1 Notes on the Karmarkar algorithm

1.1 What's wrong with the simplex algorithm?

Consider the problem:

	$x1$	$x2$
$\max 2x1 + x2$	0	0
$5x1 + x2 \leq 5$	1	0
	0	5

It is easy to see that the optimal solution is $x1 = 0, x2 = 5$. However, the usual version of the simplex method uses $x1$ as an entering variable at the first step, since the objective coefficient is larger. This results in a wasted step, since we later have $x1 = 0$.

This single wasted step is not serious, but we can construct larger problems in which this phenomenon occurs again and again:

	$x1$	$x2$	$x3$
	0	0	0
$\max 4x1 + 2x2 + x3$	1	0	0
$5x1 + x2 \leq 5$	0	5	0
$10x1 + 5x2 + x3 \leq 30$	0	5	5
	1	0	20
	0	0	30

The table at the right shows the five steps needed by this problem. To see the pattern which develops in comparing this example to the first one, note that $x3$ in the second problem corresponds to $x2$ in the first one, and that $2x1 + x2$ corresponds to $x1$— the objective is $2(2x1 + x2) + x3$ and the last constraint is $5(2x1 + x2) + x3 \leq 30$.

The algorithm starts by making $2x1 + x2$ as large as possible, just as the first problem starts by making $x1$ as large as possible. Since the constraints on $x1, x2$ are the same as in the first problem, the first two steps are the same as in the first problem. Then the algorithm realizes that $x3$ should be made as large as possible, with $2x1 + x2$ going back to zero.

When we compare the second problem with the first, we see that the steps of the first problem are carried out, and then we have to do the same steps going backward [look at what happens to $x1, x2$ in the last three steps compared to first three]. This means that, although the second problem is only slightly larger than the first, the number of steps has more than doubled. This pattern can be continued. The problem below has $x1, x2, x3$ go through the same steps as in the second problem, then do those steps in reverse order (try it if you want— the table shows what happens).

$$
\begin{gathered}
\max\ 8x1 + 4x2 + 2x3 + x4 \\
5x1 + x2 \leq 5 \\
10x1 + 5x2 + x3 \leq 30 \\
20x1 + 10x2 + 5x3 + x4 \leq 155 \\
[30 = 5(5) + 5] \\
[155 = 5(30) + 5]
\end{gathered}
$$

$x1$	$x2$	$x3$	$x4$
0	0	0	0
1	0	0	0
0	5	0	0
0	5	5	0
1	0	20	0
0	0	30	0
0	0	30	5
1	0	20	35
0	5	5	80
0	5	0	105
1	0	0	135
0	0	0	155

If we continue this sequence, we obtain examples with n variables and $n-1$ constraints which take $.75(2^n) - 1$ steps.

[This specific sequence of examples depends in an essential way on the use of the "reduced cost" rule for choosing entering variables. However, (1) similar examples have been constructed for some other rules; (2) there is no rule for which it has been proved that bad behavior cannot occur.]

1.2 What the Karmarkar Algorithm Does

The algorithm works with a linear program

$$
\begin{aligned}
&\min cx \\
&Ax = \vec{0} \\
&\textstyle\sum x_i = 1 \\
&x \geq \vec{0}
\end{aligned}
\tag{1}
$$

Let z^* be the objective function value of the optimal solution.[1] Suppose we are given a feasible solution $x = a$ to (1) with $a_i > 0$ for all i, and $z < z^*$. The algorithm gives a feasible a' (all $a'_i > 0$) and z' with $z \leq z' \leq z^*$ such that

$$\frac{(ca' - z')^n}{\prod a'_i} \leq K \frac{(ca - z)^n}{\prod a_i} \tag{2}$$

where $K \approx 2e^{-1} \approx .736$.

Let a be a starting solution and J be the objective function value after m iterations. The arithmetic-geometric mean inequality implies any feasible solution to this LP must satisfy $\prod x_i \leq n^{-n}$. Using this and (2) gives

$$\frac{(J - z^*)^n}{n^{-n}} \leq \frac{(J - z')^n}{n^{-n}} \leq K^m \frac{(ca - z)^n}{\prod a_i}$$

which implies $J - z^* \leq K^{m/n}$ times a constant. This guarantee that the distance from the current objective function value to the optimal value decreases exponentially is what makes the Karmarkar algorithm important.

