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

LARGE STEPS PRESERVING POLYNOMIALITY IN KARMARKAR'S ALGORITHM

CAHIER N° 77
mai 1987

A. LISSER
N. MACULAN
M. MINOUX

TABLE DES MATIERES

DES GRANDS PAS DE DEPLACEMENT PERMETTANT DE CONSERVER LA POLYNOMIALITE DE L'ALGORITHME DE KARMARKAR

RESUME

L'objet de cette note est de montrer comment des pas de déplacement plus grands que ceux proposés par N. KARMARKAR (1984) peuvent être utilisés tout en conservant la polynomialité de la méthode.

Ainsi, la classe de méthodes de recherche mono-dimensionnelle proposée dans cette note peut être utilisée dans l'implémentation de l'algorithme de Karmarkar permettant d'obtenir une meilleure efficacité aussi bien sur le plan théorique que pratique.

LARGE STEPS PRESERVING POLYNOMIALITY IN KARMARKAR'S ALGORITHM

ABSTRACT

The purpose of this note is to show how much larger step sizes than the one originally proposed by N. KARMARKAR (1984) can be allowed while preserving polynomiality of the method.

Thus, the class of one-dimensional search schemes suggested in this note can be used to obtain implementations of Karmarkar's algorithm achieving both theoretical and practical efficiency.

1. INTRODUCTION

Karmarkar's polynomial time algorithm for linear programming (see KARMARKAR (1984)) finds an optimal solution to problems stated into the form :

$$(LP) \quad \text{minimize} \quad c^T x$$
$$\text{subject to} \quad x \in \Omega \cap S$$

where $c \in Z^n$, $x \in R^n$ and the solution set is the intersection of the simplex $S = \{x \in R^n \quad e^T x = 1, x \geq 0\}$ and the linear subspace $\Omega = \{x \in \mathbb{R}^n \mid A x = 0\}$, $e^T = (1, 1, \ldots, 1)$, A is an integer matrix with m rows and n columns, and $m < n$, rank $(A) = m$.

In addition one assumes that :

i) the center of the simplex $x_o = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})^T$ is a feasible solution to (LP), thus it can be taken as a starting point ;

ii) (LP) has an optimal solution x^* such that $c^T x^* = 0$.

At each iteration k where the current point is x^k, the problem is transformed by applying a projective transformation π_k such that the image y^k of the current point x^k is, again, the center of the simplex $S = \{y \in \mathbb{R}^n \mid e^T y = 1, y \geq 0\}$. A direction h, such that $\|h\| = 1$, is then computed in the transformed space, the next iteration in this space is defined by :

$$y^{k+1} = y^k + \alpha r h,$$

where $r = \dfrac{1}{\sqrt{n(n-1)}}$ is the radius of the largest sphere inscribed in S and α is a positive (step size) coefficient. By applying the inverse projective transformation $(\pi^k)^{-1}$, y^{k+1} is then mapped back to x^{k+1}, the next iterate in the original space.

The proof of the polynomiality of the algorithm depends on a proper choice of the parameter α. First of all, to ensure that y^{k+1} still lies in the simplex S for any move direction h, α should not be taken greater

than 1. Indeed, even smaller values of α are required in the proof, and the constant value $\alpha = \alpha_K = \frac{1}{4}$ is suggested by KARMARKAR (1984) as a good compromise.

However, when the constant $\alpha_K < 1$ is taken throughout, the maximum number of iterations required by the algorithm to achieve a target reduction factor 2^{-f} $(f > 0)$ on the objective function value is of order $O[n(f + \log n)]$.

Computational experiments confirm in that case that the number of steps grows, on the average, linearly with respect to the dimension of the problem. Since the work per iteration in Karmarkar's algorithm is significantly greater than in the simplex method (DANTZIG (1963)), it is seen that such a choice of the α parameter cannot result in competitive implementations.

A number of implementations set up independently by various research workers in the field tried the natural idea of allowing larger step sizes to be taken in the course of the algorithm (see for instance NICKELS and al. (1985)).

In all cases, significant reductions in the number of iterations necessary to get a given accuracy were reported, but the price to pay for practical efficiency was that none of these schemes were able to maintain theoretical efficiency, i.e. polynomial complexity.

The purpose of this note is to show that it is indeed possible to implement Karmarkar's algorithm with large adaptive step sizes in order to preserve the same worst case polynomial complexity behaviour and to achieve better practical efficiency.

