A SIMPLE PRESENTATION OF KARMARKAR'S ALGORITHM

CLOVIS C. GONZAGA*

Abstract. We describe a very simple statement of Karmarkar's algorithm, making clear the mathematical tools used by the method. This allows a short proof of polynomial convergence with much geometrical insight.

1. Introduction. Karmarkar's algorithm [9] appeared in 1984 and triggered a revolution in the field of Optimization. It is an algorithm for solving the most important problem in the field, the linear programming problem:

$$\begin{array}{rrcl} \text{minimize} & c'x & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0, \end{array} \tag{1.1}$$

where $c \in I\!R^n$, $b \in I\!R^m$, A is an $m \times n$ full-rank matrix, $m < n$.

This problem was first solved over fifty years ago by Dantzig [4]. He devised the simplex method, which gave rise to many variants and up to now is the most widely used methodology. Although this is an excellent algorithm in practice, there are examples for which the number of iterations grows exponentially with the space dimension. During the seventies, the study of computational complexity had become very important, and the most desirable theoretical property of an algorithm became its capacity for solving a problem with a number of arithmetical operations growing polynomially with the problem size.

Khachiyan [10] presented a polynomial algorithm for linear programming in 1978, for a definition of problem size which we shall discuss in Section 6, but its performance was very bad in practical problems. So, in 1984, there was a crisis in Optimization: its most important problem had two approaches, one good in practice but with a bad theoretical behavior, and the other with reverse properties. Karmarkar's algorithm brought a great relief, ending this dichotomy. Its computational complexity is polynomial and lower than that of Khachiyan's algorithm, and its performance is comparable to the performance of simplex.

Other facts were important at the moment Karmarkar's method appeared: the cost of computation had become sufficiently low for researchers to have access to powerful workstations and to personal computers; computational linear programming had evolved very much, and large-scale sparse systems could be routinely solved. Karmarkar's algorithm follows an approach which is radically different from the simplex method: while this method is very algebraic, jumping from vertex to vertex of the polyhedron defined by the problem, Karmarkar's algorithm stays in the relative interior of the polyhedron, far from its boundary. The new methodology derived from this approach became known as "interior point methods", and has generated up to now thousands of publications.

In the end of this paper we shall briefly discuss what happened after the diffusion of Karmarkar's result. Our scope here is doing a self-contained presentation of his algorithm. We are actually showing a variant of Karmarkar's method, but it is completely equivalent to the original algorithm up to section 5. This variant is due to several authors, for instance Anstreicher [2], Gonzaga [7].

*Federal University of Santa Catarina, Florianópolis, Brazil. clovis@mtm.ufsc.br.

Karmarkar's algorithm as presented in the original landmark paper [9] is not easy. Several years of experience with it and related methods allow us now to describe its ingredients and prove its properties in a very simple way.

The ingredients are:

- Scaling
- Steepest descent direction
- Potential function
- Homogeneity

The first two items are classical in non-linear programming, and we begin by showing a general model for their application, the scaling-steepest descent (SSD) algorithm. Section 3 describes the potential function and its properties and section 4 describes Karmarkar's algorithm. The polynomial complexity is proven in section 5, and some extensions are discussed in the concluding sections.

Notation: Matrices and vectors are represented respectively by upper and lower case letters. Subindices are used for the components of a vector, the transpose of A is denoted by A' , and we define $e = [1, \ldots, 1]'$. The non-negative orthant of $\mathbb{R}^n$ will be denoted by $\mathbb{R}^n_+$, and its interior (the set of positive vectors) by $\mathbb{R}^n_{++}$.

2. The scaling-steepest descent (SSD) algorithm. Consider the following non-linear programming problem with linear constraints:

$$\begin{array}{rrcl} \text{minimize} & f(x) & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0, \end{array} \tag{2.1}$$

where $f : \mathbb{R}^n_+ \to \mathbb{R}$ is continuously differentiable, $b \in \mathbb{R}^m$, A is an $m \times n$ full-rank matrix, $m < n$.

The SSD algorithm for this problem is simply the steepest descent method with each iteration preceded by a scaling transformation. A scaling is a transformation $x = Dy$, where D is a diagonal matrix with positive diagonal entries. Each iteration starts at a point x^k and the scaling will be $x \to y := X^{-1}x$ where $X = \text{diag}(x_1^k, \ldots, x_n^k)$, so that x^k is mapped into e . The stopping rule is left unspecified in this algorithm model.

