On First Experiences with the Implementation of a Newton Based Linear Programming Approach

by

A. Wanka, Cologne

Abstract

This paper presents the implementation of an exterior point linear programming approach introduced by Betke [3, 4]. In every iteration of this algorithm Newton's method is used in order to determine nearest points of two convex sets. The method is simple to implement, fully exploits sparsity and at the presence of rounding errors it achieves high precision and stability.

Beside the nice geometric proceeding a particular interest in this method is derived from the following observation: In Karmarkar's as well as in this algorithm one has to solve a linear equation system of the form $AD^2A^Ty = b$. The solution of this system is the most time consuming part in linear programs. While in Karmarkar's method the entries of the diagonal Matrix D are the coordinates of the iteration point, the diagonal entries are either 0 or 1 in our method. Hence throughout this paper D is a purely combinatorial matrix which yields to reasonable numbers of rank one updates of the corresponding Cholesky factor of AD^2A^T. Moreover, the Cholesky factors become sparser than in Karmarkar's approach.

Keywords: Linear Programming, Newton's method, Cholesky factorization, rank one updates

1 Introduction

In 1984 Karmarkar proclaimed a new era of linear programming initialized by his new algorithm [13]. Since then a flood of research has taken place, firstly to achieve or to accelerate the announced performance of being 50 times faster than the simplex method and secondly to relax the standard form of linear programs Karmarkar started with. The latter question has been discussed more theoretically in many papers (see e.g. the special issue of Algorithmica Vol.1, No.4 (1986)). So the still open dispute is: Will the Karmarkar algorithm, a variant or some completely different method (see e.g. Renegar [18], Chang & Murty [17]) outperform the established simplex method? Implementations of Adler et al. [1], Gill et al. [9, 10] and Mc Shane et al. [16] demonstrate in a fairly impressive way that their codes of Karmarkar variants are already at least competitive with the simplex method. To this point the works of Karmarkar & Sinha [14] and Todd [21] on the application of Karmarkar's algorithm on special structured linear programs deserve particular attention.

In his original paper Karmarkar considers problems of the form:

$$\begin{aligned} \min c^T x & \\ \text{subject to } Ax = 0 & \\ e^T x = n & \\ x \geq 0 \ , & \end{aligned} \tag{1}$$

where A is an $(m \times n)$-matrix, x, c and e are n-vectors where all coordinates of e are one. Furthermore, he assumes the optimum x^* of (1) to satisfy $c^T x^* = 0$. The rough proceeding of Karmarkar's method from a current feasible point x to the next is as follows:
At first compute the projective transformation

$$y = \frac{nD^{-1}x}{e^T D^{-1} x} \tag{2}$$

where D is a diagonal matrix defined by

$$d_j = x_j, \quad j = 1, \ldots, n \ . \tag{3}$$

A matrix B is defined by

$$\begin{pmatrix} A\ D \\ e^t \end{pmatrix} \tag{4}$$

and the projection p of the objective vector c is determined by

$$p = -[I - B^T (BB^T)^{-1} B] Dc \ . \tag{5}$$

A new point y is then defined by

$$y = e + \delta p \ , \tag{6}$$

where $\delta > 0$ is the steplength parameter, and the new iterate $\bar{x}$ by the inverse projective transformation

$$\bar{x} = \frac{nDy}{e^T Dy} \ . \tag{7}$$

This procedure stops when $c^T x$ is sufficiently small.

Apparently the most time consuming part in Karmarkar's algorithm is the computation of the objective vector's projection by (5) and herein the involved solution of the equality system

$$AD^2 A^T z = d \ . \tag{8}$$

To get this projection step (5) with lesser computational effort, Dennis et. al. [6], Goldfarb et. al. [12], Karmarkar [13] and Shanno [19] have investigated relaxations. Dennis et. al. introduces to that end a variable metric approach, whereas Karmarkar and Shanno apply the following rank one update notion: Let

$$\bar{D}^2 = D^2 + T \ , \tag{9}$$

where $\bar{x}, x$ are successive iteration points. Then

$$\begin{aligned} A\bar{D}^2 A^T &= AD^2 A^T + ATA^T \\ &= AD^2 A^T + \sum_{j=1}^{m} t_j a_j a_j^T \ . \end{aligned} \tag{10}$$

