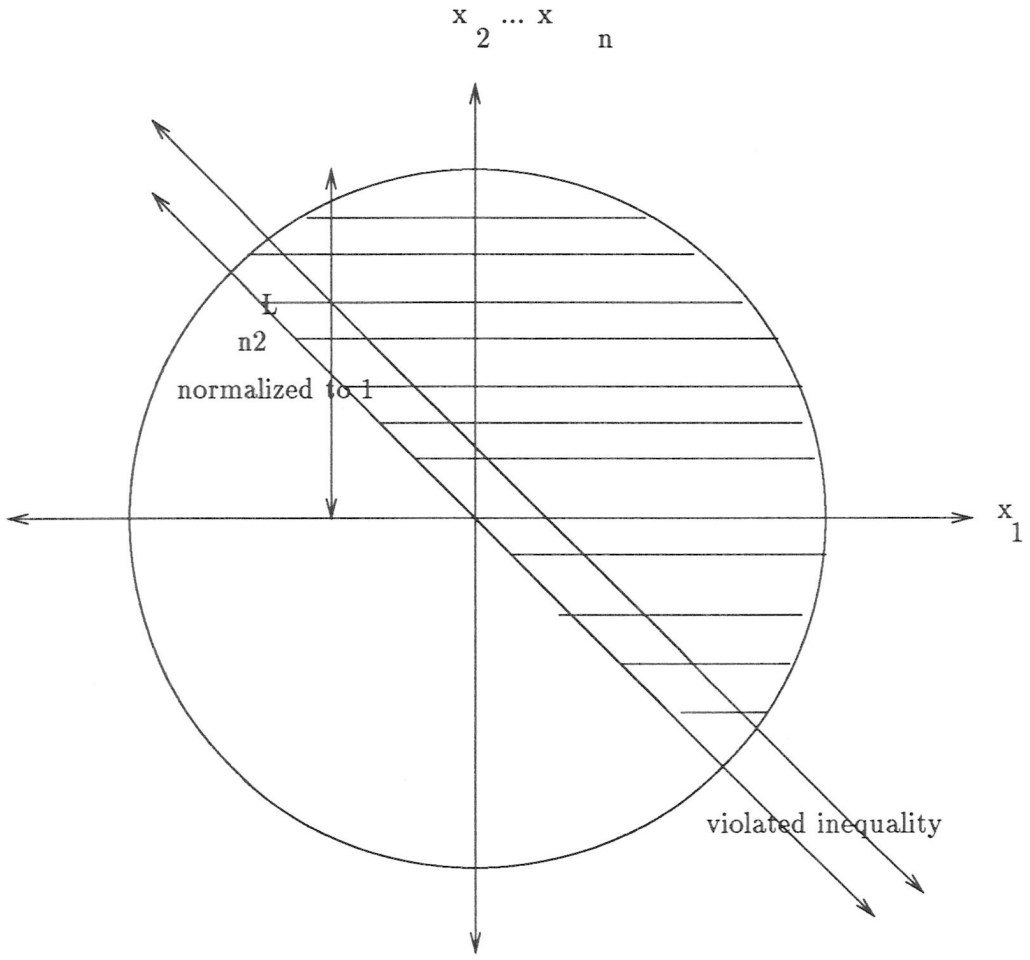

Figure 3: Intersection of the sphere and a halfspace of any orientation

$y^T B^{-1} y = (Tx)^T B^{-1} Tx$
$= x^T (T^T B^{-1} T) x$
Hence $A^{-1} = (T^T B^{-1} T)$
$\Rightarrow A = T^{-1} B T^{-T}$. ...(7)

By construction, the transform T maps any unit vector u to (-1, 0, ...0). Let B now be the canonical Khachian matrix that we had obtained before, given by equation (6).

A is then given by:
$A = T^T (n^2/(n^2-1)(I - (2/(n+1))diag(1, 0, ..., 0))T$
Using $T^T T = I$ and $Tu = (-1, 0, ...0)$
and simplifying, we get
$A = (n^2/(n^2-1))(I - (2/n+1)uu^T)$ ... (8)

Note that we can always replace uu^T by $aa^T/(a^T a)$ for non unit vectors a. Thus we have now found the effect of an orthogonal transformation on the canonical Khachian matrix. Hence, we can now solve for the required ellipsoid irrespective of the orientation of the halfspace at the origin.

In order to be able to apply this technique recursively, all we need is to use a transform that maps this new ellipsoid back to a spheroid at the origin. Note that scaling is a special case of an affine transform. If the ellipsoid is described by the inequalities
$x^T A^{-1} x \leq 1$
then since A is positive definite, we may write
$A = P^T P$.
It is then easy to see that the required transform is given by
$y = P^{-T} x$
Proof:
$y^T y = x^T P^{-1} P^{-T} x$
$= x^T (P^T P) x = x^T A^{-1} x$ □

In this transformed domain, we know how to construct an ellipsoid that encapsulates the half spheroid. We apply the same techniques that we have just developed as follows:

Consider the half-space $a^T x \leq 0$.
Then applying the transformation $x = P^T y$
$a^T x = a^T (P^T y = (Pa)^T y$
Hence, in the transformed space, the halfspace obtained is
$b^T y = (Pa)^T y = 0$
In this transformed space now, we can construct the Khachian matrix by the formula:
$B = (n^2/(n^2-1)(I - (2/(n+1))(Pa)(Pa)^T/((Pa)^T(Pa)))$
Now, using (7) we can reconstruct A by the formula
$A = P^T BP$
Simplifying we obtain,
$A = (n^2/(n^2-1)(A - (2/(n+1))(Aa)(Aa)^T/(a^T Aa)$ ...(9)

This is the expression for the Khachian matrix for every successive iteration. All that remains to be done now is to recover the center of this new ellipsoid.

