CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
(Université de Paris-Dauphine)
Unité Associée au CNRS n° 825

VARIANTS OF KARMARKAR's ALGORITHM

CAHIER N° 90

mars 1989

A. LISSER

P. TOLLA

CONTENTS

ETUDE DE QUELQUES VARIANTES DE L'ALGORITHME DE KARMARKAR

RESUME

L'objet de ce cahier est de montrer à quel point l'étude et l'expérimentation de quelques variantes de l'algorithme de KARMARKAR ont permis d'élaborer un logiciel plus performant que la version originelle présentée par KARMARKAR.

Plusieurs méthodes ont été testées ; il s'agit en particulier : de la forme primale-duale, de la méthode des deux phases, de quelques méthodes de calcul de la projection du vecteur de la fonction économique, des méthodes de recherche monodimensionnelle ainsi qu'un essai de traitement implicite des bornes supérieures des variables.

Mots-clés : Forme primale-duale, méthode des deux phases, projection, dichotomie, bornes supérieures des variables.

VARIANTS OF KARMARKAR's ALGORITHM

ABSTRACT

The purpose of this paper is to show how studying few variants of KARMARKAR's algorithm allowed us to perform a software faster than the original version presented by KARMARKAR.

Several methods have been implemented such as : primal-dual form, two phasis method, few methods of the calculation of the projected vector, dichotomy method and an implicit treatment of the upper bounds of the variables.

Keywords : Primal-dual form, two phasis method, projection, dichotomy, upper bounds of variables.

I - INTRODUCTION

In the full of 1984, Narendra KARMARKAR's new projective algorithm for linear programming has caused quite a stir in the press (The New York Times, Time Magazine, Science, ...) mainly because of an implementation of the algorithm presented by the author out performed one implementation of the classic simplex method by a factor of over 50 on medium-scale problems.

N. KARMARKAR (1984) demonstrated that his algorithm has a polynomial bound on the worst case running time better than the ellipsoid algorithm.

In this paper, we improve the efficiency of KARMARKAR's algorithm by :

- modifying the calculation of the projection of the vector $y = Dc$;
- showing that it is possible to implement KARMARKAR's algorithm with large stepsizes preserving the polynomial complexity behaviour and improving the practical efficiency ;
- proposing heuristics for implicit treatment of the upper bounds of the variables.

II - KARMARKAR's ALGORITHM IN BRIEF

KARMARKAR's algorithm does not solve the standard linear programming problems but finds an optimal solution to the restricted problem :

$$\text{minimize } c^t x$$
$$\text{s.t. } x \in \Omega \cap \Delta$$

where $c \in Z^{n+1}$, $x \in R^{n+1}$ and $\Omega = \{x / A x = 0\}$ is the solution subspace of a homogeneous system of linear equations and Δ is the simplex contained in R^{n+1} and defined by :

$$\Delta = \{x / e^t x = 1, x \geq 0\}.$$

A is an integer matrix with m rows and n columns and m < n, rank(A) = m. We shall make few assumptions :

(i) the problem is feasible and the center of the simplex $x_0 = (1/n, ..., 1/n)^t$ is a feasible point (i.e. $x_0 \in \Omega$) ;

(ii) the minimum value of the objective function is zero, i.e. $c^t x^* = 0$;

(iii) a stopping criteria q is given such that $c^t x / c^t x_0 \leq 2^{-q}$.

The main algorithm

The main algorithm begins with a starting point x_0 which lies in the interior of the polytope. Let the iteration k be the current iteration of KARMARKAR's algorithm and let $x^k = (x^k_1, ..., x^k_{n+1})^t$ be the current point.

(1) Let $B = \begin{bmatrix} AD \\ \hline e^t \end{bmatrix}$

where $D = \text{diag}(x^k_1, ..., x^k_{n+1})$ and e is a vector with all components equal to 1.

(2) Let $c_p = [I - B^t(BB^t)^{-1} B]Dc$ be the orthogonal projection of Dc onto the null space of B.