In his original method $t_j \neq 0$ $(j = 1, \ldots, m)$ implying m rank one updates per iteration of the corresponding Cholesky factorization. The basic idea for his modified algorithm is to replace t_j by 0 if t_j is small. Karmarkar proved, using a particular scaling, that the resulting algorithm converges to the optimum and that the complexity is even $\sqrt{n}$ better than the original. Note, however, that the modified algorithm leads to an increase of the number of iterations, since the projection of c is only approximately determined.

In this paper we are now going to report on the implementation of an exterior point algorithm, in which again the solution of (8) is the crucial point. In this approach D is replaced by a purely combinatorial diagonal matrix, which changes from iteration to iteration only in a few entries. Thus the use of the above mentioned rank one updates is more natural and yield in addition to sparse Cholesky factors. The following gives an outline of the paper.

Section 2 describes the algorithm and its implementation. Section 3 surveys the so far obtained numerical results. They can be summarized by the fact that the tested code is almost as fast as the simplex method implementation MINOS 5.1. The speed of MINOS appears to be reachable if we are able to overcome and accelerate some numerically crucial steps.

2 Linear Programming by Minimizing Distances

Every linear programming problem can be restated as

$$\begin{aligned} & \min t \\ \text{subject to } & \begin{pmatrix} A \\ c^T \end{pmatrix} l + \begin{pmatrix} b \\ t \end{pmatrix} \geq 0 \ , \end{aligned} \tag{11}$$

where A is a $(n-1) \times d$-matrix, rank $A = d$, b is a $(n-1)$-vector and c a d-vector with $c \neq 0$. Incorporating c into A and t into b the associated feasibility problem is: Does there exist an $l \in \mathbb{R}^d$ with $Al + b(t) \geq 0$? Betke/Gritzmann [5] suggested to replace this feasibility problem by a more strict one which determines distances as follows:
Let E be the parametric affine plane

$$E(t) := \{x \in \mathbb{R}^n \mid x = Al + b(t) \ , l \in \mathbb{R}^d\} \ , \tag{12}$$

and S the positive orthant

$$S := \{x \in \mathbb{R}^n \mid x \geq 0\} \ . \tag{13}$$

Then for a fixed t the feasibility problem is particularly solved, if you know either that the euclidean distance $d(E(t), S)$ is positive or you know x with $x \in E(t) \cap S$. Betke/Gritzmann noticed that for this problem the ellipsoidal algorithm can be used to achieve a polynomial time method for linear programming. In order to replace the use of the rather theoretical ellipsoidal algorithm by a procedure which works well in practice Betke has suggested the application of Newton's method. It starts with some $x(t) \in E(t)$ and determines the Newton direction with respect to the distance function $d(E(t), S)$.

Theorem 1 *Let t be fixed.*

1. *The sequence of points generated by Newton's algorithm converges to the minimum of $d(E(t), S)$.*

2. *There exists a constant $\varepsilon > 0$ with the following property: If $d(x, S) - d(E(t), S) < \varepsilon$ for $x \in E(t)$, then Newton's algorithm gives the exact solution in at most $n + d$ further steps.*

Proof: Betke [3]

Corollary 2 *Newton's algorithm solves the feasibility problem of linear programming in a finite number of steps.*

Suppose one has a routine F solving the feasibility problem of linear programs, it is well known that with binary search F solves the complete linear program, at least for rational input data. An other fairly general procedure might be to start with a lower bound of the objective function value. If the corresponding feasibility problem returns the result infeasible (or equivalently $d(E(t), S) > 0$) then increase the lower bound of the objective function appropriately and continue this proceeding until $d(E(t), S) = 0$. In fact, with the above notation the linear program (1) is equivalent to the computation of $t^* = \min\{t \in \mathbb{R}, d(E(t), S) \leq 0\}$. For the determination of a suitable increase of the lower bound the ensuing algorithm applies the this time one-dimensional Newton method. Opposed to binary search this proceeding leads to drastical reductions in the number of phase II iterations.