ALGORITHM 2.1. *SSD: Suppose given a feasible initial point $x^o > 0$ and $\delta \in (0, 1]$. Set $k := 0$.*

REPEAT

Scaling: $X := \text{diag}(x_1^k, \ldots, x_n^k)$, $\bar{A} := AX$, $\bar{f}(y) := f(Xy)$.

Direction: $h := -P\nabla\bar{f}(e)$, where $P := I - \bar{A}'(\bar{A}\bar{A}')^{-1}\bar{A}$ is the projection matrix into $\text{Null}(\bar{A})$.

Linesearch: Find $\bar{\lambda} \in \text{argmin}\{\bar{f}(e + \lambda h) \mid \lambda \geq 0 \ , \ e + \lambda h \geq 0\}$.

Result: $y^* := e + \delta\bar{\lambda}h$.

Back to the original space: $x^{k+1} := Xy^*$.

$k := k + 1$.

UNTIL convergence.

The constant δ can be made equal to 1 if the line search is known to generate a strictly positive point, as happens for functions that grow indefinitely near the boundary of $\mathbb{R}^n_+$. Otherwise, it must be smaller than 1 . If $f(\cdot)$ is a linear cost function, then this is the affine-scaling algorithm first published by Dikin [5, 6] and independently rediscovered by Vanderbei, Meketon and Freedman [15], by Barnes [3] and others. It is not known to have polynomial behavior, but very good computational results were obtained with δ between 0.95 and 0.99 by Adler et al. [1].

3. Problem formulations: the potential function. We shall consider the linear programming problem in a special formulation:

$$
\begin{array}{rrcl}
\text{minimize} & c'x & & \\
\text{subject to} & Ax & = & 0 \\
& a'x & = & 1 \\
& x & \geq & 0,
\end{array}
\tag{3.1}
$$

where $c, a \in I\!R^n$, $a \geq 0$, $b \in I\!R^m$, A is an $m \times n$ full-rank matrix, $m < n$.

Assumptions:

(i) The feasible set $S = \{x \in I\!R^n_+ \mid Ax = 0, a'x = 1\}$ is bounded.

(ii) The point $x^o = e$ is feasible.

(iii) $c'x$ is not constant on S.

(iv) (A strong assumption) The cost of any optimal solution $\hat{x}$ is $c'\hat{x} = 0$.

The format is very general. The feasible set in problem (2.1) can be put in this format by introducing a new variable w constrained by $w = 1$ and writing $Ax - bw = 0$.

The first assumption can be replaced by the assumption that the optimal set is bounded, and is not too strong. The second one is not bad, since there are standard procedures (see Karmarkar [9]) to generate an initial interior point x^o. A scaling operation takes x^o to e. (iii) discards the trivial case in which all feasible points are optimal. (iv) is very strong, and will be removed in section 6.

Karmarkar's potential function.

Karmarkar defined the following function for positive points in $I\!R^n$:

$$
x \in I\!R^n_{++} \mapsto f(x) \; = \; n \log(c'x) - \sum_{i=1}^n \log x_i \; = \; \log \frac{(c'x)^n}{\prod_{i=1}^n x_i},
\tag{3.2}
$$

and defined the following problem, for a given integer $L > 0$:

$$
\text{Find } x \in S \text{ such that } f(x) < -2nL.
\tag{3.3}
$$

The exact meaning of the condition $f(x) < -2nL$ will be explained in the study of complexity. For the time being, note the following intuitive arguments:

The first part of $f(x)$ is what we want very negative. The second part is known as the logarithmic barrier function:

$$
x \in I\!R^n_{++} \mapsto p(x) \; = \; -\sum_{i=1}^n \log x_i .
$$

It grows indefinitely near the boundary of $I\!R^n_+$ and is bounded from below since S is bounded.

Remember that the value of an optimal solution is zero: the only way to obtain $f(x)$ very negative is to have $c'x$ very small. Then x will be near the boundary of $I\!R^n_+$ and the penalty will be high (greater than 2^{-L}). Then, the condition will imply in $n \log(c'x) < -nL$, or $c'x < e^{-L}$. The number L can be interpreted as the precision to which we want the cost reduced. We will show later that there is a precision (the length of the input data) that allows us to retrieve an exact optimal solution.