We know that in the transformed space, the center of the ellipsoid is at the point:
$(1/(n+1), 0, ...0)$

First we want to undo the effect of rotation. Let T be an orthogonal transformation as before. Then the center is given by
$T^{-1}(-1/(n+1), 0, ...0)$
$= -1/(n+1)T^{-1}(-1, 0, ..., 0)$
$= -u/(n+1)$
However the the halfspace that we wish to consider in our transformed space is of the form

$b^T y = 0$

where $b = Pa$

We had said before non-unit vectors may also be handled by the same techniques simply by dividing by the magnitude of the vector. So the center we are looking for is given by the expression:

$-1/(n+1)Pa/\sqrt{((Pa)^T Pa)}$

$= -Pa/((n+1)(\sqrt{(a^T P^T Pa)}))$

$= -Pa/\sqrt{((a^T Aa)(n+1))}$

Recalling that the transformation we applied was P^{-T}, we can recover the center in the original space by simply undoing the effect of the transform. Thus the center is given by:

$(-1/(n+1))(P^T Pa)/(\sqrt{(a^T Aa)})$

$= (-1/(n+1))(Aa)/(\sqrt{(a^T Aa)})$

This is an expression for the origin of the new ellipse assuming that the center of the current ellipsoid is at the origin. Instead, if we let the current origin be translated to any point x, then all we need to do is to shift the new origin by the same x. Hence the new origin is given by the expression:

$x_{n+1} = x+$

$(-1/(n+1))(Aa)/(\sqrt{(a^T Aa)})$ ...(10)

This is exactly the expression we desire.

Consolidating our results, we are now ready to write down the ellipsoid algorithm.

3.3 The Ellipsoid algorithm

Input: An $m \times n$ system of linear strict inequalities $Ax < b$, of size L.

Output: An n vector x such that $Ax \leq b$, if such a vector exists;"no" otherwise.

Initially, we begin with a spheroid of radius $n2^L$. The Khachian matrix corresponding to this spheroid is given by $A - 0$. j counts the number of iterations made so far. t_j is the center of the ellipsoid at the j^{th} iteration. A_j is the matrix describing the ellipsoid at the j^{th} iteration; i.e, the matrix is described by the inequalities

$(x - t_j)^T B_j^{-1}(x - t_j) \leq 1$

1:(Initialize) Set $j := 0$, $t_0 := 0$

$A_0 := n^2 2^{2L} I$

If the current center of the ellipsoid is a solution, we are done. If not and the number of iterations has exceeded the quantity shown (derived earlier) then we are sure that no solution exists. Otherwise we can go on.

2: (test) If t_j is a solution to the system then return t_j (test) If $j > 16n(n+1)L$ then return "no".

3: Choose an inequality that is not satisfied by t_j say $a^T t_j \geq b$

Set

$t_{j+1} := t_j -$

$A_j a/((n+1)(\sqrt{(a^T A_j a)})$

(The new center of the ellipsoid, using expression(10))

$A_{j+1} := (n^2/(n^2-1))(A_j -$

$(2/(n+1))((A_j a)^T (A_j a)/(a^T A_j a)$

(The Khachian matrix for the new ellipsoid, using expression (9))

$j := j + 1;$

go to 2.

4 Conclusions

The goal of this paper is mainly one of exposition. We have shown that the ellipsoid algorithm can be captures in a rather intuitive geometric framework. We believe that this approach is extendible to other types of methods such as the interior point methods too.

Indeed Strang [6] has presented the *rescaling* algorithm, a simpler version of the Karmarkar algoerithm in a manner that nicely fits our derivational framework. Replacing rescaling by projective transforms and linear objective functions by the potential function approach that we had discusses, we believe will be sufficient to reconstruct the Karmarkar algorithm as well.

References

[1] C.H.Papadimitrou and K.Steiglitz. *Combinatorial Optimization.* Prentice-Hall Inc., 1982.

[2] L.G.Khachian. A polynomial time algorithm for linear programming. *Soviet Math Dokl*, 20:191–194, 1979.

[3] N.K.Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[4] John H. Reif and William L. Scherlis. Deriving efficient graph algorithms. *ACM Transactions on Programming Languages and Systems*, 1982. pending revision.

[5] Alexander Schrijver. *Theory of Linear and Interger programming.* Wiley Interscience publication, 1985.

[6] Gilbert Strang. *Linear Algebra and its applications,third edition.* HBJ Inc., 1988.

[7] Vaidya.P. A new algorithm for minimizing convex functions over convex sets. *Proceedings of the 30th IEEE FOCS*, 1989.

.