Theorem 3 *$d(E(t), S)$ is a in t decreasing, nonnegative convex function. Furthermore there exists a number $\rho > 0$, such that $d(t)$ is linear in $I = [t^* - 2^{-\rho L}, t^*]$ where L is the length of the binary encoding of the input.*

Proof: Betke [4]

Corollary 4 *Newton's algorithm solves linear programming problems in a finite number of steps.*

In order to describe the new algorithm in more formal terms, it is appropriate to express the euclidean distance of $x = Al + b(t) \in E(t)$ and the positive orthand S by

$$d(l, t) := d(x, S) =: (x^T D(l, t) x)^{1/2} \tag{14}$$

where $D(l,t)$ is a diagonal matrix with $d_j := \begin{cases} 1 & \text{if } x_j < 0 \\ 0 & \text{else} \end{cases}$. The algorithm can now be stated as follows:

Algorithm

$$i, j = 0 \tag{15}$$

CHOOSE a lower bound t_0 for the optimum (16)

CHOOSE $l_0 \in \mathbb{R}^d$ (17)

WHILE $\nabla d(l_i, t_j) = 2A^T D(l_i, t_j)(Al_i + b(t_j)) \neq 0$ (18)

  COMPUTE p_i by: $\nabla d(l_i, t_j) + A^T D(l_i, t_j) A p_i = 0$ (19)

  DETERMINE d_i by: $d(l_i + \delta_i p_i) = \min_\delta d(l_i + \delta_i p_i)$ (20)

  $l_{i+1} = l_i + \delta_i p_i \quad , i = i + 1$ (21)

  END

IF $(j = 0$ and $d(l^*, t_0) = 0) \;\Rightarrow\;$ 'problem unbounded' (22)

IF $d'(l^*, t_j) := \dfrac{e_n^T D(l^*, t_j) x}{d(l^*, t_j)} = 0 \;\Rightarrow\;$ 'problem infeasible' (23)

LET $t_{j+1} := t_j - \dfrac{d(l^*, t_j)}{d'(l^*, t_j)} \quad , j = j + 1$ (24)

COMPUTE $d(l^*, t_j), d'(l^*, t_j)$ as described above (25)

IF $d(l^*, t_j) = 0 \;\Rightarrow\;$ 'optimal' (26)

GOTO (18)

In all LP-methods the speed of each iteration is dominated by the time to solve an equation system. Therefore, there are roughly speaking two strategies to achieve good performances of LP-codes, namely, to use relatively small matrices in this equation system as the simplex method does with using bases or to work with the large matrix of all restrictions in order to achieve low iteration numbers. Karmarkar's as well as the described method follow the second strategy. Fortunately, ready tools to solve (19) are available using the Cholesky factorization

$$A^T D A = \bar{R}\bar{R}^T \tag{27}$$

with $\bar{R}$ an upper triangular matrix. However, if A contains dense columns, the factor $\bar{R}$ becomes incredibly dense and all advantages of sparsity are lost. Furthermore, since D is a degenerated diagonal matrix, $A^T DA$ becomes very ill conditioned yielding an uncomfortable unprecise search direction. As a remedy of that fact one can replace (19) by the equivalent least square problem

$$|D(l_i,t_j)(Al_i+b(t_j))+D(l_i,t_j)ARz| \ . \tag{28}$$

Note that this least square problem includes an appropriately chosen triangular matrix R which accelerates the convergence of the iteratively applied conjugate gradient method. The matrix R, used in (28), is the Cholesky factor of

$$P\tilde{A}^T D\tilde{A}P^T = RR^T \tag{29}$$

where $\tilde{A}$ is obtained from A by deleting dense columns. We call a column dense if it contains more than 50 nonzero entries. The permutation matrix P is computed by the subroutine GENQMD of the SPARSEPAK-A package (see George & Liu [7]). This minimum degree ordering matrix P is applied to achieve a possible sparse preconditioner R. The data structure of the nonzero entries of the Cholesky factor R is computed by SPARSEPAK'S subroutine SMBFCT. If this structure is attained (with $d_j=0$ if row j is dense, $d_j=1$ otherwise), we keep it fixed throughout the run of the algorithm. The values of R, however, changes from iteration to iteration as follows: Let x_i, x_{i+1} be two successive iteration points. By definition the diagonal entry d_j of D changes from 1 to 0 if $(x_i)_j<0$, $(x_{i+1})_j\geq 0$ and d_j changes from 0 to 1 if $(x_i)_j\geq 0$, $(x_{i+1})_j<0$. Thus from iteration i to iteration $i+1$ the Cholesky factor can be determined by the sequence of rank one up and down dates:

