A VECTOR-JUMP HEURISTIC FOR KARMARKAR'S LINEAR PROGRAMMING ALGORITHM

Teresa Baile DeWitt and L. Vincent Edmondson

1. Introduction. A significant amount of research has been done since 1984 in the area of interior-point algorithms for linear programming (LP). Some of these algorithms have rivaled the simplex method for LP. This paper presents a variant of Narendra Karmarkar's interior-point algorithm [1] for solving LP problems. The goal was to improve the efficiency of the algorithm by keeping track of the direction vectors in successive iterations and then heuristically exploiting this information to jump beyond the current iterate. A detailed description of this heuristic and experimental results are included.

2. Linear Programming. The general linear programming problem can be expressed by the following model.

Minimize

$$z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

subject to the constraints

$$\begin{gathered}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\
\vdots \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_n
\end{gathered}$$

and

$$x_1 \geq 0, \quad x_2 \geq 0, \quad \cdots, x_n \geq 0.$$

The variables $x_1, x_2, \cdots x_n$ are the decision variables while a_{ij}, b_i, and c_j are the parameters of the model. The function z is the objective function which is minimized or maximized depending on the problem.

In Karmarkar's algorithm, a restricted form of the above problem is required and is expressed in matrix notation.

$$\text{Minimize } \vec{c}^T \vec{x}$$

$$\begin{aligned} \text{subject to } A\vec{x} &= \vec{0} \\ \vec{e}^T \vec{x} &= 1. \end{aligned}$$

The matrix and vectors are defined as:

$\vec{c}$ is the vector consisting of the coefficients of the objective function.
A is the matrix consisting of the coefficients of the functional constraints.
$\vec{e}$ is a vector of n 1's.

Another matrix that is used in the algorithm is D. D is a diagonal matrix with the current $\vec{x}$ iterate forming the diagonal. Karmarkar's restricted form requires that the minimum value of the objective function be zero.

Karmarkar's method for solving LP problems starts at the center point of the feasible region. After a transformation step, the objective function is projected onto the null space of the matrix of constraints, yielding a direction and steplength. After this step has been taken, the point is transformed back onto the original simplex and becomes the new $\vec{x}$ iterate. A feasibility test is applied and the objective function value is computed and tested for optimality. This process moves the solution towards the boundary where the optimal solution must be. The iterative portion of the algorithm stops when the value of the objective function is acceptably close to zero. The final step is to move the current $\vec{x}$ iterate to the nearest vertex of the feasible region.

The technical description of Karmarkar's algorithm follows:

Step 0: Let $\vec{x}^{(k)} = (1/n, 1/n, \ldots, 1/n)$ and $k = 0$. This sets the first iterate equal to the center of the simplex.

Step 1: Define a matrix B as

$$B = \begin{pmatrix} AD \\ \vec{e}^T \end{pmatrix},$$

which is a matrix of functional constraints.

Step 2: Orthogonally project the objective vector $D\vec{c}$ onto the null space of B, yielding $\vec{c}_p$,

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

Step 3: Define r to be the radius of the largest sphere you can inscribe in the unit simplex:

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

Normalize $\vec{c}_p$ and obtain the scaled direction vector $\vec{p}$:

$$\vec{p} = \sqrt{\frac{\vec{c}_p}{|\vec{c}_p|}}.$$

Step 4: Take a steplength $\alpha(0 < \alpha < 1)$ from the center of the simplex in the $-\vec{p}$ direction and that will give the new $\vec{x}'$:

$$\vec{x}' = \vec{e}/n - \alpha r\vec{p} \quad \text{where} \quad \vec{e}/n = \vec{a}_0.$$

(Since we are minimizing the function we go in the $-\vec{p}$ direction. Karmarkar suggests $\alpha = 0.25$).

Step 5: Do the inverse transformation to get back to the $\vec{x}$ iterate

$$\vec{x}^{(k+1)} = \frac{D\vec{x}'}{\vec{e}^T D\vec{x}'}.$$

This gets us back to the original simplex.

Step 6: Test for infeasibility using the following potential function,

$$f(\vec{x}) = \sum_{i=1}^{n} \ln\left(\frac{\vec{c}^T\vec{x}}{x_i}\right).$$

If $f(\vec{x}^{(k)}) - f(\vec{x}^{(k+1)}) < \delta$, then stop because the problem is infeasible or unbounded. (δ is a function of α such that $\delta = \alpha - \frac{\alpha^2}{2} - \frac{\alpha^2}{1-\alpha}$).

Step 7: Check for optimality. If $\vec{c}^T\vec{x}^{(k+1)} < \epsilon$ then we purify $\vec{x}$ and stop. If $\vec{c}^T\vec{x}^{(k+1)} \geq \epsilon$ then $k = k+1$ and go back to Step 1. (To purify $\vec{x}$ means to move to the nearest boundary point that does not worsen the objective function.)