2. KARMARKAR'S ALGORITHM WITH AN EMBEDDED ONE DIMENSIONAL SEARCH SCHEME

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 :

$$f(x) = \sum_{i=1}^{n} \log \frac{c^T x}{x_i} \qquad (1)$$

This potential function is defined at every point x lying in the interior of the simplex S and goes to infinity as one approaches (from the interior) any point $\bar{x}$ of the boundary such that $c^T \bar{x} > 0$. Moreover, it can be shown (see TODD and BURREL (1985)), that f is linearly unimodal (the level sets are convex sets). Looking at the function along a particular move direction h, this means that the function is unimodal and goes to infinity as the step size α approaches α_{max}, the maximum value of α for which $y^k + \alpha r h$ stays in the simplex S (see figure 1). Of course, $\alpha_{max} \geq 1$, and it is easy to verify that α_{max} can be as large as $(n - 1)$ since the distance between the center and any extreme point of the simplex is $\sqrt{\frac{n-1}{n}}$.

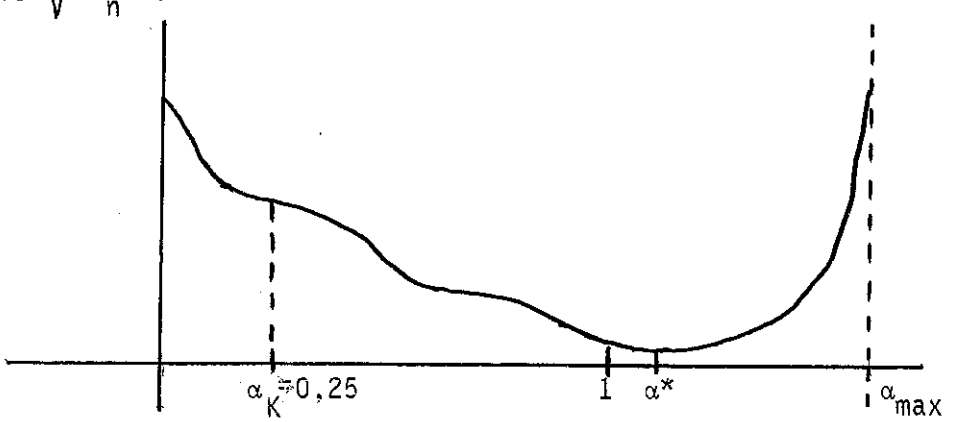

Figure 1

Now, we wish to approximate α^* where the minimum of f along h is attained, by computing the value of f at a number of points, but since we want to perform all computations in polynomial time, we introduce the following restrictions :

- A fixed upper bound M on the number of potential function evaluations is imposed ;

- All arithmetic computations are carried out using numbers representable by a fixed number of digits (in practice, the size of the words associated with the computer) and assuming that the output of each arithmetic computation can be given some confidence interval (depending on the accuracy of the built-in arithmetic functions) ; so, if an arithmetic operation Op is applied to an argument u, $\lceil Op(u) \rceil$ (resp. $\lfloor OP(u) \rfloor$) will denote the largest (resp. the smallest) value in the confidence interval ; since all computations are carried out on numbers with constant size representations, both values $\lceil Op(u) \rceil$ and $\lfloor Op(u) \rfloor$ can be computed in constant time $O(1)$.

We can then state :

Lemma 1 : For f defined by (1), obtaining $\lceil f(x) \rceil$ and $\lfloor f(x) \rfloor$ for any input x can be done in polynomial time $O(n)$.

Proof : Just observe that :

$$\lceil f(x) \rceil = \lceil n \lceil \log c^T \bar{x} \rceil \rceil - \sum_{i=1}^{n} \lfloor \log x_i \rfloor \quad \text{and}$$

$$\lfloor f(x) \rfloor = \lfloor n \lfloor \log c^T x \rfloor \rfloor - \sum_{i=1}^{n} \lceil \log x_i \rceil \quad \square$$

We know that at the current iterations the original problem is transformed via a projective transformation, then the potential function is transformed under the form :

$$g(y) = \sum_{i=1}^{n} \log \frac{c^T D_a y}{y_i}, \qquad (2)$$

where $a = x^k$ and D_a is the diagonal matrix whose diagonal elements are the components of vector a. As shown in KARMARKAR (1984) when x and y correspond through the projective transformation $y \xrightarrow{T_a} x = \frac{D_a y}{e^T D_a y}$ then $f(x)$ and $g(y)$ only differ by a constant factor, so any reduction in $g(y)$ results in the same reduction in $f(x)$.

