COS 524 Class Notes

Draft of March 2, 1992 Ken Steiglitz

x.1 Karmarkar's Algorithm [Ka]

The simplex algorithm stays on the boundary of the feasible set, and this fact has been blamed for its poor theoretical worst-case complexity. It seems that the ellipsoid algorithm is able to guarantee good asymptotic performance because it does not restrict its attention to boundary points, but can probe freely in the domain of its variables. It is essentially a feasibility-testing algorithm, however, and it solves the optimization problem LP only by solving a succession of related feasibility problems. At this point, the consensus is that the ellipsoid method, although it first showed that LP$\in$ P, and has some important theoretical consequences, is not competitive with simplex in practice.

In 1985, Karmarkar described another polynomial-time algorithm for linear programming, one which operates more directly within the feasible space of the original problem. The algorithm is interesting enough to study for its own sake, but is even more so because there are cases where it appears to be computationally faster than simplex. These cases rely for computational speed on the constraint matrix being sparse, and also require careful implementation.

Karmarkar's algorithm has spawned a number of variants, which are termed *interior-point* algorithms, because they share the property of moving strictly within the feasible set, rather than on its boundary. Most of these variants are closely related to classical iterative techniques for minimizing nonlinear functions subject to constraints. In this and the next section we describe the original version of Karmarkar's algorithm.

Karmarkar's algorithm works with the following form of linear programming problem, which, while apparently very special, can be shown to be equivalent to general LP (Problem x-1).

$$\min c'x \quad c, x \in R^n \tag{x.1}$$

$$x \in N_A \cap \Delta \tag{x.2}$$

where

$$N_A = \{x \mid Ax = 0\} \tag{x.3}$$

(the nullspace of A), and

$$\Delta = \{x \mid x \geq 0, e'x = 1\} \tag{x.4}$$

The vector e will be used throughout this section to denote the vector of 1's, $e = (1, 1, 1, \cdots, 1)'$, and thus the constraint $e'x = 1$ means

$$\sum_{1}^{n} x_i = 1 \qquad \text{(x.5)}$$

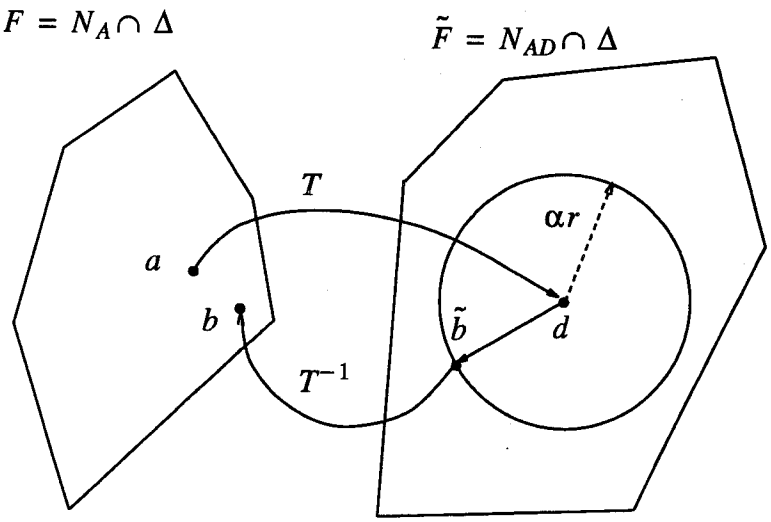

Fig. x-1 Karmarkar's algorithm. The initial feasible point a in F is mapped by T to the central point d in $\tilde{F}$. An appropriate cost function is then minimized over the sphere about d with radius αr, to produce the point $\tilde{b}$, and the transformation T^{-1} then carries $\tilde{b}$ back to the feasible point b, completing one iteration of the algorithm.

The set Δ is therefore the $(n-1)$-dimensional simplex defined by the intersection of this hyperplane with the nonnegative orthant. The feasible set in this formulation is the intersection of the nullspace of the operator A with Δ. From now on we use the notation $F = N_A \cap \Delta$ for this set.

We also make the following assumptions, which can also be shown to be non-restrictive (Problem x-2).

- The minimum value of the cost function is precisely 0; that is, $\min c'x = 0$.
- The point $d = e/n$ is feasible. It is used as an initial feasible point of the algorithm. (The letter d will be used throughout this section to represent this particular point.)