3. Scalar Multiplier Modification.

Step 2 of Karmarkar's algorithm, the projective transformation step, is the most complicated and time consuming step of the algorithm. Consequently, this is where attention was focused when researching a modification to Karmarkar's method. The modification made is to choose a scalar multiplier that would be applied in Step 5 after m iterations. This scalar would be greater than 1 and would effectively increase the current $\vec{x}$ iterate an increment beyond where it was. Then the algorithm would proceed as normal until m iterations had occurred again and the modification would again be invoked.

It is interesting to note how the scalar multiplier uses the history of past iterations. The current $\vec{x}$ iterate can be considered as the sum of all the vectors used to obtain the previous $\vec{x}$ iterates. The multiplier in effect takes what is a developing directional pattern and moves the current $\vec{x}$ iterate a fraction beyond where it would have been. This additional step is in the direction dictated by the vectors that were found in the earlier iterations.

One consequence of the scalar multiplier modification is the possibility of stepping outside the boundary of the feasible region. To account for that possibility Karmarkar's restricted form is exploited. The restricted form forces the minimum objective function value to be zero. If the scalar multiplier modification results in a negative value for the objective function, then the scalar multiplier is permanently set back to 1.0, thereby returning to Karmarkar's original algorithm.

The modification to Karmarkar's method changes the algorithm in the following manner:

Step 0 sets the first $\vec{x}$ iterate equal to the center of the simplex.

Step 1 defines the matrix B.

Step 2 orthogonally projects the vector onto the null space of B.

Step 3 defines the radius of the largest sphere that can be inscribed in the simplex. The scaled direction vector is calculated in this step also.

Step 4 is where the step is taken from the center of the null space and the new $\vec{x}'$ vector is found.

Step 5 transforms the $\vec{x}'$ vector back to the original space to obtain the new $\vec{x}$ iterate. This is where the scalar multiplier is applied creating a different $\vec{x}$ iterate.

Step 6 tests for infeasibility.

Step 7 tests for optimality.

4. Experimental Results. There were five sample problems used to test the scalar multiplier modification. They are:

Problem A: Minimize $2x_1 + x_2 - 2x_3$
subject to $x_1 - x_3 = 0$.

Problem B: Minimize $x_1 + x_2 - x_3$
subject to $x_2 - x_3 = 0$.

Problem C: Minimize $3x_1 + 3x_2 - x_3$
subject to $2x_1 - 3x_2 + x_3 = 0$.

Problem D: Minimize $x_1 + 2x_2 - x_3$
subject to $x_1 - x_3 = 0$.

Problem E: Minimize $x_1 - x_2 + 6x_3$
subject to $x_1 - x_2 = 0$.

The above problems all had the additional constraint: $x_1 + x_2 + x_3 = 1$.

To illustrate the effects of the Scalar Multiplier method, the following table shows the results of the first five iterates and also the tenth iterate for Problem C using Karmarkar's original algorithm and the modified algorithm with the scalar $= 1.01$ and $m = 2$.

Iteration	Karmarkar	Scalar Multiplier
1	$x_1 = 0.270339$ $x_2 = 0.317585$ $x_3 = 0.412076$ obj val = 1.351696	$x_1 = 0.270339$ $x_2 = 0.317585$ $x_3 = 0.412076$ obj val = 1.351696
2	$x_1 = 0.208148$ $x_2 = 0.302037$ $x_3 = 0.489815$ obj val = 1.040740	$x_1 = 0.207526$ $x_2 = 0.301881$ $x_3 = 0.490593$ obj val= 1.037630
3	$x_1 = 0.152697$ $x_2 = 0.288174$ $x_3 = 0.559129$ obj val = 0.763483	$x_1 = 0.150989$ $x_2 = 0.287747$ $x_3 = 0.561264$ obj val = 0.754945
4	$x_1 = 0.107860$ $x_2 = 0.276965$ $x_3 = 0.615175$ obj val = 0.539300	$x_1 = 0.104903$ $x_2 = 0.276226$ $x_3 = 0.618871$ obj val = 0.524517
5	$x_1 = 0.074308$ $x_2 = 0.268577$ $x_3 = 0.657115$ obj val = 0.371540	$x_1 = 0.070186$ $x_2 = 0.267546$ $x_3 = 0.662268$ obj val = 0.350929
10	$x_1 = 0.010184$ $x_2 = 0.252546$ $x_3 = 0.737271$ obj val = 0.050918	$x_1 = 0.002891$ $x_2 = 0.250723$ $x_3 = 0.746386$ obj val = 0.014456

Initially, all of the test problems were solved using Karmarkar's original algorithm. The stopping criterion was a value less than 0.0000001 for the objective function. This gave the number of iterations Karmarkar's algorithm would take to solve the problems.