(3) The objective function can be improved by moving from the center of the simplex x_0 in the direction of $-c_p$:

$$b' = x_0 - \alpha r c_p / \|c_p\|$$

where r is the radius of the largest inscribed sphere in the simplex, α is a parameter between 0 and 1 and $\|c_p\|$ is the euclidien norm of the vector c_p.

(4) The next point x^{k+1} is obtained by applying to b' the projective transformation (see KARMARKAR (1984)) :

$$x^{k+1} = D b' / (e^t D b').$$

As shown in MINOUX (1976), the cost of computing the search direction (step (2)) can be reduced by replacing the calculation of the pseudo-inverse of matrix by solving an unconstrained minimization of a convex quadratic function.

An improved implementation suggested by LISSER, MACULAN and MINOUX, which consists in performing an approximate one dimensional minimization of the potential function at each iteration, is presented in this paper and extended to medium scale problems.

We now turn to discuss possible ways of reducing the computational effort for getting the search direction $-c_p$.

III - CALCULATION OF THE PROJECTED VECTOR y = Dc BY SOLVING A CONSTRAINED QUADRATIC PROGRAM

The step (2) of KARMARKAR's algorithm consists in finding the direction -cp of steepest descent within the polytope. MINOUX (1986) showed that the projection of the vector y = Dc onto the null space defined by $Bc_p = 0$ can be stated as the following least squares problem :

(I) minimize $\|Dc - c_p\|$
s.t. $B\ c_p = 0.$

MINOUX proposed to solve the problem in a reduced dimension subspace. We propose to solve problem (I) in n-dimensional space of original variables.

Let us assume the matrix A a full rank matrix and apply the optimality conditions to the problem (I) :

(II) $I_n c_p + B^t y = Dc$ (II.1)

$B\ c_p = 0$ (II.2)

where y is the lagrangian multipliers vector and I_n the identity matrix.

The normal equations are obtained by the system :

$$B\,B^t\,y = B\,Dc \qquad \text{(II.3)}$$

which can be solved by CHOLESKY's method (see SHANNON (1985)) or by iterative methods.

We apply the conjugate gradient method (see MINOUX (1986) and LISSER (1987)) to solve the system (II.3).

The projection vector cp is obtained by replacing the vector y in (II.1) by the solution of the normal equations.

Note that the projection problem can be viewed as solving two sub-quadratic programs (see HOOKER (1986)) :

$$\text{(III)} \quad \text{minimize } \|z - Dc\|$$
$$\text{s.t. } ADz = 0$$

and compute cp by projecting the result of the program (III) on the normalization constraint :

$$\text{(IV)} \quad c_p = P\,z^*$$

where z^* is the solution vector of the program (III) and P is the matrix projection onto $\{z \,/\, e^t z = 0\}$. P is defined by :

$$\text{(V)} \quad P = [I - ee^t/n].$$

It is easy to see that solving problems (III) and (IV) reduces significantly the effort of the calculation of the vector cp. The third step of KARMARKAR's algorithm

consists in moving in the direction -cp across a distance αr where α is the stepsize and r is the radius of the largest inscribed sphere in the simplex, i.e. r = 1/sqrt(n(n - 1)).

However, when the constant $\alpha = 0.25$ (see KARMARKAR (1984)), the number of iterations grows linearly with respect to the dimension of the problem. We show in the next section that a large adaptative stepsize can be allowed to obtain implementations of KARMARKAR's algorithm achieving both theoretical and practical efficiency.

IV - KARMARKAR's ALGORITHM WITH AN EMBEDDED ONE DIMENSIONAL SEARCH SCHEME

As shown in LISSER, MACULAN and MINOUX (1987), the basic idea of the proposed scheme consists in carrying out a one dimensional search along the same displacement direction as defined in KARMARKAR (1984) in order to approximate the minimum of the potential function in this direction.

Recall that the potential function is defined by :

$$f(x) = \sum_{i=1}^{n} \log\left[\frac{c^t x}{x_i}\right].$$

Note that f is defined at every point x lying in the interior of the simplex and is linearly unimodal (see TODD and BURREL (1985)). This potential function goes to infinity as the stepsize α approaches α_{max}, i.e. the maximum value of α for which $b'_k = e/n - \alpha r c_p / \|c_p\|$ does not reach the boundary of the polytope.