$$R_{i+1}R_{i+1}^T = R_iR_i^T \pm a_{l_1}a_{l_1}^T \pm \ldots \pm a_{l_k}a_{l_k}^T \ . \tag{30}$$

These up and down dates can be accomplished by the wellknown Givens rotations (see [11, 19, 20]). Remember that within the introduction we have shown that (30) is valid also for Karmarkar's algorithm. In Karmarkar's original version l_k equals d. Also in his modified version l_k is on the average $d/2$ and therefore the effort to determine R_{i+1} from R_i are not negligible. You might suspect, that also in our algorithm the number of rank one up or down dates (or equivalently the number of sign switches between successive iteration points) may be large. In general, this might be true, especially in the first iterations. The elaboration of the Newton step reveals, however, that we are well advised to restrict this number of sign switches. Notice, that the affine plane $E(t)$ is divided by the coordinate hyperplanes into many regions. These regions are characterized by the sign pattern all included points have. The major problem of finding a point x of $E(t)$ to be the nearest to the positive orthand is now, to detect that region of E having the sign pattern of x. Taking into account that for the computation of the search direction in (19) we assume D to correspond to the sign pattern of the current iteration point, then it becomes apparent that the search direction is determined by a more locally driven strategy. Hence, it is suitable to restrict the steplength not only by (20) but also by a more combinatorial criterion, namely, to bound the maximal number of sign switches from x_i to $x_i + \delta z$ (our suggestion here is the number 3).

Nevertheless, the determination of the steplength δ is fairly easy and accomplished with almost no computational effort, since δ of (20) is $\frac{-z^T D x_i}{z^T D z}$. As an alternative approach for the steplength we have implemented the control of the deviation of the gradient at those points where $x_i + \delta z$ changes a sign. But in general this did not accelerate the algorithm.

A third step, beside (19) and (20), that we didn't implement straight forwardly is (24) and concerns the scaling of the objective function row after every phase II step. Remember that the objective function row looks like

$$y = c_T - t \geq 0 \ . \tag{31}$$

Suppose that for a fixed t the point $\bar{x}$ of E , which is closest to S, has been determined and that $y_{old} := c_T \bar{x} - t$. Note that after a phase II step (23)

$$y := y_{new} \geq 0 \text{ and that at the optimum } y = 0 \ .$$

Our experiments have shown that a scaling of (31) such that

$$\frac{y_{old}}{y_{new}} = -\text{factor} \tag{32}$$

yields more stable improvements of the increase of the objective function value by every step (24). We achieved good results with a factor around 12.

3 Numerical Results

In this section we are going to summarize the performance of the described Newton method on problems we have picked up from a general LP test set. This set comprises 53 input data which are available via Netlib [1]. We have

[1]for details send electronic mail to netlib @ anl-mcs or to research! netlib saying "send index from LP/data".

used standard techniques of linear programming to bring all of these input data sets into the form (11).

A more detailed description of the problems dealt with are given by table 1. The data sets are ordered by increasing numbers of nonzeros. Our computational results were achieved under the following environment: We used double precision on an IBM 4361 and the source code was compiled with the IBM Fortran 77 compiler VS Fortran Version 2 using NOSDUMP, NOSYM and OPT(3). All times given are for a complete run, including data input and output. We compare our result with the code of MINOS 5.1 (August 1987) which is an efficient Fortran code of the simplex method. The default values of parameters were used throughout. As far as available we also confront our results with those of Adler et al. (affine version of Karmarkar's algorithm, IBM 3081-K, MINOS 4.0: default parameters), Gill et al. (Newton's barrier method, DEC VAXstation II, MINOS 5.2: scale option 2, partial pricing 10) and Mc Shane et al. (primal-dual barrier method, VAX 8650, MINOS 5.0: default parameters).