Homogeneity of the potential function.

The potential function is zero-degree homogeneous, i.e.:

$$\text{For any } \alpha > 0 \,,\; x > 0 \quad , \quad f(\alpha x) = f(x).$$

This is trivial by substitution in the definition of $f(\cdot)$. We then reach the interesting conclusion that

The constraint $a'x = 1$ can be ignored in problem (3.3).

In fact, define $Q = \{x \in I\!R^n_{++} \mid Ax = 0\}$, and the problem

$$\text{Find } x \in Q \text{ such that } f(x) < -2nL. \tag{3.4}$$

Let x be a solution for this problem. Since $a \geq 0$ and $x > 0$, $a'x > 0$. Then $x/a'x$ is feasible for the original problem, and solves (3.3).

4. Karmarkar's algorithm. Karmarkar's algorithm is the SSD algorithm with $\delta = 1$, applied to problem (3.4).

Let us write the complete algorithm and then make some comments.

ALGORITHM 4.1. *Karmarkar: Assume given $x^o = e$, $x^o \in S$ and $L > 0$. Set $k := 0$.*

REPEAT
- Scaling: $X = \text{diag}(x_1^k, \ldots, x_n^k)$, $\bar{A} := AX$, $\bar{c} := Xc$.
- Direction: $h := -\dfrac{n}{\bar{c}'e} P\bar{c} + e$, where $P = I - \bar{A}'(\bar{A}\bar{A}')^{-1}\bar{A}$ is the projection matrix into $\text{Null}(\bar{A})$.
- Linesearch: $\bar{\lambda} := \text{argmin}\{\bar{f}(e + \lambda h) \mid \lambda \geq 0 \,,\; e + \lambda h > 0\}$.
- Result: $y^* := e + \bar{\lambda}h$.
- Back to the original space: $x^{k+1} := Xy^*$.
- $k := k + 1$.

UNTIL $f(x^k) < -2nL$

$\bar{x} := \dfrac{x^k}{a'x^k}$

Remarks.

(i) The scaled function, defined by $\bar{f}(y) := f(Xy)$ satisfies

$$\bar{f}(y) \;=\; f(Xy) \;=\; n\log(\bar{c}'y) - \sum_{i=1}^{n} \log(y_i x_i^k).$$

It follows that

$$\bar{f}(y) \;=\; n\log(\bar{c}'y) - \sum_{i=1}^{n} \log(y_i) + \text{constant},$$

and the constant term can be ignored in the line search. The gradient is given by

$$\nabla\bar{f}(e) = \frac{n}{\bar{c}'e}\bar{c} - e.$$

Since $e \in \mathrm{Null}(\bar{A})$, $Pe = e$.

(ii) The line search is well defined, since $\bar{f}(\cdot)$ grows indefinitely near the boundary of the orthant (except in the ignored lucky case of hitting an optimal solution). There is no need for a precise line search. In fact, the convergence proof below is valid for any procedure capable of producing a solution not worse than $\lambda = 0.2/\|h\|$.

(iii) The sequence (x^k) can be made feasible for the original problem by introducing the instruction

$$x^k := x^k / a'x^k$$

after each iteration. In this case, the stopping rule can be replaced by "UNTIL $c'x^k < \epsilon$", for a small ϵ .

Analysis of the algorithm.

The rest of this section will be dedicated to proving that each iteration produces

$$f(x^{k+1}) < f(x^k) - 0.1.$$

From this result it follows that in no more than $30nL$ iterations the potential function is reduced by $3nL$. We show in the next section that this reduction is sufficient to detect an optimal solution when the data are all rational and L is related to the input size. We begin by studying properties of the logarithmic function near 1.