We want α to be as large as possible subject to the condition that every component of b'_k be no less or equal to zero. It is easy to see that α_{max} can be as large as (n - 1) (see LISSER, MACULAN and MINOUX (1987)). To approximate α^* where the minimum of f along $c_p / \|c_p\|$ is attained, we optimize the following program :

(VI) minimize $g(e/n - \alpha r c_p / \|c_p\|)$
s.t. $(e/n - \alpha r c_p / \|c_p\|) \geq \epsilon$

where g is the potential function in the transformed space and ϵ is a small positive number (see HOOKER (1986)). It is shown in LISSER, MACULAN and MINOUX (1987) that all computations of the one-dimensional search procedure can be performed in polynomial time.

Obviously, many possible search procedures using the unimodality property such as golden section search, FIBONACCI search, etc. could be used (see MINOUX (1986) and LISSER (1987)).

It is easy to see that small number of function evaluations is cheaper than the other computations to be performed per iteration.

Numerical experimentations show that this variant of KARMARKAR's algorithm is efficient when compared to the one proposed by KARMARKAR (1984), but as the problem treated handles bounds of the variables, we proposed a few heuristics to treat the bounds of the variables separately like in the simplex method.

V - ON THE TREATMENT OF THE BOUNDS OF THE VARIABLES

KARMARKAR (1984) has suggested procedure using flipping technique to handle the upper bounds of the variables (see LUSTIG (1985)). Note that a proper proof of convergence has not yet been exhibited and the procedure has not worked well on problems of medium size.

TOLLA (1987a) has suggested a heuristic to treat the bounds of the variables implicitly and we proposed a procedure which works well in practice (see LISSER (1987)). TODD (1988a) has proposed a variant of KARMARKAR's algorithm to permit to handle the bounds of the variables just after the projection step of the algorithm.

A linear program with upper bounded variables may be defined by :

Maximize $c^t x$

s.t. $A x \leq b$

$0 \leq x \leq \beta$

Let the sets Γ and P be defined by :

$\Gamma = \{x / A x \leq b, x \geq 0\}$,

$P = \{x / x \leq \beta\}$.

Theorem (see TOLLA (1987b))

Let $x^1 \in \text{Int}(\Gamma \cap P)$ and $x^2 \in \text{Int}(P)$ such that $c^t x^2 > c^t x^1$, let $I_{cv}(x^2)$ be the set of the indices of the components of x^2 which do not satisfy the bounds on x, and let x^μ defined by :

$x^\mu = (1 - \mu)x^1 + \mu x^2$ with $0 \leq \mu \leq 1$.

If at least one of the bounds is violated by a component of x^2, the optimal point (with regard to the objective function) of $[x^1, x^2] \cap Z \cap P$ is :

$x^* = (1 - \mu)x^1 + \mu x^2$

where

$$\mu^* = \underset{i \in I_{cv}(x^2)}{\text{Min}} \begin{cases} (\beta_i - x^1_i)/(x^2_i - x^1_i) \text{ if } x^2_i - x^1_i > 0 \\ x^1_i/(x^1_i - x^2_i) \text{ if } x^2_i - x^1_i < 0 \end{cases}$$

Modification of KARMARKAR's algorithm

Step 1

Let $k = 0$ and $x^k = (x^k_1, ..., x^k_n)^t$ and $D = \text{DIAG}(x^k_1, ..., x^k_n)$

$$B = \left| \begin{array}{c} AD^k \\ - - - \\ e^t \end{array} \right|$$ (without the constraints on the bounded variables).

Step 2

Calculation of c_p and $c' = c_p / \|c_p\|$.

Step 3

$b_k = e/n - \alpha r c'$

where α is the stepsize and r the radius of the inscribed sphere in the simplex.

Step 4

$x'^{k+1} = D^k b^k / (e^t D^k b^k)$.

