CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision

(Université de Paris-Dauphine)

Unité Associée au C.N.R.S. n° 825

NEW SUGGESTED IMPLEMENTATIONS OF KARMARKAR's ALGORITHM

CAHIER N° 71

mai 1986

M. MINOUX (*)

(*) Directeur de Recherche, LAMSADE, Université de Paris-Dauphine et Ministère de la Défense.

CONTENTS

DE NOUVELLES IMPLEMENTATIONS DE L'ALGORITHME DE KARMARKAR

RESUME

Des travaux récents ont montré que l'algorithme de Karmarkar peut être en fait mis en oeuvre de façon à réduire significativement le nombre d'itérations. Cependant, l'effort de calcul par itération (qui implique la projection d'un vecteur sur le sous-espace nul d'une matrice ayant les dimensions de la matrice des contraintes) apparaît alors comme la principale source d'inefficacité dans le calcul.

Ce papier se veut être un pas vers des mises en oeuvre pratiques plus efficaces de l'algorithme de Karmarkar. On montre d'abord que le sous-problème de recherche de la direction de déplacement peut se formuler comme un problème d'optimisation quadratique sans contrainte dans un sous-espace de dimension généralement plus petite que celle du problème original. On montre ensuite comment l'application de diverses procédures de type "gradient conjugué" permet de résoudre ce problème soit de façon exacte, soit de façon approchée, et conduit ainsi à toute une famille d'implémentations où existe une grande souplesse de choix concernant les compromis précision-effort de calcul. Indépendamment de sa souplesse et de sa facilité de mise en oeuvre, cette nouvelle approche permet de tirer avantage, au mieux, de la faible densité des matrices de contraintes.

NEW SUGGESTED IMPLEMENTATIONS OF KARMARKAR's ALGORITHM

ABSTRACT

Recent work has shown that Karmarkar's algorithm can indeed be implemented in such a way as to get significant reduction in the number of iterations. However, the computational effort per iteration (projecting a vector onto the null space of a matrix of the same size as the constraint matrix) then appears to be the main source of computational inefficiency.

This paper is intended as a step towards more practical and efficient implementations of Karmarkar's method. First it is shown that the direction-finding subproblem can be reformulated as an unconstrained quadratic minimization problem on a subspace of dimension usually significantly smaller than the dimension of the original problem. Then it is shown how various conjugate gradient procedures can be applied to solve this problem, either exactly or approximately, leading to a whole family of implementations with high flexibility regarding the choice of the best tradeoffs between accuracy and computation effort in the solution of the direction-finding subproblems. Besides its flexibility and ease of implementation, the new approach takes full advantage of the sparsity of the original constraint matrix.

1. INTRODUCTION

The new polynomial algorithm due to N. KARMARKAR (1984) for solving linear programs is certainly one of the most significant progress recently achieved in the Mathematical Programming area. However some controversy soon arose around the author's claims of the new method being "50 to 100 times faster than the simplex method". In fact, various computational experiments carried out independently by a number of researchers (see e.g. TOMLIN (1985), LUSTIG (1985), NICKELS, RODDER, XU and ZIMMERMANN (1985)) soon led to the conclusion that the asserted superiority of the new algorithm might not be that overwhelming. Indeed, if it is now recognized that KARMARKAR's methods can be implemented in such a way as to produce a very low number of iterations on the average (see e.g. NICKELS & alii, LISSER, MACULAN & MINOUX (1985)) the computational effort per iteration involves the calculation of the pseudoinverse of a matrix with roughly the same size as the constraint matrix, and this may be prohibitive.

In this paper, we suggest possible ways of improving the practical efficiency and ease of implementation of Karmarkar's method by showing that :

- the projection subproblem to be solved at each iteration can be replaced by the problem of unconstrained minimization of a convex quadratic function defined on a subspace of reduced dimension ;
- that this quadratic problem can be solved either exactly or, more interestingly, approximately via a conjugate gradient method ;
- that this conjugate gradient approach is well suited for exploiting the sparsity (if any) of the original constraint matrix.

2. KARMARKAR's ALGORITHM IN BRIEF

For simplicity we will consider here the method as presented originally in Karmarkar (1984). Of course, the improved implementation suggested here could be extended in a rather obvious way to most of the variants eventually

described by a number of authors (see e.g. MEGGIDO (1985), LUSTIG (1985), GAY (1985), TODD & BURRELL (1985)). This is, in particular, the case of the variant which consists in performing an exact or approximate one-dimensional minimization of the potential function at each iteration as suggested in TODD & BURRELL (1985) and, independently, in LISSER, MACULAN & MINOUX (1985).