The logarithm near 1

In one dimension, using Taylor's formula, we can write for $\lambda > -1$

$$\log(1+\lambda) = \lambda - \frac{\lambda^2}{2(1+\mu)^2} \quad , \mu \text{ between } 0 \text{ and } \lambda.$$

If we now impose $\lambda \geq -0.5$, and consequently $\mu \geq -0.5$,

$$-\frac{\lambda^2}{2(1+\mu)^2} \geq -\frac{\lambda^2}{2(0.5)^2},$$

and we conclude that

$$-\log(1+\lambda) \leq -\lambda + 2\lambda^2 \qquad \text{for } |\lambda| \leq 0.5. \tag{4.1}$$

The barrier function near e **.**

Now in $I\!R^n$, use this relationship to examine $e + \lambda h$, with $\|\lambda h\|_\infty \leq 0.5$:

$$\begin{aligned} p(e+\lambda h) &= -\sum_{i=1}^{n} \log(1+\lambda h_i) \\ &\leq -\sum_{i=1}^{n} \lambda h_i + 2\lambda^2 \sum_{i=1}^{n} h_i^2 \\ &= -\lambda e'h + 2\lambda^2 \|h\|^2. \\ &= \lambda \nabla p(e)'h + 2\lambda^2 \|h\|^2. \end{aligned} \tag{4.2}$$

The potential function near e .

Finally, consider the potential function

$$f(e+\lambda h) = f^1(e+\lambda h) + p(e+\lambda h) \quad \text{, where } f^1(x) = n \log c'x.$$

Since the logarithm function is concave, $f^1(e+\lambda h) \le f^1(e) + \lambda \nabla f(e)'h$. Adding this to (4.2),

$$f(e+\lambda h) \le f(e) + \lambda \nabla f(e)'h + 2\lambda^2 \|h\|^2 \quad \text{for } \|h\|_\infty \le 0.5. \tag{4.3}$$

This means that the linear approximation about e is good, with an error bounded by a quadratic that depends only on the distance from e .

Improvement in one iteration.

Consider now one iteration, after scaling. The line search chooses a point on the line $e + \lambda h$, where $h = -P\nabla \bar{f}(e) = -\beta P\bar{c} + e$, $\beta = n/\bar{c}'e > 0$.

FACT 4.2. *$P\bar{c}$ has a non-positive component.*

Proof. Note initially that for any $y \in \text{Null}(\bar{A})$, $\bar{c}'y = (P\bar{c})'y$. Then at an optimal solution $\hat{y} = X^{-1}\hat{x}$, $(P\bar{c})'\hat{y} = 0$ and $\hat{y} \ge 0$, $\hat{y} \ne 0$. This is impossible if $P\bar{c} > 0$, completing the proof. ◻

FACT 4.3. $\|h\| = \|P\nabla \bar{f}(e)\| \ge 1$.

Proof. From the lemma above, since $P\bar{c}$ has a non-positive component, $h = e - \beta P\bar{c}$ must have a component greater than or equal to 1 . It then follows that $\|h\| \ge 1$, completing the proof. ◻

LEMMA 4.4. *At iteration k ,*

$$f(x^{k+1}) - f(x^k) \;=\; \bar{f}(e + \bar{\lambda} h) - \bar{f}(e) < -0.1.$$

Proof.

For any $\lambda > 0$ such that $\lambda\|h\| \le 0.5$,

$$\bar{f}(e+\bar{\lambda} h) \le \bar{f}(e+\lambda h) \le \bar{f}(e) + \lambda \nabla \bar{f}(e)'h + 2\lambda^2\|h\|^2,$$

from (4.3). But for any vector $v \in I\!R^n$, $v'Pv = (Pv)'Pv$ and then

$$\nabla \bar{f}(e)'h \;=\; -\nabla \bar{f}(e)'P\nabla \bar{f}(e) \;=\; -\|h\|^2.$$

It follows that

$$\bar{f}(e+\bar{\lambda} h) \le \bar{f}(e) - \lambda\|h\|^2 + 2\lambda^2\|h\|^2.$$

Choosing λ such that $\lambda\|h\| = 0.2$,

$$\bar{f}(e+\bar{\lambda} h) \le \bar{f}(e) - 0.2\|h\| + 0.08$$

and since by fact 4.3 $\|h\| \ge 1$,

$$\bar{f}(e+\bar{\lambda} h) \le \bar{f}(e) - 0.12,$$

completing the proof. ◻

5. Complexity. Consider a linear programming problem in standard format:

$$
\begin{array}{rrcl}
\text{minimize} & c'x & & \\
\text{subject to} & Ax & = & b \\
& x & \geq & 0,
\end{array}
\tag{5.1}
$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, A is an $m \times n$ full-rank matrix, $m < n$.