Step 5

$x'^{k+1} \in \text{Int}(\Gamma)$. If $x'^{k+1} \in \text{Int}(P)$, then $x^{k+1} = x'^{k+1}$; else, compute μ^* as indicated below. Let $\mu'^{k+1} = \text{Min}(\mu^*, 1)$. Select μ^{k+1} such that $0 < \mu^* < \mu'^{k+1}$; then $x^{k+1} = [1 - \mu^{k+1}]x^k + \mu^{k+1} x'^{k+1} \in \Gamma \cap P$.

Step 6

If $c^t x^{k+1} - c^t x^k < \epsilon$, stop ; else, $k = k + 1$; go to step 1.

REMARK : If one variable x^k_j is very close to its bound β_j, one can put $x^k_j = \beta_j$ and solve the reduced linear program ; this procedure is often used in linear programming and may reduce the CPU time.

VI - NUMERICAL RESULTS

Tests have been carried out on minimal cost flow problems. The linear program take the following form :

$$\text{minimize } c^t x$$
$$\text{s.t. } Ax \leq IAS$$
$$Ix \leq IBP$$
$$x \geq 0$$

with x, c^t and IBP (bounds) : (N x 1) integer vectors, IAS (RIGHT HAND SIDE) : (M x 1) integer vector and A (M x N) integer matrix, each of its columns contains two non zero elements - 1 and + 1.

The row indices were randomly generated between 1 and M (number of rows of A). The components of the cost vector and of the RHS were generated randomly between 1 and 10.

Two linear programming codes were used : MPSX/MIP and one implementing KARMARKAR's method coded in FORTRAN and tested on IBM 3090 in double precision arithmetic, plus FRIFLO code used in EDF (Electricité de France) and written by EA and MAURRAS (1981).

Test problems were randomly generated with respectively 10 rows and 30 columns, 20 rows and 60 columns, 30 rows and 90 columns, 40 rows and 120 columns, 50 rows and 150 columns, 500 rows and 1.000 columns, 1.000 rows and 1.500 columns, 1.500 rows and 2.000 columns, 2.000 rows and 2.500 columns and 2.500 rows and 3.000 columns.

Table I contains the results for the three codes, the number of iterations and the CPU time for the small size test problems. Recall that the method used in KARMARKAR's code is the primal-dual, the step size $\alpha = 0.95$ and the arithmetic precision is 10E-08. The CHOLESKY method was implemented and used in the calculation of the projection of the vector Dc (see LISSER (1987)).

Note that the CPU time of KARMARKAR's code is markedly larger than the one obtained by the two other codes. Two reasons for this : the first is essentially the explicit computation of the orthogonal projection at each iteration and the explicit treatment of the upper bounds of the variables. The second one is the primal-dual form which makes larger the dimension of the matrix.

Table II contains the results of the KARMARKAR's algorithm using the two phasis method. The CPU time is divided by at least a factor 4 compared to the primal-dual form (from 3936.24 seconds to 797.76 seconds for the fifth test). The primal-dual form proposed by KARMARKAR (1984) gives the maximum CPU time and the maximum number of iterations.

Table III contains the results of KARMARKAR's algorithm using the conjugate gradient method used in the calculation of the projected vector cp (see MINOUX (1983) and LISSER (1987)). The second column shows the dimension of the matrix of the constraints (bounds included), the third column shows the number of iterations of KARMARKAR's algorithm, the fourth one shows the interval where the number of iterations of the conjugate gradient method (CGM) at each iteration of KARMARKAR's algorithm is enclosed in the first phase. The results of the second phase are shown in the fifth and sixth columns. The last column contains the CPU time.

The most important fact observed in this table is the CPU time which is significantly less than the one obtained in table II (only 102.97 seconds instead of 797.76 seconds for the last test problem). The reason for this is the sparsity of the matrix which is exploited when the CGM is used (see MINOUX (1987)).