Obviously, many possible search procedures using the unimodality property such as golden section search, Fibonacci search, etc. could be used

(see for instance MINOUX (1983, chapter 3)). However, here, since the directional optimum is, in most cases, located near the boundary of the simplex (resulting in α values close to α_{max}) even simpler schemes can be used, such as the one proposed below (y^k and h denote the current point and the search direction in transformed space).

Procedure 1

(i) Choose $M > 1$ (maximum number of function evaluations)
set $\alpha := 0.25$,
$t := 1$,
compute $\lceil z \rceil := \lceil g(y^k + \alpha\ r\ h) \rceil$ and
$\lfloor z \rfloor := \lfloor g(y^k + \alpha\ r\ h \rfloor$;

(ii) (current step) If $t \geq M$ goto (iii), otherwise
$\alpha' := \alpha + \frac{\alpha_{max} - \alpha}{2}$,
compute $\lceil z' \rceil := \lceil g(y^k + \alpha'\ r\ h) \rceil$ and
$\lfloor z' \rfloor := \lfloor g(y^k + \alpha'\ r\ h \rfloor$
if $\lceil z' \rceil > \lfloor z \rfloor$ (no guaranteed improvement on the potential function) goto (iii),
otherwise $\alpha := \alpha'$
$\lceil z \rceil := \lceil z' \rceil$
$\lfloor z \rfloor : \lfloor z' \rfloor$
$t := t + 1$
return to (ii) ;

(iii) The best step size found is α.
Output $y^{k+1} = y^k + \alpha\ r\ h$ (the new current point in the transformed space).

The above search scheme enjoys the following properties :

Property 1 : If y^{k+1} is the output of Procedure 1, then $g(y^{k+1}) \leq g(y^k + 0.25\ r\ h)$, in other words the new point obtained gives the potential function a value which can only be better than the Karmarkar step ($\alpha_k = 0.25$).

Property 2 : When a constant maximum number of function evaluations is imposed then the worst case time complexity of Procedure 1 is $O(n)$.

In fact we show in next section that any search procedure enjoying properties 1 and 2 would work.

3. COMPLEXITY ANALYSIS

It is worth observing that property 2 is not only of theoretical interest. In fact it should be compared with the complexity of the direction finding problem at each iteration which is $O(n^{2.5})$ per step on the average in the refined version of the algorithm (the simpler version gives $O(n^3)$ per step).

Since in practice, very small values of M (in the range 5 to 10) will be sufficient to get a pretty good approximation to the minimum, we see that the work involved by the one dimensional searching will stay in all cases negligible as compared with the global computational effort for one iteration. This can be formalized by the property below.

Property 3 : The complexity of one Karmarkar's iteration does not change when embedding any one dimensional search procedure satisfying property 2.

We are not ready to state the main result.

Theorem : The worst case complexity of Karmarkar's algorithm with any embedded one dimensional search procedure satisfying properties 1 and 2 is the same as that of the original algorithm.

Proof : In view of property 3 we have only to show that the number of iterations to achieve any given reduction factor (2^{-f} say) in the objective function is the same (in the worst case sense) for the two algorithms.

Following Karmarkar's proof, taking $\alpha = \alpha_k = 0.25$ we get :

$g(y^k + \alpha_k \, r \, h) \leq g(y^k) - \delta$, where

δ is the quantity appearing in Theorem 4 of KARMARKAR (1984) ($\delta = \frac{1}{8}$).

Now applying a one-dimensional search procedure satisfying property 1 we get a y^{k+1} such that

$$g(y^{k+1}) \leq g(y^k + \alpha_k \, r \, h) \leq g(y^k) - \delta,$$

and since the original potential function f and the transformed potential function g differ only by a constant term, we get

$$f(x^{k+1}) \leq f(x^k) - \delta, \tag{3}$$

where x^{k+1} is the image of y^{k+1} in the original space.

Of course in general, due to the one-dimensional search procedure $f(x^{k+1}) << f(x^k) - \delta$ will be achieved, but (3) states that, in the worst case, one obtains at least the same decrease in the potential function as in the fixed step method with $\alpha_k = 0.25$.