Assume that the feasible set is bounded, and that all data are integer (or equivalently rational). To simplify the treatment, assume that the cost of an optimal solution $\hat{x}$ is zero.

Define the number l as the total length of the input in binary representation, that is, the summation of the number of bits used by all entries in A, b and c. The significance of this number is the following: no operation containing products and sums of data without repetitions can result in a number greater than 2^l. The determinant of a matrix is dominated by the product of the column norms, and hence $|\det M| \leq 2^l$ for any square submatrix M of $[\, A \mid b \,]$.

Define $L = l + n + 1$.

The next lemma shows that 2^{-L} is indeed a very small number:

LEMMA 5.1. *If x is a vertex of the feasible set, then:*
(i) For $i = 1, \ldots, n$, either $x_i = 0$ or $x_i > 2^{-L}$.
(ii) Either $c'x = 0$ or $c'x > 2^{-L}$.
For any feasible point x,
(iii) $c'x \leq 2^L$, $e'x \leq 2^L$.

Proof. The system $Ax = b$ can be partitioned as

$$A_B x_B + A_N x_N = b,$$

where x_B is a vector made of the positive components of x and A_B is a matrix with the corresponding columns of A. Then $x_N = 0$ and the system can be written as $A_B x_B = b$.

If x is a vertex then the columns in A_B are linearly independent. Assume that A_B is square (or eliminate redundant equations to make it square). Then the solution of the system is given by Cramer's rule, for $i \in B$:

$$x_i = \frac{\Delta_i}{\Delta},$$

where $\Delta = \det A_B$ and Δ_i is the determinant obtained by substituting b for the column A_i.

Let l_1 and l_c be respectively the total number of bits used respectively by $\{A, b\}$ and by c.

(i) is true because Δ_i is integer and $|\Delta| \leq 2^{l_1} \leq 2^L$.

(ii) follows from (i) and the integrality of c.

To prove (iii), consider first the vertex x as above and $i \in B$. Since $|\Delta_i| \leq 2^{l_1}$ and $\Delta \neq 0$ is integer, $x_i \leq 2^{l_1}$. It follows that $e'x \leq n2^{l_1} \leq 2^L$, and since $c_i x_i \leq 2^{l_1 + l_c} = 2^l$, $c'x \leq n2^l \leq 2^L$. This establishes (iii) for all vertices of S.

Since S is bounded, any feasible point can be expressed as a convex combination $\sum_{j \in J} \lambda_j x^j$ of vertices, where $\lambda_j \geq 0$ for $j \in J$ and $\sum_{j \in J} \lambda_j = 1$. It follows that $c'x = \sum_{j \in J} \lambda_j c' x^j \leq 2^L \sum_{j \in J} \lambda_j = 2^L$. A similar argument proves that $e'x \leq 2^L$, completing the proof. ∎

This lemma implies that if we find a vertex with cost smaller than 2^{-L} then it must be an optimal solution.

We now state without proof a known result (see [11]): given any feasible point x there exists a process called "purification of x " that finds with no more than $O(n^3)$ computations a vertex $\bar{x}$ such that $c'\bar{x} \leq c'x$.

The purification algorithm is similar to the simplex method, reducing one variable to zero in each iteration by a pivoting operation, so that no more than $n-1$ pivots are needed to reach a vertex.

Now we are very near our final result: we must prove that Karmarkar's algorithm reaches a point x with $c'x \leq 2^{-L}$ in polynomial time. From this point, a purification detects an exact optimal solution.

THEOREM 5.2. *Consider algorithm 4.1 applied to problem (3.1) with integer data and initial point $x^o = e$. The algorithm followed by a purification finds an optimal solution in $O(nL)$ iterations, and with no more than $O(n^4 L)$ arithmetical operations.*

Proof. Let $(x^k)_{k=0,1,\ldots}$ be the sequence generated by the algorithm. By lemma 5.1,

$$f(x^o) = f(e) = n\log(c'e) \leq n\log(2^L) \leq nL.$$

By lemma 4.4, at iteration k

$$f(x^k) < f(x^o) - 0.1k \leq nL - 0.1k.$$

After no more than $30nL$ iterations, the algorithm stops with

$$f(x^k) = n\log(c'x^k) - \sum_{i=1}^{n} \log x_i^k < -2nL.$$

Again from lemma 5.1, $\sum_{i=1}^{n} \log x_i^k < nL$, and thus

$$\log(c'x^k) < -L.$$

This implies that $c'x^k < e^{-L} < 2^{-L}$. A purification of x^k finds an optimal solution.