Tables III and IV contain the results of KARMARKAR's algorithm with the primal form on the first two problems for various fixed values of the stepsize α (from $\alpha = 0.25$ to $\alpha = 0.99$). Note that larger values of α markedly decrease the number of iterations and the CPU time (see LISSER, MACULAN and MINOUX (1987)).

Tables VI, VII, VIII, IX and X contain the results of the modified algorithm using the dichotomy method described in LISSER, MACULAN and MINOUX (1987)

in the second phase. The stepsize α is fixed at $\alpha = 0.95$ for the first phase. The third column of this table gives the number of iterations for the first phase and the number of iterations for the second phase. The fourth column shows the interval of the number of iterations of CGM for the first phase and the number of iterations of the CGM of each iteration for the second phase.

Two important facts are observed : the first is the number of iterations which is less than 10 for all the problems. The CPU time of the last problem is 19.78 seconds instead of 102.92 seconds when $\alpha = 0.95$. The second prominent feature is that the value of α is always significantly greater than 1 (see LISSER, MACULAN and MINOUX (1987)).

Table XI shows the results of the medium size test problems obtained by KARMARKAR's algorithm using the dichotomy method in the second phase. Note that the number of iterations of the second phase is about eight iterations for all the problems.

Tables XII and XIII contain the computational results with the dichotomy method applied to phase I and phase II of KARMARKAR's algorithm. The number of iterations is about 5 for phase I and 10 for phase II for the whole of the problems.

Observe that the optimum value is lightly perturbed when the dichotomy method is used in the first and in the second phases. It is obvious that the CPU time decreases when the dichotomy method is applied to the two phasis.

The simplex method treats separately the upper bounds of the variables, the procedure described in the previous section gives the results contained in table XIV. The dichotomy method is used in the two phasis. Note how the CPU time decreases when the upper bounds are treated implicitly (from 522.547 seconds to 63.37 seconds for the last medium size test problem).

The heuristic used to treat implicitly the upper bounds of the variables perturbs the value of the optimum. To avoid this disadvantage, one can implement the variant of the algorithm proposed by TODD (1988a)).

The most important result is contained in table XV : in fact, our modified algorithm appears more efficient and is about 5 times faster than MPSX/MIP code.

Finally, the modified algorithm could be further improved if an efficient updating method is applied. Tests of this remain for future implementations.

	PRIFLO			MPSX			KARMARKAR (Primal-Dual) $\alpha = .95$; EPS = 10^{-8}			
DIMENSION	OBJECTIVE	ITER	CPU TIME	OBJECTIVE	ITER	CPU TIME	DIMENSION OF THE MATRIX	O. P. T.	ITER	CPU TIME
10-30	412	16	.0083	412	15	.72	50-100	411.998	78	7.6
20-60	1019	42	.0114	1019	42	.74	100-200	1018.995	104	89.68
30-90	1516	59	.0166	1516	64	.77	150-300	1515.997	123	575.85
40-120	2108	80	.0210	2108	84	.78	200-400	2107.996	134	1825.03
50-150	2619	95	.276	2619	112	.81	250-500	2618.99	142	3936.24

TABLE I : Computational results of the code PRIFLO of EDF (Electricité de France), MPSX/MIP code and KARMARKAR's algorithm (all times in CPU seconds on an IBM 3090)

Tests	Dimension	Iterations Phase I	Iterations Phase II	CPU Time
10-30	30-50	10	33	2. 42
20-60	60-100	12	48	27. 41
30-90	90-150	14	61	119. 32
40-120	120-200	17	72	348. 3
50-150	150-250	15	81	797. 76

TABLE II : Computational results with the CHOLESKY method applied to the projection computation of the vector Dc

DIMEN-SION	DIMENSION OF THE MATRIX	PHASE I		PHASE II		CPU TIME
		Iter. (Kmarkarkar)	CG Iter. Interval	Iter. (Karmarkar)	GC Iter. Interval	
10-30	30-50	10	[2 - 33]	33	[21 - 34]	1. 77
20-60	60-100	12	[2 - 47]	48	[28 - 66]	8. 301
30-90	90-150	14	[2 - 57]	61	[35 - 131]	25. 67
40-120	120-200	17	[2 - 64]	72	[27 - 285]	59. 57
50-150	150-250	15	[3 - 61]	81	[34 - 270]	102. 92