Finally, we assume we are given an integer q which serves to define the termination condition

$$\frac{c'x}{c'd} \leq 2^{-q} \tag{x.6}$$

As we know from previous discussion, when $q = \Theta(L)$ where L is the size of the input problem, this is sufficient to determine the solution to the linear program exactly.

The bare mathematical statement of Karmarkar's algorithm itself is simple, but the details involved in showing it works and is polynomial can be tedious. We begin with an intuitive explanation of the basic idea. (See Fig. x-1.) As in the ellipsoid algorithm, the liberating idea is to break free from the surface of the polytope of feasible points, and to operate somehow in its interior. In fact, the key step transforms the feasible space so that we can do an unconstrained optimization within a sphere — without fear of hitting a boundary. At each stage the point a in the feasible set F is mapped by a transformation T to the center d of a new set $\tilde{F}$. The transformation T is nonlinear, and depends on a. We next perform an unconstrained minimization within a sufficiently small sphere with respect to an appropriate cost function, to arrive at the new point OTa, and then reverse the transformation to get the new point $a = T^{-1}OTa$. (See Fig. x-2.) The success of the scheme depends on the careful choice of the transformation T and the cost function used in $\tilde{F}$.

We begin with a definition of the transformation T. Given a feasible point $a \in F$, $a > 0$, define the $n \times n$ diagonal matrix $D = diag(a_i)$. Then T is defined by

$$T(x) = \frac{D^{-1}x}{e'D^{-1}x} \tag{x.7}$$

Notice that if any component of a is 0, T is not well defined. This will not cause a problem, however, because throughout the algorithm we will be dealing with feasible points whose components are strictly positive. It is not hard to show the following properties of the mapping T (Problem x-3):

$$T(a) = \frac{e}{n} = d \tag{x.8}$$

$$T^{-1}(x) = \frac{Dx}{e'Dx}, \text{ for } x \in \Delta \tag{x.9}$$

$$T{:}\Delta \rightarrow \Delta, \text{ and is one-to-one and onto} \tag{x.10}$$

$$T \text{ maps vertices of } \Delta \text{ to vertices of } \Delta \tag{x.11}$$

$$T{:}\ F \rightarrow \tilde{F} = N_{AD} \cap \Delta \tag{x.12}$$

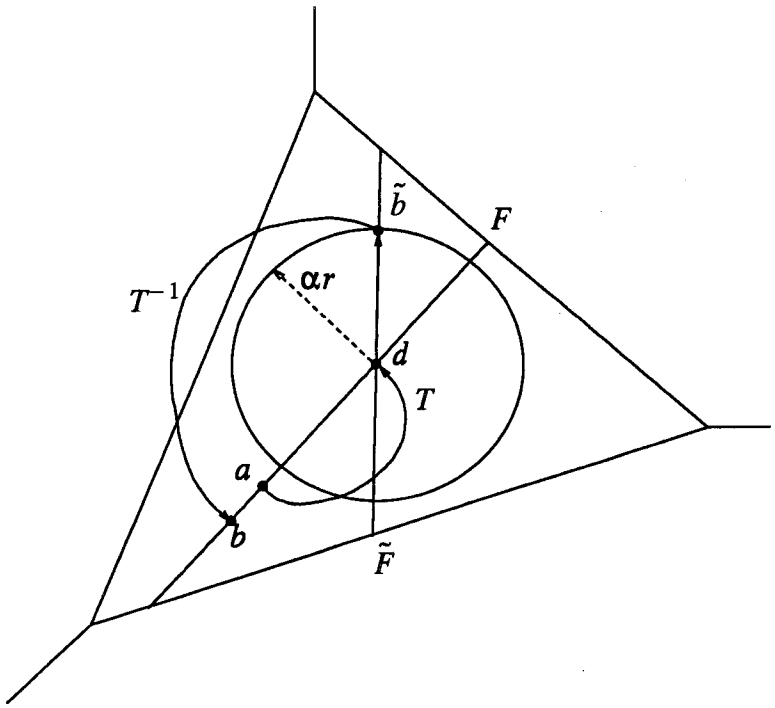

Fig. x-2 A more precise illustration of Karmarkar's algorithm The feasible set F and its transformed version $\tilde{F}$ both lie in the nonnegative slice of the hyperplane $e'x = 1$, and the central point d lies in both subsets.

We will fix the notation $\tilde{F}$ for the transformed version of F, which by (x.12) turns out to be $N_{AD} \cap \Delta$.