Let the problem be stated in the following "canonical" form :

$$(P)\quad \begin{cases} \text{Minimize} \quad c^T x \\ \text{subject to :} \\ A.x = 0 \\ \sum_{i=1}^{n} x_i = 1 \\ x \geq 0 \end{cases}$$

where $n > m$ and $x \in \mathbb{R}^n$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$.

As in KARMARKAR (1985) we make the assumption that the initial point

$$x^0 = \frac{1}{n}(1, 1, \ldots)^T = \frac{1}{n}.e$$

is a feasible solution to (P) (here e denotes the n-vector with all components equal to 1).

This assumption is justified by the fact that it is always possible to convert any linear program given in standard form and for which a feasible solution is known, into the form (P) with $\frac{1}{n}\, e$ as a feasible solution (see section 5 of KARMARKAR's paper).

The constraint set of (P) is the intersection of the null space of A (a linear variety) and the simplex $S = \{x \,/\, x \geq 0, \sum_{i=1}^{n} x_i = 1\}$. Note also that the starting point $x^0 = \frac{1}{n}\, e$ is nothing but that the center of the simplex S.

The algorithm generates a sequence of iterates $x^0, x^1, \ldots, x^k$ according to the following process :

Current iteration of Karmarkar's algorithm

(a) Let $x^k = (d_1, d_2, \ldots, d_n)^T$ be the current point. Set $D = \text{diag}(d_1, \ldots, d_n)$, the diagonal matrix with d_i as the i^{th} diagonal entry.

Let $B = \begin{bmatrix} AD \\ \hline e^T \end{bmatrix}$ be the $(m + 1) \times n$ matrix formed by matrix AD augmented with a row of all 1's.

(b) Compute $\hat{c}$, the orthogonal projection of the vector Dc onto the null space of B, i.e. :

$$\hat{c} = [I - B^T(B\,B^T)^{-1}\,B]\,Dc.$$

Let $u = \frac{\hat{c}}{\|\hat{c}\|}$ be the unit vector in the direction of $\hat{c}$.

(c) Let $b' = \frac{1}{n}.e - \theta\,u$ be the point obtained by moving from the center of the simplex S $(\frac{1}{n}\,e)$ a step of length θ in the direction $-u$ (the main restriction on θ is that all components of b' should be > 0).

(d) The next iterate x^{k+1} is obtained by applying to b' the (projective) transformation

$$x^{k+1} = b = \frac{D\,b'}{e^T\,D\,b'}.$$

Concerning the choice of step lengths θ, Karmarkar's original recommendation is to take

$$\theta = \alpha.r$$

where $r = \frac{1}{\sqrt{n(n-1)}}$ is the radius of the largest sphere inscribed in the simplex S and $\alpha = \frac{1}{4}$. As shown in Karmarkar (1984) this choice is sufficient to ensure polynomiality of the algorithm. However it leads to a

number of iterations growing as a linear function of the number n of variables. Experimental results reported by NICKELS, RODDER, XU and ZIMMERMANN (1985) and LISSER, MACULAN & MINOUX (1985) show that allowing much longer step sizes (with the restriction of staying in the interior of the simplex S) may lead to a dramatic reduction in the overall number of iterations. Since the cost of evaluating the potential function value at specific points is much lower than the cost of computing the search direction u, the best strategy at present, seems to be to perform an approximate minimization of the potential function along $-u$.

As shown in LISSER, MACULAN and MINOUX, only a rough approximation of the optimal step in the direction is sufficient for achieving very good convergence properties (4 to 6 function evaluation at each step). The experimental number of iterations of Karmarkar's algorithm then seems to grow roughly as a logarithmic function of the number of variables. Moreover it is easily realized that, if the maximum number of potential function evaluations allowed in the approximate one dimensional optimization is bounded by a fixed constant, then the polynomiality of the algorithm is preserved (this is because (i) computing the potential function at some point is polynomial ; (ii) the actual reduction in the potential function resulting from the approximate one-dimensional search is at least equal to the reduction achieved by Karmarkar's step).

We now turn to discuss possible ways of reducing the computational effort for getting the search direction u at step (b) of the algorithm.