The total number of iterations is $k \leq 30nL$, and therefore $k = O(nL)$. The number of computations per iteration is dominated by the projection computation, which can be performed in $O(n^3)$ arithmetical operations. The total number of operations is then $O(n^4 L)$ in the algorithm, which dominates the $O(n^3)$ operations needed in the purification, completing the proof. □

6. The algorithm for unknown optimal cost. This section describes the procedure for using and updating lower bounds for the optimal cost devised by Todd and Burrell [13], in a variant formulation.

If the value $\hat{v}$ of an optimal solution is known, then it is easy to adapt the algorithm to solve problem (3.1) with cost $c'x - \hat{v}$. This can be done by noting that for feasible points $c'x - \hat{v} = c'x - \hat{v}a'x = (c - \hat{v}a)'x$. All we must do is apply the algorithm on the problem

$$\underset{x \in S}{\text{minimize}} (c - \hat{v}a)'x.$$

After scaling in each iteration the cost will be $\bar{c} - \hat{v}\bar{a}$, where $\bar{c} = Xc$ and $\bar{a} = Xa$. By lemma 4.2, $P\bar{c} - \hat{v}P\bar{a}$ has a non-positive component.

Lower bounds.

Suppose now that $\hat{v}$ is not known, but that a lower bound $v^k \leq \hat{v}$ is known in the beginning of iteration k of the algorithm, and the cost $c - v^k a$ is used.

If $P\bar{c} - v^k P\bar{a}$ has a non-positive component, then the conclusion of lemma 4.4 is still valid, since its proof was based in the non-positivity of the projected cost vector.

If $P\bar{c} - v^k P\bar{a}$ is positive, then a new lower bound for $\hat{v}$ can be found by increasing v^k until one of the components of this vector reaches the value zero. It is easy to see that the result v^{k+1} of this procedure is indeed a lower bound: for $u < v^{k+1}$ the vector is positive and as we just saw, such u cannot be equal to $\hat{v}$.

At the end of the iteration, we have:

$$\begin{aligned} f(x^{k+1}) &= n\log(c'x^{k+1} - \hat{v}) + p(x^{k+1}) \\ &\leq n\log(c'x^{k+1} - v^{k+1}) + p(x^{k+1}) && \text{since } v^{k+1} \leq \hat{v} \\ &\leq n\log(c'x^{k} - v^{k+1}) + p(x^{k}) - 0.1 && \text{by lemma 4.4} \\ &\leq n\log(c'x^{k} - v^{k}) + p(x^{k}) - 0.1 && \text{since } v^{k} \leq v^{k+1}. \end{aligned}$$

For iteration k we get

$$f(x^k) \leq n\log(c'x^o - v^o) + p(x^o) - 0.1k,$$

and the complexity proof follows the same path, for a reasonable initial lower bound v^o.

The complete algorithm will be:

ALGORITHM 6.1. *Karmarkar with Todd and Burrell updates: Assume given* $x^o = e$, $x^o \in S$, $v^o \leq \hat{v}$ *and* $\epsilon > 0$. *Set* $k := 0$.

REPEAT
- Scaling: $X = \operatorname{diag}(x_1^k, \ldots, x_n^k)$, $\bar{A} := AX$, $\bar{c} := Xc$, $\bar{a} := Xa$.
- Projection matrix into Null($\bar{A}$): $P = I - \bar{A}'(\bar{A}\bar{A}')^{-1}\bar{A}$
- Lower bound:
  - If $P\bar{c} - v^k P\bar{a} \not> 0$ then $v^{k+1} := v^k$
  - Else $v^{k+1} := \min\{u \mid P\bar{c} - uP\bar{a} \not> 0\}$.
- Direction: $h := -\dfrac{n}{\bar{c}'e - v^{k+1}}(P\bar{c} - v^{k+1}P\bar{a}) + e$.
- Linesearch: $\bar{\lambda} := \operatorname{argmin}\{\bar{f}(e + \lambda h) \mid \lambda \geq 0 \, , \, e + \lambda h > 0\}$.
- Result: $y^* := e + \bar{\lambda}h$.
- Back to the original space: $x^{k+1} := Xy^*$.
- Back to the feasible set: $x^{k+1} := x^{k+1}/a'x^{k+1}$.
- $k := k + 1$.