Next, we need to set the radius αr of the sphere centered at the point d so as to ensure that it stays in the nonnegative orthant. Figure x-3 shows that the radius r is the distance from the center d to a point of the form $(0, \frac{1}{n-1}, \frac{1}{n-1}, \ldots, \frac{1}{n-1})$, and therefore

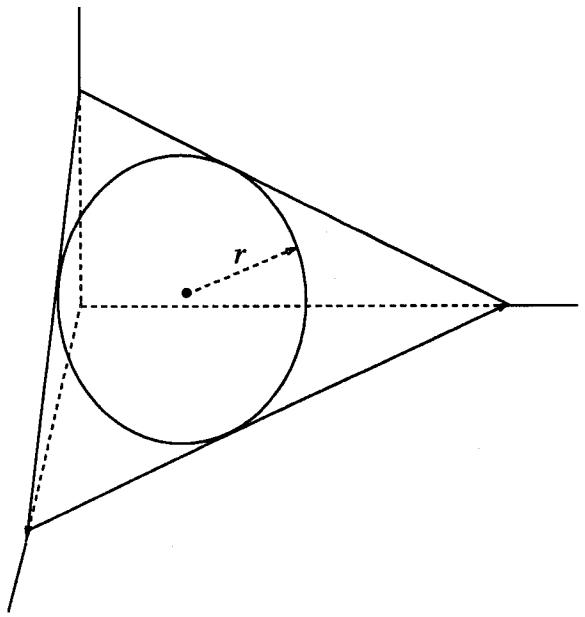

Fig. x-3 Calculation of the radius r.

$$r = \| (0, \frac{1}{n-1}, \frac{1}{n-1}, \ldots, \frac{1}{n-1}) \| - \| (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}) \| \quad \text{(x.13)}$$

$$= 1/\sqrt{n(n-1)}$$

The constant α is a safety factor, which allows for arithmetic roundoff error, and also allows for the fact that we will use a linearized version of the cost function in $\tilde{F}$, in place of the correct, but nonlinear cost function $c'T^{-1}x$. We will see later that the choice $\alpha = 1/4$ is adequate for our purposes.

The linearized cost vector we use in the transformed space $\tilde{F}$ is

$$\tilde{c} = Dc \quad \text{(x.14)}$$

and so, by (x.7), the cost we actually use can be written in terms of the original x as

$$\tilde{c}'\tilde{x} = \frac{c'x}{e'D^{-1}x} = \frac{c'x}{\sum_{j=1}^{n} (x_j/a_j)} \quad \text{(x.15)}$$

From this we see that $\tilde{c}'\tilde{x}$ can be considered an approximation to the scaled version $c'x/n$ of the original cost, provided x does not move too far away from our a in F.

Denote the sphere with center d and radius αr by $S(d, \alpha r)$, and define the $(n+1)\times n$ matrix

$$B = \left[\frac{AD}{e'}\right] \tag{x.16}$$

which is just the matrix AD over a row of 1's. The actual minimization is described in the next theorem.

Theorem x-1 (Karmarkar's Algorithm) The following algorithm minimizes $\tilde{c}'x$ over $\tilde{F} \cap S(d, \alpha r)$, $\alpha < 1$.

1. Compute

$$c_p = [I - B'(BB')^{-1}B]\tilde{c} \tag{x.17}$$

2. Normalize:

$$\hat{c_p} = c_p / \| c_p \| \tag{x.18}$$

3. Step a distance αr in the direction $-\hat{c_p}$:

$$\tilde{b} = d - \alpha r \hat{c_p} \tag{x.19}$$

Proof It is easy to verify first that the new point $\tilde{b}$ remains in $\tilde{F}$, because $B\hat{c_p} = 0$, and so

$$B\tilde{b} = \left[\frac{0}{1}\right] \tag{x.20}$$

which, together with the fact that $\tilde{b} \geq 0$, expresses the condition for feasibility in the transformed set $\tilde{F}$. In fact, c_p is defined the way it is to be in the nullspace of B, and the operation in (x.17) is an orthogonal projection into that space.