Problem	*Rows*	*Columns*	*Nonzeroes*	*Optimum*
Afiro	28	32	88	-4.647E+02
Adlittle	57	97	465	2.254E+05
Sc 205	206	203	552	-5.220E+01
Scagr7	130	140	553	2.331E+06
Share2b	97	79	730	-4.157E+02
Recipe	92	180	752	-2.666E+02
Vtpbase	199	203	914	1.298E+05
Share1b	118	225	1182	-7.658E+04
Bore3d	234	315	1525	1.373E+03
Scorpion	389	358	1744	1.878E+03
Capri	272	353	1786	2.690E+03
Scagr25	472	500	2029	-1.475E+07
Sctab1	301	480	2052	1.412E+03
Brandy	221	249	2150	1.518E+03
Israel	175	142	2358	-8.966E+05
Etamacro	401	688	2489	-7.557E+02
Grow7	141	301	2633	-4.778E+07
Bandm	306	472	2659	-1.586E+02
E226	224	282	2767	-1.875E+01
Scsd1	78	760	3148	8.666E+00
Beaconfd	174	262	3476	3.359E+04
Stair	357	467	3857	-2.512E+02
Scrs8	491	1169	4029	9.042E+02
Scfxm2	661	914	5229	3.666E+04
Scsd6	148	1350	5666	5.050E+01
Ship04s	403	1458	5810	1.798E+06
Scfxm3	991	1371	7846	5.490E+04
Sctab2	1091	1880	8124	1.724E+03
Grow22	441	946	8318	-1.608E+08
Ship04l	403	2118	8415	1.793E+06
Ship08s	779	2387	9501	1.920E+06

Table 1

Problem	*Iterations*		*MINOS / Running Times*			
Name	Minos	Newton	McShane	Adler	Gill	Newton
Afiro	8	19	1.67	0.7	0.2	1.1
Adlittle	97	60	2.50	1.8	0.5	0.6
Sc 205	131	39	2.29	1.8	0.8	0.9
Scagr7	92	52	1.14	1.4	0.7	0.6
Share2b	117	65	1.66	0.9	0.6	0.5
Recipe	33	16	-	-	0.6	0.5
Vtpbase	286	256	-	-	0.2	0.2
Share1b	284	125	0.81	2.1	0.9	0.4
Bore3d	114	214	-	-	0.4	0.1
Scorpion	139	70	2.22	2.3	0.4	0.5
Capri	295	281	-	-	0.3	0.1
Scagr25	92	115	3.16	7.0	2.2	0.1
Sctap1	375	81	2.97	1.7	0.8	0.7
Brandy	323	109	1.22	0.9	1.2	0.4
Israel	327	98	0.20	0.4	0.6	0.1
Etamacro	604	919	-	-	0.4	0.03
Grow7	151	60	-	-	0.9	0.3
Bandm	463	297	1.67	1.8	1.1	0.3
E226	686	153	2.05	1.6	0.9	0.5
Scsd1	220	65	5.39	2.7	1.1	0.4
Beaconfd	87	55	0.38	0.3	0.2	0.3
Stair	482	289	-	-	0.6	0.1
Scrs8	933	275	1.46	1.9	1.1	0.4
Scfxm2	828	387	2.60	1.7	1.7	0.2
Scsd6	550	71	8.01	4.5	1.8	0.6
Ship04s	148	117	2.94	2.2	0.3	0.3
Scfxm3	1284	471	3.70	2.6	2.2	0.3
Sctap2	1726	60	4.77	2.6	1.1	1.4
Grow22	880	81	-	-	2.2	0.7
Ship04l	226	119	3.11	2.5	0.4	0.3
Ship08s	231	162	4.59	4.4	0.6	0.3

Table 2

Table 2 collects the results of the different algorithms. It shows the iteration numbers as well as the respective ratios of the running times of the

MINOS code and the running times of the corresponding Karmarkar version. Here we suppress the surprising fact concerning the iteration numbers of each Karmarkar method, namely, that their iteration numbers are almost fixed in the range of 30 to 50 over all dimensions and sizes of problems. We would like to mention, that, in contrast to others, we haven't used a pre- or post solve phase to identify redundant constraints or variables to be 0. We have definitely no experience with such procedures and therefore we can't say anything about what we would gain if we had incorporated appropriate routines. What we can say about Newton's method is, that it is simple to implement, fully exploits sparsity and at the presence of rounding errors it achieves high precision and stability.