3. SOLVING THE DIRECTION FINDING PROBLEM BY A CONJUGATE GRADIENT ALGORITHM IN A SUBSPACE OF REDUCED DIMENSIONS

Looking for the projection of some vector $y = D\,c \in \mathbb{R}^n$ onto the null space of B (a $(m+1) \times n$ matrix) can be stated as the least squares problem of minimizing the L^2 norm :

$$(I)\begin{cases} \|x - y\| \\ \text{under the constraints :} \\ B\,x = 0 \end{cases}$$

Some proposals following this idea of treating the direction-finding subproblem in Karmarkar's algorithm as a least squares problem have already been made, e.g. in LUSTIG (1985) (where efficient algorithms developped in PAIGE & SAUNDERS (1982) are used). However, in these approaches, the resulting least squares problem is stated in the n dimensional space of the original variables. One of the advantages of the new approach suggested below is to lead to a least squares problem in a subspace of significantly reduced dimension.

Problem (I) above is a constrained quadratic minimization problem. The idea here is to reformulate it as an unconstrained quadratic minimization problem. Assuming A to be a $m \times n$ matrix of full rank m ($m < n$) - a fairly standard assumption indeed in linear programming - consider any basis of A (*), i.e. a $m \times m$ regular submatrix A_I corresponding to some subset $I = \{1, \ldots, m\}$ of m basic columns.

Now, the system : $B.x = 0$ which is equivalent to

$$\begin{cases} A\,D\,x = 0 & (1) \\ e^T x = 0 & (2) \end{cases}$$

can be rewritten (premultiplying (1) by A_I^{-1}) as :

$$\begin{cases} D_I\,x + A_I^{-1}\,D_J\,x_J = 0 & (1') \\ e^T x = 0 & (2) \end{cases}$$

where the partition of A into $[A_I, A_J]$ (J being the subset of indices of the non basic columns) induces the partition $[x_I, x_J]$ of the variables and $[D_I, D_J]$ of the diagonal matrix D.

(*) Observe that finding a basis of a linear system may be a difficult problem in itself. However, in practice a basis is often at hand or easy to derive from the structure of the problem dealt with. Moreover, if this were not the case, it would always be possible to augment the given system by m artificial variables (with sufficiently large cost coefficients) in order to obtain a $m \times m$ unit matrix as a basis. Notice that in the latter case, the full rank hypothesis is automatically fullfilled.

We denote $A_I^{-1} A_J$ by $R = (r_{ij})_{\substack{i \in I \\ j \in J}}$. R is thus a $m \times (n - m)$ matrix.

Note here that, contrary to what occurs in the simplex method, there is no restriction, a priori (such as feasibility conditions), on the choice of a basis A_I.

Assuming, for notational simplicity, that the basic variables are the first m variables, the system (1')-(2) can then be rewritten as :

$$\begin{cases} d_i x_i + \sum_{j \in J} r_{ij} d_j x_j = 0 \quad \forall\, i = 1, \ldots, m & (3) \\ \sum_{k=1}^{n} x_k = 0 & (4) \end{cases}$$

The first m equations (3) can be used to eliminate the variables x_I from the last one yielding for (4) :

$$- \sum_{i=1}^{m} \left(\frac{1}{d_i} \sum_{j \in J} r_{ij} d_j x_j\right) + \sum_{j \in J} x_j = 0$$

or equivalently :

$$\sum_{j \in J} \left(1 - \sum_{i=1}^{m} \frac{r_{ij} d_j}{d_i}\right) x_j = 0 \qquad (5)$$

The problem of projecting the vector $y = D c$ onto the null space of B is thus equivalent to finding the minimum of $\|x - y\|^2$ subject to (3)-(5).

Writing $\|x - y\|^2 = \|x_I - y_I\|^2 + \|x_J - y_J\|^2$

and eliminating the basic variables x_I by (3) yields

$$(II)\quad \begin{cases} \text{Minimize } f(x) = \sum_{i=1}^{m} \left(y_i + \sum_{j\in J} \frac{r_{ij}\, d_j}{d_i}\, x_j\right)^2 + \sum_{j\in J} (x_j - y_j)^2 \\ \text{subject to} \\ \sum_{j\in J} \rho_j\, x_j = 0 \end{cases} \qquad (6)$$

where, $\forall\ j \in J$, $\rho_j = 1 - \sum_{i=1}^{m} \frac{r_{ij}\, d_j}{d_i}$.