Now let z be any point in $\tilde{F} \cap S(d, \alpha r)$. We would like to show that its cost can be no smaller than that of $\tilde{b}$. From the fact that $B(\tilde{b} - z) = 0$ we can write

$$B'(BB')^{-1}B(\tilde{b}-z) = 0 \tag{x.21}$$

Equation (x.17) implies that

$$(\tilde{c} - c_p)' = \tilde{c}'B'(BB')^{-1}B \tag{x.22}$$

Combining (x.21) and (x.22) we get

$$(\tilde{c} - c_p)'(\tilde{b} - z) = 0 \tag{x.23}$$

and therefore

$$\tilde{c}'(\tilde{b} - z) = c_p'(\tilde{b} - z) \tag{x.24}$$

Next we examine

$$c_p'(\tilde{b} - z) = \| c_p \| \ \hat{c}_p'[d - \alpha r\hat{c}_p - z] \tag{x.25}$$

$$= \| c_p \| \ [\hat{c}_p'(d - z) - \alpha r] \tag{x.26}$$

Using the fact that $| \hat{c}_p'(d - z)| \leq \| d - z\| \leq \alpha r$ we conclude that

$$\tilde{c}'(\tilde{b} - z) = c_p'(\tilde{b} - z) \leq 0 \tag{x.27}$$

or

$$\tilde{c}'\tilde{b} \leq \tilde{c}'z \tag{x.28}$$

which is what we wanted to prove. □

The three steps of Theorem x-1 constitute Karmarkar's algorithm. We now need to prove it is theoretically efficient.

x.2 Complexity Analysis of Karmarkar's Algorithm

There now remains the task of showing that the $\tilde{b}$ produced in Theorem x-1 improves the cost enough at each iteration to make the entire algorithm work in polynomial time. Karmarkar's idea is to use an intermediate function, called a *potential function*, to measure the progress of the algorithm. This function has the property that it varies approximately as the logarithm of the true cost, but will penalize us if we approach too close to the boundary. In particular, for $x \in F$, we define the potential function by

$$f(x) = \sum_{j=1}^{n} \ln \frac{c'x}{x_j} = n\ln(c'x) - \sum_{j=1}^{n} \ln x_j \qquad \text{(x.29)}$$

Note that $f(x)$ is invariant with respect to a scale change: that is, $f(kx) = f(x)$ for any scalar multiplier k. For a point in the transformed space, $z \in \tilde{F}$, we define another potential function $\tilde{f}$ by f evaluated at the pre-image of z:

$$\tilde{f}(z) = f(T^{-1}(z)) = f(\frac{Dz}{e'Dz})$$

$$= f(Dz) \qquad \text{(x.30)}$$

(because $1/e'Dz$ is a scalar multiplier)

$$= \sum_{j=1}^{n} \ln \frac{\tilde{c}'z}{z_j} - \sum_{j=1}^{n} \ln a_j \qquad \text{(x.31)}$$

which we will need later. Thus, the function $\tilde{f}$ has the same form as f, except that $\tilde{c}$ replaces c, and there is an additive constant.

The basic idea of the rest of the convergence analysis of the algorithm is that each iteration results in at least a constant decrease in $\tilde{f}$. That is, we will show

$$\tilde{f}(\tilde{b}) \leq \tilde{f}(d) - \delta \qquad \text{(x.32)}$$

where δ is a constant, and $\tilde{b}$ and d are as used in Theorem x-1. The way $\tilde{f}$ is defined means that this implies

$$f(b) \leq f(a) - \delta \qquad \text{(x.33)}$$

and this in turn will imply convergence of the entire algorithm in polynomial time.

We begin by showing that there is at least one point in the sphere $S(d, \alpha r)$ in the transformed space that has the improved cost (x.32). After that, we will be able to show that the point found in Theorem x-1 also has such a good cost.

Theorem x-2 There is a point $z \in \tilde{F} \cap S(d, \alpha r)$ such that

$$\tilde{f}(z) \leq \tilde{f}(d) - \delta \qquad \text{(x.34)}$$

where $\delta = \ln(1 + \alpha)$.

Proof Let z^* be a point in $\tilde{F}$ where $\tilde{c}'x$ achieves its minimum value. We know that this occurs on the boundary of $\tilde{F}$, so $z^* \notin S(d, \alpha r)$. (See Fig. x-4.)