1.3 Sphere Problems

We will need to minimize a linear objective on the intersection of a sphere and some hyperplanes:

$$\begin{array}{l} \min dw \\ Bw = \vec{0} \quad \sum w_i = 1 \\ ||w - (1/n, \ldots, 1/n)|| \leq s \end{array} \tag{3}$$

This problem is "easy" to solve by projecting d onto an appropriate subspace.[2]

The cases we will be interested in are $s = \alpha(n(n-1))^{-1/2}$ [henceforth denoted by αr], for $0 < \alpha < 1$, and $s = R = (n-1)^{1/2} n^{-1/2}$. Note that all feasible solutions to (3) with $s = \alpha r$ have positive components, and that $Bw = \vec{0}$, $\sum w_i = 1$, and $w \geq \vec{0}$ implies w is a feasible solution to (3) with $s = R$.

[1]Early papers assumed z^* was known, but we will use a technique from [1] to avoid this

[2]In implementations, special tricks are used to get approximate solutions more quickly

We will always be concerned with a sphere problem determined by a feasible solution $x = a$ to (1), with $d_i = (c_i - M)a_i$ for some $M > 0$ and B such that

$$Bw = A \begin{pmatrix} a_1 w_1 \\ \dots \\ a_n w_n \end{pmatrix}$$

[in other words, B is A multiplied by a diagonal matrix]

For given a, M, s we will denote the optimal solution to (3) by $w(a, M, s)$ and the objective function value by $V(a, M, s)$.

1.4 The Iterative Step

Given a and z, find $V(a, z, R)$. If $V(a, z, R) \leq 0$, let $z' = z$ and

$$a' = \gamma(a_1 w_1, \dots, a_n w_n) \tag{4}$$

for $w = w(a, z', \alpha r)$ and γ chosen so that $\sum a'_i = 1$. If $V(a, z, R) > 0$, find $z' > z$ with $V(a, z', R) = 0$ and get a' from (4) as above.

Since $Aa' = \gamma Bw = \vec{0}$, a' is a feasible solution to (1). If $z' > z$ and x^* is the optimal solution to (1), $V(a, z', R) = 0$ implies that for some $\delta > 0$

$$\sum (c_i - z')a_i(\delta x_i^*/a_i) = \delta(z^* - z' \sum x_i^*) \geq 0$$

Since $\sum x_i^* = 1$, $z' \leq z^*$.

Since $(1/n, \dots, 1/n)$, $w(a, z', \alpha r)$, and $w(a, z', R)$ are on the same line,

$$w(a, z', \alpha r) = (1 - \alpha/n - 1)(1/n, \dots, 1/n) + (\alpha/n - 1)w(a, z', R)$$

$V(a, z', R) = dw(a, z', R) \leq 0$ and $\sum a_i = 1$ imply

$$dw(a, z', \alpha r) \leq (1 - \alpha/n - 1)d(1/n, \dots, 1/n) = 1/n(1 - \alpha/n - 1)\left(\sum c_i a_i - z'\right)$$

Since $\sum a'_i = 1$

$$ca' - z' = \sum (c_i - z')a'_i = \gamma \sum (c_i - z')a_i w_i = \gamma dw(a, z', \alpha r)$$

Combining these gives

$$ca' - z' \leq \gamma n^{-1}(1 - \alpha/n - 1)(ca - z') \tag{5}$$

This gives us a relationship between the numerators in (2). To get a lower bound on $\prod a'_i$, we use the fact (proven in [2]) that, for any w, if $\sum w_i = 1$ and $\sum (w_i - 1/n)^2 = \alpha^2 r^2$, then

$$\prod w_i \geq n^{-n}(1 + \alpha/n - 1)^{n-1}(1 - \alpha) \tag{6}$$

When we use (6), (4), and (5) in (2) the γ^n and n^{-n} terms cancel and we get

$$K = (1 - \alpha/n - 1)^n (1 + \alpha/n - 1)^{1-n}(1 - \alpha)^{-1}$$

which is close to $2e^{-1}$ for $\alpha = (n-1)/(2n-3)$ and n [not too] large.

1.4.1 Updating z

We are concerned with the behavior of $V(a, M, R)$ as a function of M. Let $g_i = c_i a_i$. There are vectors e, f [related to projections of g and a] such that $w(a, M, R) = R(e + Mf)/||e + Mf||$.

$$\frac{R}{||e + Mf||}(g - Ma) \cdot (e + Mf) = 0$$

can be solved for M as a quadratic equation.

References

[1] K. Anstreicher. *Analysis of a Modified Karmarkar Algorithm for Linear Programming.* Yale School of Organization and Management Working Paper #84 (series B). August 1, 1985.

[2] C. Blair. The Iterative Step in the Linear Programming Algorithm of N. Karmarkar. *Algorithmica* 1 (1986), 537–539.

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