The problems were then solved using the Scalar Multiplier modification. Based on the number of iterations Karmarkar's algorithm used, m, the number of iterations between the implementations of the scalar multiplier, was chosen to be 2, 4, or 8. The scalar multiplier values tried were 1.001, 1.01, 1.10, 1.25, and 1.50. The results of the program runs are exhibited in Tables A, B, and C.

All the tables display the number of iterations needed to reach the solution at each scalar multiplier and for Karmarkar's algorithm. They also display the number of iterations, k, that occurred before the objective function value became negative and the scalar multiplier defaulted back to the value of 1. Table A is for $m = 2$, Table B is for $m = 4$, and Table C is for $m = 8$.

TABLE A — For $m = 2$

Scalar Multiplier (SM)	SM=1 after k iterations	Problem A	Problem B	Problem C	Problem D	Problem E
1.001	$k = 17$	**27**	**27**	35	**27**	**27**
1.01	$k = 11$	32	33	37	33	33
1.1	$k = 6$	32	34	35	34	34
1.25	$k = 5$	31	31	**34**	31	31
1.50	$k = 4$	33	33	35	33	33
Karmarkar's Original		34	36	38	37	37

TABLE B — For $m = 4$

Scalar Multiplier (SM)	SM=1 after k iterations	Problem A	Problem B	Problem C	Problem D	Problem E
1.001	$k = 19$	34	35	36	35	35
1.01	$k = 13$	34	35	37	35	35
1.10	$k = 10$	**26**	**26**	**32**	**30**	**26**
1.25	$k = 7$	35	35	36	35	35
1.50	$k = 7$	30	30	34	31	31
Karmarkar's Original		34	36	38	37	37

TABLE C — For $m = 8$

Scalar Multiplier (SM)	SM=1 after k iterations	Problem A	Problem B	Problem C	Problem D	Problem E
1.001	$k = 22$	35	**36**	**31**	**36**	**36**
1.01	$k = 15$	35	**36**	37	**36**	**36**
1.1	$k = 8$	35	37	37	37	37
1.25	$k = 8$	35	37	37	37	37
1.50	$k = 8$	35	37	37	37	37
Karmarkar's Original		34	36	38	37	37

The results obtained from running the modified program are a fairly strong indicator that the Scalar Multiplier modification has promise as an effective variant of Karmarkar's interior-point algorithm. With the exception of Problems A and B at $m = 8$ and one incidence of Problem A at $m = 4$, all other problems at every different combination of m and SM showed at the worst case an equal number of iterations as Karmarkar and in most cases, an improvement over Karmarkar. The improved cases ranged from a 3% improvement to a 30% improvement with 11 of the 15 possible test scenarios showing a double digit percentage of improvement. Also, the exceptions to equaling or improving Karmarkar are all just one iteration more in every case. Therefore, any decrease in efficiency was a very small decrease.

There are other results of the research that stand out as being significant. First of all, the patterns of improvement are very interesting. In Table A for instance, four out of five most improved cases occur when $SM = 1.001$. In Table B, an even stronger pattern is evident with all five most improved cases occurring when $SM = 1.10$. Even in Table C, which exhibits the weakest evidence for the modification, four of the five most improved cases occur when $SM = 1.001$.

Delving deeper into the patterns, also note that when $m = 2$, (the smallest value of m), the smallest value of SM gave the best results. When $m = 4$, the intermediate value for m, $SM = 1.10$, an intermediate value for SM also gave the best results. It appears that the Scalar Multiplier method is most effective when used often ($m = 2$) and with a scalar multiplier close to 1.0. Given that Karmarkar's original algorithm does not guarantee a decrease in the objective function at every iteration, it does appear reasonable that a smaller scalar multiplier would be best. Whenever the objective function increases, Karmarkar's algorithm effectively "backs up" towards the center of the feasible region before proceeding. A larger value for the scalar multiplier would likely increase the probability of this occurrence. Consequently, smaller steps taken more often appear to be a good starting point for setting the parameters of the modified algorithm.

5. Future Research. The research presented here indicates that the Scalar Multiplier modification can affect improvement over Karmarkar's interior-point algorithm. The results lead to open questions for future research. First, will this method scale up and be as effective on larger problems? Second, would the method work better if, instead of resetting the scalar multiplier to 1.0 whenever the objective function value becomes negative, the method is employed throughout the entire problem? Finally, would dynamically changing the scalar multiplier and/or m (perhaps as a function of the number of iterations) yield better results than leaving them as static values?

Reference

1. N. Karmarkar, "A New Polynomial-time Algorithm for Linear Programming," *Combinatorica* 4 (1984), 373–395.

Teresa Baile DeWitt
Versailles High School
309 S. Monroe
Versailles, MO 65084

L. Vincent Edmondson
Department of Mathematics and Computer Science
Central Missouri State University
Warrensburg, MO 64093-5045
email: vincee@cmsuvmb.cmsu.edu