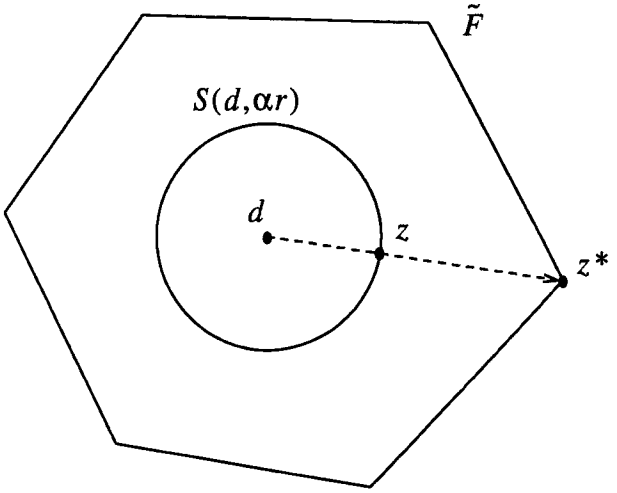

Fig. x-4 The point z lies at the intersection of the line between the center d and the point z^* in $\tilde{F}$.

The point we want to use is the point farthest in the direction from d to z^* but still within the sphere $S(d, \alpha r)$, namely

$$z = (1 - \lambda)d + \lambda z^* \qquad \text{(x.35)}$$

where λ is a positive number less than 1. We know that this point $z \in \tilde{F}$ because $\tilde{F}$ is convex. Taking costs, we get

$$\tilde{c}'z = (1 - \lambda)\tilde{c}'d + \lambda\tilde{c}'z^* \tag{x.36}$$

Now we know from (x.15) that the corresponding costs of $x \in F$ and of $\tilde{x} \in \tilde{F}$ differ by a positive factor, and that the minimum value of the cost in F is zero, so the last term here is zero, and we have

$$\frac{\tilde{c}'d}{\tilde{c}'z} = \frac{1}{1 - \lambda} \tag{x.37}$$

We next use (x.31) to write the improvement in the potential function

$$\tilde{f}(d) - \tilde{f}(z) = \sum_{j=1}^{n} \ln \frac{\tilde{c}'d}{\tilde{c}'z} \frac{z_j}{d_j} \tag{x.38}$$

and, using (x.35), (x.37), and $d_j = 1/n$,

$$= \sum_{j=1}^{n} \ln(1 + \frac{n\lambda}{1 - \lambda} z_j^*) \tag{x.39}$$

We next bound this from below by using the fact (Problem x-6) that for nonnegative A_i

$$\sum_i \ln(1 + A_i) \geq \ln(1 + \sum_i A_i) \tag{x.40}$$

so that we can write, using also the fact that $e'z^* = 1$,

$$\tilde{f}(d) - \tilde{f}(z) \geq \ln(1 + \frac{n\lambda}{1-\lambda}) \tag{x.41}$$

Finally, to put this in terms of known quantities, we need a lower bound for λ, which can

be derived by noting

$$\alpha r = \| z - d \| = \| \lambda(z^* - d) \| \leq \lambda R \qquad \text{(x.42)}$$

where $R/r = n - 1$ (Problem x-4), so that

$$1 + n\frac{\lambda}{1-\lambda} \geq 1 + \frac{n\alpha}{n-\alpha-1} \geq 1 + \alpha \qquad \text{(x.43)}$$

Combining this with (x.41) yields the result. □

We are now ready to prove the main result — that the point $\tilde{b}$ produced in Karmarkar's algorithm also provides a good decrease in the potential function.

Theorem x-3 If we choose $\alpha = 1/4$ and $n \geq 4$ in Karmarkar's algorithm, the point $\tilde{b}$ in (x.19) satisfies

$$\tilde{f}(\tilde{b}) \leq \tilde{f}(d) - \delta \qquad \text{(x.44)}$$

with the constant $\delta = 1/10$.

Proof Define the function

$$g(x) = n \ln \frac{\tilde{c}'x}{\tilde{c}'d}, \qquad \text{(x.45)}$$

let $\tilde{b}_0$ be a point in $\tilde{F} \cap S(d, \alpha r)$ where $\tilde{f}(x)$ achieves its minimum value, and write the identity

$$\begin{aligned} \tilde{f}(d) - \tilde{f}(\tilde{b}) = {} & [\tilde{f}(d) - \tilde{f}(\tilde{b}_0)] + [\tilde{f}(\tilde{b}_0) - (\tilde{f}(d) + g(\tilde{b}_0))] \\ & - [\tilde{f}(\tilde{b}) - (\tilde{f}(d) + g(\tilde{b}))] + [g(\tilde{b}_0) - g(\tilde{b})] \end{aligned} \qquad \text{(x.46)}$$