We interprete table 2 as follows:

With respect to the iteration numbers the Newton method behaves far better than the Simplex method but worse than Karmarkar's method (almost constantly 30 - 50 iterations for every problem). The reason why we end up with higher iteration numbers as Adler et. al. or Mc Shane et. al. has already been mentioned in the last section. In the Newton algorithm one has to find the region on E with the sign pattern of the point having minimal distance to the positive orthand. Hence, the distance minimization problem includes a combinatorial problem that makes it hard to compute globally good search directions. On the other side, Karmarkar's algorithm is restricted to feasible points, so that his search directions do not depend on a combinatorial property. We conclude that our savings of faster updates of the Cholesky factors in the Newton method are so far not large enough to compensate the cost caused by higher numbers of iterations we end up with.

As a matter of fact the current implementation of the Newton algorithm is in general not faster than Karmarkar's algorithm or the MINOS code. Some facts indicate, however, that a substantial improvement of the algorithm is possible. We will mention four facts which seem to be of particular importance.

For the input datas of Beaconfd, Scagr 25, Ship 04S, Ship 04L and Ship 08L the preprocessing already lasts as long as MINOS needs to solve the complete problem. But this preprocessing comprises only the reading of the input and the accomplishing of the subroutines GENQMD and SMBFCT to compute the data structure of a possible sparse Cholesky factor of $A^T DA$. Comparing this with Mc Shane et.al.'s results suggest to replace the SPARSEPAK routines by the corresponding ones of SMPAK (Scientific Comuting Associates, 1985).

The deletion of dense rows from A improves the speed of GENQMD dramatically and we obtained several good performance results. On the bad side, for the Israel input data we got the effect, from reducing the dense parameter to 35, that the running time was a factor 10 higher than before. So it seems to be worthwhile to investigate the question of finding better permutation matrices in less time.

Beside the evaluation of the Cholesky factors the performance of the routine used for solving the least square problem (28) is decisive for the speed of Newton's algorithm. Very roughly speaking, half of the obtained running times are due to that step (if Newton's algorithm worked extremely bad, this share is even higher). Perhaps the replacement of the used conjugate gradient method by a more robust one can provide remedy. To that point, the recently upcomming normal equation approach (see Gill et. al. [10], Mc Shane et. al. [16]) using Schur complements might decisively improve the performance.

The use of a more efficient dynamic data structure for the Cholesky factor R may also accelerate the speed of the algorithm. Recall that one point that attracts Newton's method is, that R is sparser than in Karmarkar's approach (some diagonal elements of D are 0). By the reason of having a simple and quickly changeable computer code, we didn't use a dynamic data structure in this preliminary version.

4 Concluding Remarks

The described method is an algorithm for solving linear programming problems. The running times achieved by a preliminary implementation indicates , that MINOS or Karmarkar algorithm results are in a reachable distance. Bearing in mind, that by the natural and favourable possibility of updating the involved Cholesky factor the Newton algorithm might soon become a serious competitor to MINOS. The cruical step seems to be whether it will be possible to further reduce the number of iterations. With respect to precision, at the presence of rounding errors the obtained accuracy was at all problems at least 8 digits.

The polynomiality of this algorithm is still an open question. The only results we have proved are Corollaries 2 and 4 and that (24) is performed at most $O(L)$ times where L is the binary encoding of the input. Regarding the latter fact, first it is shown with standard techniques that $|d(t_0)|$, $|d'(t^0)|$, $|l|$ and $|u|$ are bounded from above by $O(2^L)$ where $l(u)$ is the lower (upper) bound for the objective function value. Furthermore $|d(t^* - \varepsilon|$ (see Theorem 3) is bounded from below by $O(2^{-L})$. Then it is fairly obvious to see that in every step either the range of d or the range of d' or $|u-l|$ decreases by a fixed factor which proves the claim. Beside some clues that $O(L)$ may be replaced by $O(n)$ the cruical question for proving polynomiality of the Newton algorithm is: Can the number of Newton steps (20) within the feasibility phase be bounded by a polynom in L ?