Now, in this worst case situation, we can apply exactly the same proof as that of theorem 1 in KARMARKAR (1984), leading to the same upper bound $O[n(f + \log n)]$ on the number of steps □

As a final remark, since the work per iteration of Karmarkar's method is $O(n^{2.5})$ on the average, we observe that we could even afford much more accurate line searches at each iteration : indeed, as long as the number of potential function evaluations does not exceed $O(n^{1.5})$ the complexity of the whole algorithm remains unaffected. This is because, contrary to what occurs in most mathematical programming algorithms, one function evaluation here is very cheap to obtain when compared with the other computations to be performed.

4. NUMERICAL RESULTS

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

$$\text{Minimize } c^T x$$
$$\begin{cases} A\,x \leq IAS \\ I\,x \leq IBP \\ x \geq 0 \end{cases}$$

with

x, c^t and IBP (bounds) : (N x 1) integer vectors.
IAS : (Right hand side) : (M x 1) integer vector.
A : (M x N) integer matrix ; each of its columns contain 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 chosen randomly between 1 and 10.

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 and 50 rows and 150 columns. No sparsity patterns were imposed on the matrix A. The modified algorithm treats the primal and dual forms simultaneously.

Two linear programming codes were used : PRIFLO and MPSX/MIP plus one implementing Karmarkar's method coded in FORTRAN and tested on IBM 3090 in double precision arithmetic. The Cholesky decomposition was implemented and used in the calculus of the projection of the fector Dc. Table I contains the results for the three codes, the number of iterations and the CPU time, where convergence was attained when λ (artificial variable) $< 10^{-8}$ and the step size $\alpha = .95$.

Note that the CPU time of Karmarkar's code is markedly larger than the one obtained by the two other codes. The reason for this is essentially the explicit computation of the orthogonal projection at each iteration.

Tables II and III contain the results of the algorithm of Karmarkar on the first two problems for various fixed values of α (from $\alpha = .25$ up to $\alpha = .99$).

Larger values of α markedly decrease the number of iterations (from 293 iterations for $\alpha = .25$ to 74 iterations for $\alpha = .99$ for the first test problem) and the CPU time was divided by a factor 8. The value of α suggested by Karmarkar ($\alpha = .25$) gives the maximum number of iterations.

Tables 4, 5, 6, 7 and 8 contain the result of the modified algorithm using the dichotomy method described in the previous sections. The potential function is computed 12 times at each iteration to obtain the value of α which corresponds to the best approximation to the minimum of the potential function. Two important facts are observed.

The first is the number of iterations which is about 12 for all the problems, the modified algorithm is 10 timesfaster than the original method with $\alpha = .95$.

The second prominent feature is that the value of α is always significantly greater than 1 and can reach $(n - 1)$ (i.e. the ratio between the radius of the circumscribed sphere and the inscribed sphere).

Finally, the modified algorithm appears stable and its efficiency could be further improved if the sparsity of the matrix was exploited and if the bounds were treated separately like in the simplex method. Tests of those remain for future implementations.

TABLE 1 : Computational results of the code PRIFLO of EDF (Electricité de France), MPSX and Karmarkar's algorithm
All times in CPU seconds on an IBM 3090

	PRIFLO			MPSX			KARMARKAR $\alpha = .95$ EPS $= 10^{-8}$		
Dimension	Objective	Iterations	CPU time	Objective	Iterations	CPU time	Objective	Iterations	CPU time
10-30	412	16	.0083	412	15	.72	412 411.998	78	7.6
20-60	1019	42	.0114	1019	42	.74	1019 1018.997	104	89.68
30-90	1516	59	.0166	1516	64	.77	1516 1515.997	123	575.85
40-120	2108	80	0.0210	2108	84	.78	2108 2107.996	134	1825.08
50-150	2619	95	0.0276	2619	112	.81	2619 2618.998	142	3936.21

TABLES II, III : Iteration counts, CPU time and different stepsizes for Karmarkar's algorithm on two selected test problems

TEST 1

Dimension	Objective	Iterations	CPU time	Alpha max
10-30	412	293	28.7	.25
10-30	412	147	14.13	.5
10-30	412	98	9.5	.75
10-30	412	78	7.6	.95
10-30	412	74	7.3	.99

TEST 2

20-60	1019	393	349.75	.25
20-60	1019	197	175.25	.5
20-60	1019	131	115.85	.75
20-60	1019	104	89.68	.95
20-60	1019	100	86.4	.99