TABLE III : Iterations counts and CPU time obtained by KARMARKAR's algorithm using the conjugate gradient method to compute the projected vector C_p

TABLE IV

Test 1

		PHASE I		PHASE II			
DIMENSION	OBJECTIVE	Iter.	CG Iter. Interval	Iter.	CG Iter. Interval	Alpha	CPU TIME
10-30	411.91	65	[2-32]	108	[20-33]	.25	11.64
10-30	411.94	30	[2-32]	57	[20-33]	.5	6.17
10-30	411.94	17	[2-32]	39	[20-33]	.75	4.26
10-30	412.01	11	[2-32]	33	[20-33]	.90	3.36
10-30	411.98	10	[2-32]	33	[20-33]	.99	3.3

TABLE V

Test 2

		PHASE I		PHASE II			
DIMENSION	OBJECTIVE	Iter.	CG Iter. Interval	Iter.	CG Iter. Interval	Alpha	CPU TIME
20-60	1018.95	71	[2-46]	156	[27-64]	.25	53.04
20-60	1018.99	33	[2-46]	84	[27-64]	.5	28.52
20-60	1019.07	19	[2-46]	56	[27-64]	.75	19.02
20-60	1018.97	15	[2-46]	51	[27-64]	.90	17.02
20-60	1018.96	12	[2-46]	48	[27-64]	.99	15.99

TABLES IV and V : Iteration counts, CPU time and different stepsizes for the two phasis method of KARMARKAR's algorithm on two selected test problems

TABLE VI

Test 1

Dimension	Phase	Iter	CG Iter	Objective	Alpha max	Alpha opt
10-30	I	11	[2-33]	-	-	.95
10-30	II	12	21	282.34	2.286	2.254
10-30		13	25	338.95	2.337	2.305
10-30		14	29	390.22	4.097	4.037
10-30		15	33	406.11	4.083	4.022
10-30		16	33	411.034	5.417	5.336
10-30		17	33	412.13	5.128	5.053

CPU time = 0.696761

TABLE VII

Test 2

Dimension	Phase	Iter	CG Iter	Objective	Alpha max	Alpha opt
20-60	I	13	[2-47]	-	-	.95
20-60	II	14	28	734.262	3.112	3.067
20-60		15	33	882.064	3.736	3.731
20-60		16	47	962.086	4.348	4.284
20-60		17	58	1003.343	5.943	5.854
20-60		18	64	1014.272	5.865	5.777
20-60		19	66	1013.123	7.406	7.294
20-60		20	66	1018.965	7.82	7.702
20-60		21	65	1019.035	7.918	7.799

CPU time = 2.619551

TABLE VIII

Test 3

Dimension	Phase	Iter	CG Iter	Objective	Alpha max	Alpha opt
30-90	I	15	[2-57]	-	-	.95
30-90	II	16	35	1123.382	5.03	4.955
30-90		17	56	1297.079	3.964	3.906
30-90		18	70	1431.856	5.864	5.776
30-90		19	89	1489.094	6.707	6.606
30-90		20	105	1509.993	8.244	8.119
30-90		21	114	1514.902	9.064	8.926
30-90		22	124	1515.735	8.15	8.027
30-80		23	140	1516.004	10.089	9.936
30-90		24	243	1516.033	9.55	9.405

CPU time = 7.18596

TABLE IX

Test 4

Dimension	Phase	Iter	CG Iter	Objective	Alpha max	Alpha opt
40-120	I	18	[2-64]	-	-	.95
40-120	II	19	27	1528.351	5.382	5.032
40-120		20	56	1831.796	5.147	5.07
40-120		21	90	2014.934	7.175	7.067
40-120		22	126	2062.698	5.727	5.641
40-120		23	132	2095.063	9.184	9.045
40-120		24	138	2103.631	9.191	9.052
40-120		25	132	2106.598	11.098	10.928
40-120		26	183	2107.068	11.714	11.536