References

[1] I. Adler, M. G. C. Resende and G. Veiga, *An Implementation of Karmarkar's Algorithm for Linear Programming*, University of California, Berkeley, 1986.

[2] I. Adler and R. C. Monteiro, *An $O(n^3L)$ Primal-Dual Interior Point Algorithm for Linear Programming*, University of California, Berkeley, 1987.

[3] U. Betke, *An Iterative Algorithm for LP with Finite Termination Property*, to appear in Zeitschrift für Operations Research.

[4] U. Betke, *Linear Programming by Minimizing Distances*, to appear in Zeitschrift für Operations Research.

[5] U. Betke and P. Gritzmann, *Projection Algorithms for Linear Programming*, to appear in Algorithmica.

[6] J. E. Dennis, A, M, Morshedi and K. Turner, *A Variable-Metric Variant of the Karmarkar Algorithm for Linear Programming*, Math. Prog. 39 (1987), 1-20

[7] A. George, J. W. H. Liu, *Computer Solutions of Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, 1981.

[8] G. de Ghellinck and J.-Ph. Vial, *An Extension of Karmarkar's Algorithm for Solving a System of Linear Homogeneous Equations on the Simplex*, Math. Prog. 39 (1987), 79-92.

[9] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin and M. H. Wright, *On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method*, Math. Prog. 36 (1986), 183-209.

[10] P. E. Gill, A. Marxen, W. Murray, M. A. Saunders and M. H. Wright, *Newton Barrier Methods for LP: Some Inplementation Aspects*, Paper presented at the ORSA/TIMS meeting, Washington, April 1988.

[11] G. H. Golub, Ch. F. van Loan, *Matrix*, North Oxford Academic Publishing 1983.

[12] D. Goldfarb and S. Mehrota, *A Relaxed Version of Karmarkar's Method*, Math. Prog. 40 (1988), 289-316.

[13] N. Karmarkar, *A New Polynomial-Time Algorithm for Linear Programming*, Combinatorica 4 (1984), 373-395.

[14] N. Karmarkar and L. P. Sinha, *Application of Karmarkar's Algorithm to Overseas Telecommunication Facilities Planning* , Paper presented at the XII. International Symposium on Mathematical Programming, Boston 1985.

[15] I. J. Lustig, *An Analysis of an Available Set of Linear Programming*, Technical Report SOL 87-11, Stanford, August 1987.

[16] K. A. Mc Shane, C. L. Monma, D. Shanno, *An Implementation of a Primal-Dual Interior Point Method for Linear Programming*, Working Paper, March 1988.

[17] K. G. Murty, Soo Y. Chang, *The Steepes Descent Gravitational Method for Linear Programming*, Working Paper, Ann Arbor, May 1988.

[18] J. Renegar, *A Polynomial Time Algorithm, based on Newton's Method, for Linear Programming*, Math. Prog. 40 (1988), 59-94.

[19] D. F. Shanno, *Computing Karmarkar's Projection Quickly*, Math. Prog. 41 (1988), 61-72.

[20] G. W. Steward, *The Effect of Rounding Errors on an Algorithm for Downdating a Cholesky Factorization*, J. Inst. Maths. Applics. (1979) 23, 203-213.

[21] M. J. Todd, *Exploiting Special Structure in Karmarkar's Linear Programming Problem*, Math. Prog. 41 (1988), 97-114.

[22] P. Vaidya, *An Algorithm for Linear Programming which requires $O((m+n)n^2 + (m+n)^{1.5}n)L)$ arithmetic operations*, AT&T Bell Laboratories, Murray Hill (1987).

Address:
Mathematisches Institut
der Universität zu Köln
Weyertal 86-90

D-5000 Köln 41
Germany