UNTIL $c'x^k - v^k < \epsilon$.

7. Conclusion. This paper described Karmarkar's algorithm, with the inclusion of a technique for dealing with the case of unknown optimal cost. This algorithm in not used in practical implementations nowadays, because a great evolution of interior point methods has led to better algorithms, both theoretically and practically. In the beginning, most of the research was concentrated in the properties of Karmarkar's algorithm, and in improvements and extensions of methods based on his potential function. Around 1986, it became clear that good interior point algorithms should follow a curve called *central path*, which was already present in classical barrier function methods for non-linear programming, and which can also be related to Karmarkar's algorithm. This gave rise to several different lines of research and many *path* *following algorithms*. The complexity was further reduced to $O(\sqrt{n}L)$ iterations and $O(n^3L)$ arithmetical computations, and primal-dual algorithms proved to perform better than purely primal methods like Karmarkar's.

The methods that follow Karmarkar's approach are known as *potential reduction methods*, surveyed in Todd [14]. The evolution of path following algorithms until 1992 is described in our tutorial paper [8], where some of the concepts used in the present paper are explained in detail. There are now at least three textbooks on interior point methods, where the reader can find many references and systematic developments of the methods: Ye [17], Roos, Terlaky and Vial [12] and Wright [16].

REFERENCES

[1] I. Adler, M. Resende, G. Veiga, and N. Karmarkar, *An implementation of Karmarkar's algorithm for linear programming*, Mathematical Programming, 44 (1989).

[2] K. Anstreicher, *A monotonic projective algorithm for fractional linear programming*, Algorithmica, 1 (1986), pp. 483–498.

[3] E. R. Barnes, *A variation on Karmarkar's algorithm for solving linear programming problems*, Mathematical Programming, 36 (1986), pp. 174–182.

[4] G. Dantzig, *Maximization of a linear function of variables subject to linear inequalities*, in Activity Analysis of Production and Allocation, T. C. Koopmans, ed., Wiley, N. York, 1951, pp. 339–347.

[5] I. I. Dikin, *Iterative solution of problems of linear and quadratic programming*, Soviet Mathematics Doklady, 8 (1967), pp. 674–675.

[6] ———, *On the speed of an iterative process*, Upravlyaemye Sistemi, 12 (1974), pp. 54–60.

[7] C. Gonzaga, *Conical projection algorithms for linear programming*, Mathematical Programming, 43 (1988), pp. 151–173.

[8] ———, *Path-Following Methods for Linear Programming*, SIAM Review, 34, n.2 (1992) pp. 167-224.

[9] N. Karmarkar, *A new polynomial time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.

[10] L. G. Khachiyan, *A polynomial algorithm for linear programming*, Soviet Mathematics Doklady, 20 (1979), pp. 191–194.

[11] K. Kortanek and Z. Jishan, *New purification algorithms for linear programming*, Naval Research Logistics, 35 (1988), pp. 571–583.

[12] Roos, C., Terlaky, T., Vial, J.P. *Theory and Algorithms for Linear Optimization:An Interior Point Approach*,Chichester, John Willey and Sons, 1997.

[13] M. Todd and B. Burrell, *An extension of Karmarkar's algorithm for linear programming using dual variables*, Algorithmica, 1 (1986), pp. 409–424.

[14] M. Todd, *Potential-reduction methods in mathematical programming*, Mathematical Programming, 76 (1997), pp. 3–45.

[15] R. J. Vanderbei, M. J. Meketon, and B. A. Freedman, *A modification of Karmarkar's linear programming algorithm*, Algorithmica, 1 (1986), pp. 395–407.

[16] S. Wright, *Primal-Dual Interior-Point Methods*, Philadelphia, SIAM Publications, 1997.

[17] Ye, Y. *Interior Point Algorithms*, Chichester, John Willey and Sons, 1997.