CPU time = 10.088

TABLE X

Test 5

Dimension	Phase	Iter	CG Iter	Objective	Alpha max	Alpha opt
50-150	I	17	[2-60]	-	-	.95
50-150	II	18	34	1806.912	5.671	5.586
50-150		19	63	2232.312	6.046	5.956
50-150		20	100	2458.383	6.494	6.397
50-150		21	173	2564.674	8.117	7.994
50-150		22	185	2595.342	6.437	6.341
50-150		23	218	2612.77	8.852	8.717
50-150		24	235	2618.146	11.391	11.216
50-150		25	280	2619.106	12.927	12.729
50-150		26	402	2619.133	10.779	10.615

CPU time = 19.780273

TABLES VI, VII, VIII, IX and X : Computational results with the dichotomy method applied to the second phase of KARMARKAR's algorithm

DIMENSION	DIMENSION OF MATRIX	Phase I		Phase II		CPU time	OPTIMUM
		Iter	CG Iter	Iter	CG Iter		
500-1000	1000-1500	40	[2, 62]	7	[38, 1002]	268.49	10815
1000-1500	1500-2000	44	[2, 58]	8	[33, 1002]	433.33	11127.063
1500-2000	2000-2500	50	[2, 55]	7	[28, 1002]	470.32	10520.235
2000-2500	2500-3000	54	[2, 52]	8	[31, 1002]	659.28	10582.988
2500-3000	3000-3500	56	[3, 52]	7	[27, 1002]	651.77	10447.249

TABLE XI : Computational results with dichotomy method applied to phase II of KARMARKAR's algorithm

TABLE XII

Dimension	Phase I		Phase II		Optimum	CPU time
	Iter	CG Iter	Iter	CG Iter		
10-30	5	[2, 33]	3	[22, 32]	412.06	.593
20-60	4	[2, 47]	6	[27, 62]	981.133	1.372
30-90	4	[2, 57]	10	[33, 146]	1514.213	6.859
40-120	4	[2, 64]	6	[27, 129]	2015.83	4.436
50-150	4	[3, 61]	10	[36, 402]	2603.3	19.017

TABLE XIII

Dimension	Phase I		Phase II		Optimum	CPU time
	Iter	CG Iter	Iter	CG Iter		
500-1000	4	[3, 62]	7	[39, 1002]	10688.641	193.11
1000-1500	4	[2, 58]	8	[33, 1002]	11080.635	313.17
1500-2000	4	[2, 55]	10	[29, 1002]	10538.911	577.03
2000-2500	4	[2, 52]	9	[29, 1002]	10664.199	572.953
2500-3000	4	[2, 62]	8	[29, 1002]	10417.653	522.547

TABLES XII and XIII : Computational results with the dichotomy method applied to phase I and phase II of KARMARKAR's algorithm

Dimension	Two phasis method : explicit treatment of the bounds of variables				Two phasis method : implicit treatment of the bounds of variables			
	Matrix A	Iter PI	Iter PII	CPU time	Matrix A	Iter PI	Iter PII	CPU time
10-30	30-50	5	8	.598	10-30	4	2	.082
20-60	60-100	4	6	1.372	20-60	4	3	.272
30-90	90-150	4	10	6.859	30-90	4	2	.409
40-120	120-200	4	6	4.436	40-120	4	2	.531
50-150	150-250	4	10	19.017	50-150	4	2	.082
500-1000	1000-1500	4	7	193.11	500-100	4	4	11.89
1000-1500	1500-2000	4	8	313.17	1000-1500	4	5	17.948
1500-2000	2000-2500	4	10	577.03	1500-2000	4	6	33.089
2000-2500	2500-3000	4	9	572.953	1500-2000	4	8	54.78
2500-3000	3000-3500	4	8	522.254	2500-3000	4	9	63.37

TABLE XIV : Computational results of KARMARKAR's algorithm with explicit/implicit treatment of the upper bounds of the variables