Our plan is to lower-bound each of the four bracketed terms. Using z from the previous theorem,

$$\tilde{f}(d) - \tilde{f}(\tilde{b}_0) \geq \tilde{f}(d) - \tilde{f}(z) \geq \ln(1 + \alpha) \tag{x.47}$$

which lower-bounds the first term. To lower-bound the second and third terms, let $w \in \tilde{F} \cap S(d, \alpha r)$, and use (x.31) to write

$$\tilde{f}(w) - [\tilde{f}(d) + g(w)] = -\sum_{j=1}^{n} \ln \frac{w_j}{d_j} \tag{x.48}$$

The inequality of Problem x-8 then yields

$$| \tilde{f}(w) - [\tilde{f}(d) + g(w)] | \leq \frac{\beta^2}{2(1 - \beta)} \tag{x.49}$$

where $\beta = \alpha\sqrt{n/(n-1)}$. To lower-bound the last term in (x.46) we observe that $g(x)$ is a monotonically increasing function of the cost $\tilde{c}'x$, and so it achieves its minimum value in $\tilde{F} \cap S(d, \alpha r)$ at the point $\tilde{b}$. Combining all these lower bounds yields finally

$$\tilde{f}(d) - \tilde{f}(\tilde{b}) \geq \ln(1 + \alpha) - \frac{\beta^2}{(1 - \beta)} \tag{x.50}$$

We leave it for Problem x-9 to show that when $\alpha = 1/4$ and $n \geq 4$ the right-hand-side is at least 1/10. □

The rest of the complexity analysis is easy. Let the result of Karmarkar's algorithm after i iterations be denoted by $b^{(i)}$, where $b^{(0)} = d$. Then

---

Theorem x-4 Karmarkar's algorithm solves the linear programming (x.1) and (x.2) with input length L in $O(nL)$ iterations and $O(n^4 L)$ arithmetic operations.

Proof We have already observed in (x.33) that the result of the previous theorem implies that in the original feasible set F

$$f(b^{(i)}) \leq f(b^{(i-1)}) - \delta \tag{x.51}$$

so after k iterations

$$f(b^{(k)}) \leq f(d) - k\delta \tag{x.52}$$

Replacing the function f by its definition and rearranging, this becomes

$$\begin{aligned} n \ln \frac{c'b^{(k)}}{c'd} &\leq \sum_{j=1}^{n} \ln b_j^{(k)} - \sum_{j=1}^{n} \ln d_j - k\delta \\ &\leq n \ln n - k\delta \end{aligned} \tag{x.53}$$

If we now choose $k = n(q + \ln n)/\delta$ iterations, where $q = \Theta(L)$, we are assured that

$$\frac{c'b^{(k)}}{c'd} \leq e^{-q} \leq 2^{-q} \tag{x.54}$$

which is sufficiently close to find an optimal vertex. The most time-consuming parts of the algorithm are the matrix inversion and multiplications in (x-17), and the time bound follows from the fact that we can perform those with $O(n^3)$ arithmetic operations. □

PROBLEMS

*x-1 Show that a standard form linear programming problem can be put in Karmarkar's form (x.1) and (x.2).

*x-2 Show that there is no loss of generality in assuming in Karmarkar's algorithm that the minimum cost is zero, and that the point $d = e/n$ is initially feasible.

x-3 Show properties Eqs. (x.8) to (x.12) of the transformation T.

x-4 Show that the feasible sets F and $\tilde{F}$ in Karmarkar's algorithm are contained in the sphere with radius $R = \sqrt{(n-1)/n}$, so that $R/r = n - 1$.

x-5 Investigate conditions which ensure that the matrix (BB') in the definition of the projected cost c_p, (x.17), is invertible.

x-6 Prove (x.40).

x-7 Prove that when $|x| \leq \beta < 1$

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

x-8 Use the result of the previous problem with $\beta = \alpha\sqrt{n/(n-1)}$ to show that when $w \in S(d, \alpha r)$ in Karmarkar's algorithm

$$\left| \sum_{j=1}^{n} \ln \frac{w_j}{d_j} \right| \leq \frac{\beta^2}{2(1 - \beta)}$$

x-9 Show that the right-hand-side of (x.50) is at least 1/10 when $\alpha = 1/4$ and $n \geq 4$.

x-10 When is $c_p = 0$ in Karmarkar's algorithm? Can this cause a problem?

x-11 [A. H. Watson] Show that F and $\tilde{F}$ in Fig. x-2 must connect the same two edges of the triangle bounding Δ.

NOTES AND REFERENCES

For putting a general linear programming problem in Karmarkar's form (Problems x-1 and x-2), see his original paper

[Ka] N. Karmarkar, "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica*, 4 (1984), 373-395.