This is a (convex) quadratic minimization problem with only one linear constraint (6) and no sign restrictions on the variables.

We suggest to apply to (II) a conjugate gradient algorithm. Indeed, in the present context, conjugate gradient algorithms enjoy several desirable properties : ease of implementation, low storage space requirements, no matrix manipulations (contrasting with Newton or Quasi-Newton methods). They usually require a rather high accuracy in the one-dimensional minimization process, but this is by no means a drawback here since the cost function being quadratic, the exact minimum in the direction can be obtained at low computational cost (see Appendix).

Due to constraint (6) the gradient will be projected at each step onto the hyperplane with equation

$$\sum_{j\in J} \rho_j\, x_j = 0 \qquad (6)$$

thus resulting in a conjugate gradient method in the corresponding subspace of dimension $n - m - 1$. However, we note that the projection operation onto a hyperplane is easy and computationally inexpensive to carry out (see formulae (7)-(8)).

The j^{th} component of the gradient of f at any point x reads :

$$\frac{\partial f}{\partial x_j} = 2 \sum_{i=1}^{m} \left(y_i + \sum_{j\in J} \frac{r_{ij}\, d_j}{d_i}\, x_j\right) \frac{r_{ij}\, d_j}{d_i} + 2(x_j - y_j).$$

Thus ∇f may be efficiently computed as follows :

(a) First, the m quantities

$$\beta_i = y_i + \sum_{j \in J} \frac{r_{ij}\, d_j}{d_i}\, x_j$$

for the current point x are determined, requiring $\mathcal{O}(m(n - m))$ operations in the case of a 100 % dense matrix R, and $\mathcal{O}(\delta\, m(n - m))$ operations in the case of a sparse matrix R with density δ (percentage of nonzero elements).

(b) Each component of the gradient is then obtained by

$$\frac{\partial f}{\partial x_j} = 2 \sum_{i=1}^{m} \beta_i \frac{r_{ij}\, d_j}{d_i} + 2(x_j - y_j)$$

thus requiring $\theta(\delta\, m\, n)$ elementary operations. The overall complexity for computing the gradient is thus $\theta(\delta\, m\, n)$ and it is seen that full advantage can be taken, with this approach, of the sparsity of the matrix R. Moreover since the initial choice of the basic matrix A_I (which allows eliminating m constraints from the problem) can be made more or less arbitrarily, this flexibility can be exploited by choosing as basic matrix A_I one leading to the minimum number of nonzero terms in $R = A_I^{-1} A_J$. Furthermore, note that (as is frequently the case in practice) if the matrix A already contains a unit matrix, sparsity of the initial matrix A is preserved in the matrix R.

Once the unconstrained gradient $\nabla f = (v_1, v_2, \ldots, v_n)^T$ has been obtained, the projected gradient $g(x)$ onto the hyperplane of equation (6) is easily deduced by :

$$g(x) = v + \mu\, \rho \tag{7}$$

$$\text{with} \quad \mu = - \frac{\rho^T v}{\|\rho\|^2}. \tag{8}$$

This is this vector $g(x)$ which should be taken as the gradient at the current point x in the implementation of the conjugate gradient algorithm.

We now turn to describe the basic p-step conjugate gradient procedure, where p is a specified integer parameter, chosen here in the range $1 \leq p \leq m - n - 1$ (since $m - n - 1$ is the dimension of the subspace of the independent variables).

The p-step conjugate gradient (CG) procedure

(a) Let x^0 be the chosen starting point. Set $w^0 = - g(x^0)$ the starting search direction. Set $k \leftarrow 0$.

(b) Current step k : choose $\lambda_k \geq 0$ such that $x^k + \lambda_k w^k$ minimizes $f(x^k + \lambda w^k)$ over $\lambda \geq 0$.

Let : $x^{k+1} = x^k + \lambda_k w^k$.

Compute $g(x^{k+1})$ and then :

$$w^{k+1} = - g(x^{k+1}) + \beta_k w^k$$

with $\beta_k = \dfrac{\|g(x^{k+1})\|^2}{\|g(x^k)\|^2}$.

(c) Stopping test : if $k = p$, STOP. Otherwise, set $k \leftarrow k + 1$ and return to (b).