TABLES IV, V, VI, VII, VII : Computational results with the dichotomy method applied to Karmarkar's algorithm

TABLE IV, TEST 1

Dimension	Iterations	Objective	CPU time	Alpha max	Optimal value of α
10-30	1	140	.103	3.56	3.46
	2	233	"	5.16	5.01
	3	322	"	5.23	5.08
	4	368	"	4.43	4.31
	5	400	"	6.01	5.83
	6	409	"	5.74	5.57
	7	411.3	"	6.45	6.26
	8	411.8	"	5.01	4.87
	9	411.9	"	5.92	5.57
	10	411.99	"	5.63	5.29
	11	411.99	"	5.23	4.92
	12	411.999	"	6.03	5.67
			1.24		

TABLE V, TEST 2

20-60	1	374	.893	5.05	4.90
	2	585	"	7.15	6.93
	3	775	"	6.3	6.11
	4	919	"	7.12	6.91
	5	989	"	8.08	7.84
	6	1009	"	6.94	6.73
	7	1017	"	10.54	10.22
	8	1018	"	5.79	5.62
	9	1018.9	"	8.59	8.07
	10	1018.97	"	7.74	7.71
	11	1018.99	"	7.13	6.92
	12	1018.998	"	8.39	8.14
			10.72		

TABLE VI, TEST 3

Dimension	Iterations	Objective	CPU time	Alpha max	Optimal value of α
30-90	1	481	4.75	5.37	5.21
	2	795	"	8.33	8.06
	3	1049	"	6.43	6.23
	4	1286	"	8.08	7.84
	5	1443	"	10.08	9.77
	6	1472	"	4.48	4.35
	7	1507	"	11.47	11.12
	8	1513	"	8.57	8.32
	9	1515.8	"	12.92	12.52
	10	1515.83	"	6.24	6.05
	11	1515.96	"	10.28	9.96
	12	1515.99	"	9.49	8.91
	13	1515.996	"	9.16	8.88
			61.65		

TABLE VII, TEST 4

40-120	1	612	13.61	5.39	5.23
	2	902	"	7.24	7.01
	3	1418	"	10.15	9.84
	4	1744	"	8.46	8.2
	5	1969	"	10.28	9.97
	6	2047	"	8.1	7.86
	7	2095	"	13.43	13.02
	8	2103	"	8.85	8.57
	9	2107.6	"	17.39	16.86
	10	2107.8	"	4.76	4.62
	11	2107.9	"	8.24	7.99
	12	2107.99	"	14.98	14.52
			163.34		

TABLE VIII, TEST 5

Dimension	Iterations	Objective	CPU time	Alpha max	Optimal value of α
50-150	1	709	27.4	5.86	5.68
	2	1153	"	9.54	9.24
	3	1751	"	10.47	10.15
	4	2148	"	9.02	8.74
	5	2443	"	11.5	11.15
	6	2536	"	8.61	8.34
	7	2582	"	9.18	8.9
	8	2614	"	17.26	16.73
	9	2617	"	7.99	7.75
	10	2618.8	"	18.91	18.33
	11	2618.88	"	4.91	4.77
	12	2618.97	"	13.5	13.09
	13	2618.99	"	13.4	12.99
			356.23		

ACKNOWLEDGEMENTS

P. TOLLA is gratefully acknowledged for careful reading of a first draft of the manuscript.

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

BIBLIOGRAPHY

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

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., EA K. (1986) : "Une adaptation de l'algorithme de Karmarkar aux problèmes de flot à coût minimal", à paraître (DER EDF).

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) : "On the complexity of linear programming", Research Report, IBM Almaden Research Center, San Jose, California.

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

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

NICKELS W., RODDER W., XU L., ZIMMERMANN H.J. (1985) : "Intelligent gradient search in linear programming", European Journal of Operational Research 22, 293_303.

SCHRIJVER A. (1985) : "The new linear programming method of Karmarkar", CWI Newsletters.

SHANNO F. (1985) : "Computing Karmarkar projections quickly", Working Paper, Graduate School of Administration, University of California.

TOLLA P. (1987) : "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 (à paraître).

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

YE Y., CHIU S.S. (1986) : "Recovering the shadow price in projection method of linear programming", Preliminary Draft, Engineering Economic Systems Department, Stanford University.