Dimension	Explicit treatment	Implicit treatment
10-30	412.06	375.03
20-60	981.133	1012.76
30-90	1514.218	1082.57
40-120	2015.83	1998.94
50-150	2603.8	2612.69
500-1000	10688.641	9787.44
1000-1500	11080.635	10366.64
1500-2000	10538.911	10017.33
2000-2500	10664.199	10200.35
2500-3000	10417.653	10171.25

TABLE XV : Objective function value obtained by KARMARKAR's algorithm with explicit/implicit treatment of the upper bounds of the variables

Dimension	KARMARKAR's code				MPSX/MIP code		
	Iter Phase I	Iter Phase II	Optimum	CPU time	Iter	Optimum	CPU time
10-30	4	2	375.03	.082	15	412	.72
20-60	4	3	1012.76	.272	42	1019	.74
30-90	4	2	1482.57	.409	64	1516	.77
40-120	4	2	1998.94	.532	84	2108	.78
50-150	4	2	2612.69	.737	112	2619	.81
500-1000	4	4	9787.44	11.89	487	10802	160.2
1000-1500	4	5	10366.64	17.948	479	11129	175.2
1500-2000	4	6	10017.33	33.089	493	10517	211.2
2000-2500	4	8	10200.35	54.78	496	10577	248.4
2500-3000	4	9	10171.25	63.37	496	10449	284.0

TABLE XVI : Computational results obtained by KARMARKAR's algorithm (with implicit treatment of the upper bounds of the variables) and MPSX/MIP code

ACKNOWLEDGEMENTS

Many thanks to Mrs. François, LAMSADE, for her careful typing of the manuscript.

REFERENCES

B.P. BURRELL, M.J. TODD (1985) : "The ellipsoid method generates dual variables", Mathematics of Operations Research 10, 688-700.

K. EA, J.F. MAURRAS (1981) : "Une adaptation des méthodes primales et duales du simplexe au problème de flots à coût minimal sur un graphe sans multiplicateur", EDF (Clamart), HR 31-0556.

C. GONZAGA (1985) : "A conical projection algorithm for linear programming", Manuscript, Department of Electrical Engineering and Computer Science, University of California, Berkeley, to appear in Mathematical Programming.

J.N. HOOKER (1986) : "KARMARKAR's linear programming", Interfaces 16, 75-90.

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

A. LISSER (1987) : Un logiciel dérivé de l'algorithme de KARMARKAR pour la résolution de programmes linéaires de grande taille, Université de Paris-Dauphine, Thèse de Doctorat.

A. LISSER, N. MACULAN, M. MINOUX (1987) : "Large steps preserving polynomiality in KARMARKAR's algorithm", Université de Paris-Dauphine, Cahier du LAMSADE n° 77.

I.J. LUSTIG (1985) : "A practical approach to KARMARKAR's algorithm", Technical Report SOL 85-5, Department of Operations Research, Stanford University, Stanford, CA.

M. MINOUX (1983) : Programmation mathématique : Théorie et algorithme, Volume I, Dunod, Paris ; English translation : John Wiley and Sons, 1986.

M. MINOUX (1986) : "New suggested implementations of KARMARKAR's algorithm", Université de Paris-Dauphine, Cahier du LAMSADE n° 71.

D.F. SHANNO (1988) : "Computing KARMARKAR's projections quickly", Mathematical Programming 41, 61-71.

M.J. TODD (1988a) : "Exploiting special structure in KARMARKAR's linear programming algorithm", Mathematical Programming 41, 97-113.

M.J. TODD (1988b) : "Improved bounds and containing ellipsoids in KARMARKAR's linear programming algorithm", Mathematics of Operations Research, Vol. 13, n° 4, 650-659.

P. TOLLA (1987a) : "Validation numérique de l'algorithme de KARMARKAR", Université de Paris-Dauphine, Cahier du LAMSADE n° 76.

P. TOLLA (1987b) : "Amélioration des performances de l'algorithme de KARMARKAR dans le cas de programmes linéaires à variables bornées supérieurement", Université de Paris-Dauphine, Cahier du LAMSADE n° 82.