It is a well-known property of conjugate gradient algorithms that an exact optimal solution is produced in a finite number of steps when applied to a quadratic function. Here f is a quadratic function of $n - m$ variables x_j $(j \in J)$ restricted to a subspace of dimension $n - m - 1$ (defined by the hyperplane constraint (6)) thus running a p-step conjugate gradient procedure with $p = m - n - 1$ will produce (up to the computer's

numerical accuracy) an exact optimal solution to the least squares problem (II) hence to the direction needed in Karmarkar's algorithm.

4. A FAMILY OF APPROXIMATE DIRECTION-FINDING PROCEDURES BASED ON CONJUGATE GRADIENTS

However, though as already pointed out, the C-G procedure can easily take advantage of the possible sparsity of the constraint matrix A, running $p = m - n - 1$ steps may be computationally too expensive. One of the interesting features of the approach suggested here is that it leads very naturally to a whole family of numerical schemes depending on the level of accuracy at which the least-squares direction-finding subproblem is solved. This is obtained by just replacing the standard C-G procedure on $m - n - 1$ steps by an iterated p-step conjugate gradient procedure (where p is chosen much smaller than $m - n - 1$) which can be stated as follows :

Iterated p-step conjugate gradient

(a) Choose p, an integer ($p << m - n - 1$). Let x^0 be the chosen starting point. Let $k \leftarrow 0$ (iteration number).

(b) At iteration t, x^t is the current point. Run a p-step C-G algorithm with x^t as the starting point. Call x^{t+1} the resulting point.

(c) Termination test : if satisfied, STOP. Otherwise, set $t \leftarrow t + 1$ and return to (b).

Many possible termination tests can be used. A simple one based on the absolute error is to check whether $\|x^{t+1} - x^t\| < \varepsilon$, where ε is some fixed predetermined tolerance. A possibly better choice could be to require that :

$$\frac{\|x^{t+1} - x^t\|}{\|y - x^t\|} < \eta$$

where η is some fixed predetermined tolerance (in this case, the stopping test is based on a relative error involving the vector y which is to be projected).

Observe that an interesting feature of this approximation scheme is that, whatever the stage at which the iterations are stopped, the resulting approximate point x^t defines a direction of displacement for Karmarkar's algorithm which exactly lies in the null space of matrix B : this guarantees that all the points obtained by moving in the direction will be feasible for (P).

It is the author's opinion that allowing, as suggested above, approximate solutions (to within controllable accuracy) to the direction-finding subproblems at each iteration in Karmarkar's algorithm may open the way to considerable computational savings eventually leading to practical implementations actually competitive with the simplex method.

Though a thorough analysis of the influence of inaccuracies in the direction on the overall behaviour of the algorithm (average number of required iterations) has not been carried out yet, there are some good reasons why this influence should be small. In particular :

- since, at each iteration, a projective transformation maps back the current point to the center of the simplex S, there is no risk that by choosing a direction which is not exactly the right one, too small step sizes will result leading to premature convergence of the sequence of points ;

- for the direction defined by an approximate solution x^k to the least squares problem (II) to be a descent direction for the linear function $(D\,c)^T x$, hence for the potential function (*), even a relatively small reduction in the norm $\|x^k - y\|$ is sufficient.

(*) $D\,c$ can be shown to be the gradient of the potential function at the current point.

We expect that computational experiments will confirm this, while making more precise the kind of tradeoff between the tolerance level and the computational effort at each iteration for minimizing the overall running time of Karmarkar's algorithm.

Acknowledgements

P. TOLLA is gratefully acknowledged for a careful reading of a first draft of the manuscript. Many thanks to Mrs. FRANCOIS, LAMSADE for her careful typing of the manuscript.

APPENDIX : Efficient computation of the exact minimum along the conjugate gradient direction

It is well-known (see e.g. MINOUX (1983), chapter 4) that in order to obtain full benefit of the good convergence properties of conjugate gradient methods, a rather high accuracy is required in the solution of the one-dimensional minimization subproblem at each step.