He also discusses there a modification of the algorithm that takes $O(n^{3.5}L)$ instead of $O(n^4 L)$ arithmetic operations.

y.3 Centers and Another Algorithm

We have just described Karmarkar's algorithm in its original form. The iterative step takes us from the space of feasible points F to the transformed space $\tilde{F}$, where we take an unconstrained step, and then back from $\tilde{F}$ to F. The complexity analysis is somewhat complicated, but reveals to some extent why we convert the initial linear program to Karmarkar's special form. For example, we constantly made use of the fact that F was contained in Δ. As mentioned in the introduction, this algorithm has spawned many variants, some closely related to classical optimization methods. In this section we will describe an algorithm that is much easier to understand geometrically, and that works directly in the original problem space of a canonical-form linear program. The algorithm

makes use of the idea of the *center* of a polytope, which we discuss next.

The key to Karmarkar's algorithm is the transformation of our current feasible point to a point sufficiently far away from the constraint boundaries. Instead of transforming the space, we can, at each step, look for such a point in our original space. Let us assume now that our initial linear program is the canonical form

$$\max\ c'x$$

$$Ax \geq b$$

where $x \in R^n$ and A is an $m \times n$ matrix, $m > n$. We denote by P the feasible set $\{x \mid Ax \geq b\}$. If we penalize approach to a boundary by the logarithm of the distance to the constraint, we get the following definition of the *center* ω of the feasible set P:

$$\omega(P) = \max_{x \in P} \sum_{i=1}^{m} \ln(a_i'x - b_i)$$

The importance of this center has been appreciated for some time, and its relation to linear programming algorithms has been studies by Bayer and Legarius [BL], Renegar [Re], and Vaidya [Va1]. It is in a very natural sense a "balance point" of the polytope — the function maximized by the center is

$$G(x) = \sum_{i=1}^{m} \ln\,(a_i'x - b_i) = \ln \prod_{i=1}^{m} (a_i'x - b_i)$$

so the center maximizes the product of distances to the constraint walls. This function G is strictly concave (see Problem y-1) and assuming $\text{int}(P) \neq \emptyset$, it follows that the center is a uniquely defined point.

The problem of finding the center is quite different from solving a linear programming problem, if for no other reason than the fact that the hard boundaries have been eliminated, and we are searching for the maximum of a smoothly varying function. It turns out, in fact, that the center can be found in polynomial time by using the classical Newton's method, although the complexity analysis is lengthy [Va1], and will not be described here. What we will show is that this can lead to a polynomial-time algorithm for linear programming in a very natural way. Vaidya gives one method in [VA2]; the

particular method we describe next is quite clear geometrically. It is attributed to Renegar and was discovered, independently, by A. H. Watson [Wa].

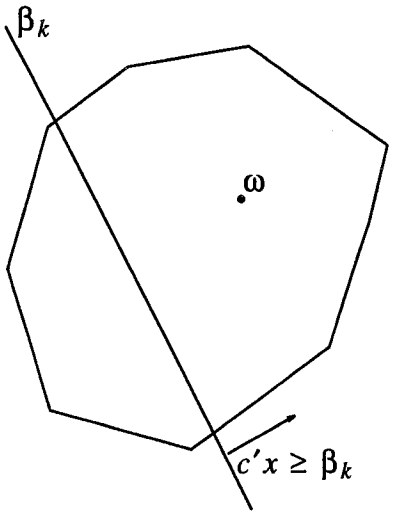

Fig. y-1 A simple linear programming algorithm based on center finding. At stage k the polytope is chopped by the constraint $c'x \geq \beta_k$, and a new center ω found.

The idea is to add the constraint $c'x \geq \beta$ to the other constraints, given that we know a solution of cost β. This chops a piece from the feasible set, and we can then find the center of the new polytope. The cost of this new center gives us a new value for β, and we can iterate. (See Fig. y-1.) As we continue this process, the successive centers produce a sequence of points that are interior to the original polytope, and whose cost converges to optimal. As in Karmarkar's algorithm, when we are within $2^{-O(L)}$ of optimal, we can stop, and if desired find an optimal vertex in polynomial time. We state the main result formally as

---

Theorem y-1 Given an algorithm for finding the center of a polytope in time $T(m, L)$, there is an algorithm for solving linear programming in time $O(m \cdot L \cdot T(m, L))$.