For the direction-finding problem in Karmarkar's algorithm, we are in the favourable situation of minimizing a quadratic function, which allows getting the exact minimum of this one-dimensional subproblem at low computational cost. Indeed, knowing the coefficients a, b, c of the cost function $(a\,\theta^2 + b\,\theta + c)$ along the current conjugate gradient direction w, when expressed as a function of θ (the step size) leads to the exact value of the optimal step : $\theta^* = -\frac{b}{2a}$. However, getting the values of the coefficients a, b, c in our problem in terms of the data at hand (the r_{ij} coefficients and the d_j values) would be computationally rather expensive. We propose, instead another method (a variant of quadratic interpolation) requiring only to evaluate the function value at only one additional point of the form $x' = x + \rho\, w$ (we omit the iteration superscript k for notational simplicity) for some arbitrarily chosen positive value of the parameter ρ (the only restriction on ρ is to avoid very small or very large values to eliminate possible influence of round-off errors).

Indeed, let $f(x)$, $f(x')$ be the function values at x and x' respectively, and g the gradient of f at x. The directional derivative of f along the direction w at the current point x is $\beta = g^T.w$ (of course, $\beta < 0$ since w is a descent direction).

Simple identification then leads to :

$c = f(x)$ (function value for $\theta = 0$),
$b = \beta$ (derivative at $\theta = 0$),
$a\,\rho^2 + b\,\rho + c = f(x')$ (function value for $\theta = \rho$),

for which we deduce :

$$a = \frac{f(x') - f(x) - \beta\,\rho}{\rho^2}$$

hence

$$\theta^* = -\frac{1}{2}\,\frac{\rho^2\,\beta}{f(x') - f(x) - \beta\,\rho}. \qquad (9)$$

From a practical point of view, a good way of choosing the parameter ρ at each iteration of the conjugate gradient algorithm is to set ρ equal to the optimal step size θ^* obtained at the previous iteration (this "dynamic rule" ensures that at each step, the ρ parameter keeps on being of the same order of magnitude as the actual optimal step size, thus reducing the effect of numerical inacurracies). At the start of the process, the first value of ρ has to be chosen arbitrarily : a good precaution is then to check that the corresponding value θ_1^* of θ^* (computed by (9)) is not much smaller or much larger than ρ (in such cases, there might be some significant numerical error on θ^* ; a good precaution could be then to recompute a better value of the optimal step size θ^* at the first iteration, by setting $\rho = \theta_1^*$ and evaluating f at $x' = x + \theta_1^* w$

BIBLIOGRAPHY

GAY D.M. (1985) : "A Variant of Karmarkar's Linear Programming Algorithm for Problems in Standard Form", *Numerical Analysis Manuscript* 85-10, ATRT, Bell Laboratories.

GONZAGA C. (1985) : "A canonical projection algorithm for linear programming", *Memorandum* n° UCB/ERL M85/61, College of Engineering, University of California, Berkeley.

KARMARKAR N. (1984) : "A new polynomial-time algorithm for linear programming", *Combinatorica* 4, 4, 373-395.

LISSER A., MACULAN N., MINOUX M. (1985) : "Approximate one-dimensional minimization of the potential function improves the convergence speed of Karmarkar's method", unpublished.

LUSTIG I.J. (1985) : "A Practical Approach to Karmarkar's algorithm", *Technical Report* SOL 85-5, Dept. of Operations Research, Stanford University, Stanford, CA.

MEGGIDO N. (1985) : "A variation on Karmarkar's algorithm (preliminary report)", IBM Research Laboratory, San Jose, California.

MINOUX M. (1983) : *Programmation Mathématique : Théorie et Algorithmes*, Vol. 1, Dunod, Paris ; English translation : John Wiley & Sons, 1986.

NICKELS W., RODDER W., XU L., ZIMMERMANN H.J. (1985) : "Intelligent gradient search in linear programming", *European Journal of Operational Research* 22, 293-303.

PAIGE C.C., SAUNDERS M.A. (1982) : "LSQR : An Algorithm for Sparse Linear Equations and Sparse Least Squares", *ACM Transactions on Mathematical Software* 8, 1, 43-71.

SCHRIJVER A. (1985) : "The New Linear Programming Method of Karmarkar", *CWI Newsletters.*

TODD M.J., BURRELL B.P. (1985) : "An Extension of Karmarkar's algorithm for linear programming using dual variables", *Technical Report* n° 648, School of Operations Research and Industrial Engineering, College of Engineering, Cornell University, Ithaca, N.Y.

TOMLIN J.A. (1985) : "An experimental approach to Karmarkar's projective method for linear programming", Ketron Inc., Mountain View, California.

YE Y., CHIU S.S. (1985) : "Recovering the shadow price in projection methods of linear programming", Preliminary Draft, Engineering Economic Systems Department, Stanford University.