Proof Suppose after k iterations we have found the center ω of the polytope

$$P_k = \{x \mid Ax \geq b, \ c'x \geq \beta_k\}$$

and it has cost $c'\omega = \beta_{k+1}$. At the point ω the gradient of G is zero, so we can write

for all $x \in P_k$

$$\nabla G'(x - \omega) = \sum_{i=1}^{m} \frac{a_i'(x - \omega)}{a_i'\omega - b_i} + \frac{c'(x - \omega)}{c'\omega - \beta_k} = 0$$

Adding one to each term, we can re-write this as

$$\sum_{i=1}^{m} \frac{a_i'x - b_i}{a_i'\omega - b_i} + \frac{c'x - \beta_k}{c'\omega - \beta_k} = m + 1$$

Using the fact that the first summation is nonnegative in the polytope P_k, we get

$$c'\omega - \beta_k \geq \frac{1}{m+1}(c'x - \beta_k)$$

This holds for all $x \in P_k$, so we can replace the cost $c'x$ by the maximum cost in P_k, say $\beta_{\max}$. Letting $c'\omega = \beta_{k+1}$ we can write this as

$$\beta_{k+1} - \beta_k \geq \frac{1}{m+1}(\beta_{\max} - \beta_k)$$

Simple rearrangement of this yields

$$\beta_{\max} - \beta_{k+1} \leq \frac{m}{m+1}(\beta_{\max} - \beta_k)$$

We know that we can stop this iteration when $\beta_{\max} - \beta_0$ is reduced by a factor of $2^{O(L)}$, assuming we start with $\beta_0 = -2^{O(L)}$. Thus, we want after K iterations

$$\left[\frac{m}{m+1}\right]^K \leq 2^{-O(L)}$$

Choosing $K = O(m \cdot L)$ and observing that

$$\left(\frac{m}{m+1}\right)^m \leq \frac{1}{2}$$

completes the proof. □

PROBLEMS

y-1 Show that the function used in the definition of center,

$$G(x) = \sum_{i=1}^{m} \ln (a_i'x - b_i) = \ln \prod_{i=1}^{m} (a_i'x - b_i)$$

is strictly concave. (Hint: see Problem 1.13.)

Notes and References

Vaidya gives an algorithm based on Newton's method for finding the center of a polytope which takes $O((mn + n^3)\cdot L)$ steps in

[Va1] P. M. Vaidya, "A Locally Well-Behaved Potential Function and a Simple Newton-Type Method for Finding the Center of a Polytope," AT&T Bell Laboratories, 198?.

The chopping method is generally attributed to Renegar. Watson described the version described here in a talk:

A. H. Watson, "A Conceptually Simple Geometric Polynomial-Time Linear Programming Algorithm and Extensions," Research Seminar, Princeton University, April 27, 1988.

With Vaidya's method of finding centers Watson's chopping method solves linear programming in $O((m^2n + mn^3)\cdot L^2)$ steps, but Vaidya shows a way to use centers that takes $O(((m + n)n^2 + (m + n)^{1.5}n)\cdot L)$ steps:

[Va2] P. M. Vaidya, "An Algorithm for Linear Programming which Requires $O(((m + n)n^2 + (m + n)^{1.5}n)\cdot L)$ Arithmetic Operations," AT&T Bell Laboratories, 198?.

Other papers that study the center of a polytope are

[Re] J. Renegar, "A Polynomial-Time Algorithm, Based on Newton's Method, for Linear Programming," MSRI 07118-86, Math. Sciences Res. Inst., Berkeley,

CA.

and

[BL] D. A. Bayer and J. C. Legarius, "The Non-Linear Geometry of Linear Programming I, Affine and Projective Scaling Trajectories," AT&T Bell Laboratories, 1986.

The idea of cutting off a part of the feasible set by passing a plane through a central point is discussed for convex programming problems in

[NY] A. S. Nemirovsky and D. B. Yudin, "Problem Complexity and Method Efficiency in Optimization," John Wiley, New York, 1983. (Translated from the Russian: "Slozhnost' Zadach i Effektivnost' Metodov Optimizatsii," 1979.)

They discuss the particular method based on the center of gravity of the convex body, and attribute that idea to

A. Yu Levin, On an Algorithm for Minimizing Convex Functions," *Doklady Akad. Nauk SSSR*, vol. 160, no. 6, 1965. (English translation: *Soviet Maths.*, vol. 6, no. 1, pp. 286-290, American Mathematical Society.)