Copyright © 1985, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

A CONICAL PROJECTION ALGORITHM FOR LINEAR PROGRAMMING

by

C. Gonzaga

Memorandum No. UCB/ERL M85/61

25 July 1985

A CONICAL PROJECTION ALGORITHM FOR LINEAR PROGRAMMING

by

C. Gonzaga

Memorandum No. UCB/ERL M85/61

25 July 1985

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Conical Projection Algorithm For Linear Programming

Clovis Gonzaga

Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720

ABSTRACT

The Linear Programming Problem is manipulated to be stated as a Non-Linear Programming Problem having as objective function Karmarkar's logarithmic potential function. The resulting problem is then solved by a master algorithm that iteratively rescales the problem and calls an internal unconstrained non-linear programming algorithm that reduces the potential function. We show that Karmarkar's algorithm is equivalent to this method in the special case in which the internal algorithm is reduced to a single line search. The new algorithm has the same complexity of Karmarkar's method, but the amount of computation is reduced by the fact that only one projection matrix must be calculated for each call of the internal algorithm.

July 25, 1985

On leave from the Dept. of Systems Engineering and Computer Science, COPPE, Federal University of Rio de Janeiro, Brasil.

A Conical Projection Algorithm For Linear Programming

Clovis Gonzaga

Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720

1. Introduction

The Linear Programming Algorithm created by Karmarkar [1] is based on the use of typical non-linear programming techniques, evolving by a sequence of line searches along internal feasible descent directions for his logarithmic potential function. It has been noticed [2,3] that the method resembles barrier function methods and methods of centers, but the precise role of non-linear programming procedures in his approach has not yet been clearly stated. The nature of the projective transformation used in Karmarkar's algorithm is vaguely understood, as well as the role played by the unit simplex on which the action takes place.

In this paper we intend to provide answers to these questions, and show that the unit simplex is not needed at all. By isolating the utilization of non-linear programming procedures the path to improve the algorithm performance is opened, resulting in greater reductions of the objective function between computations of the projection matrix.

We shall present an algorithm with two levels of hierarchy, composed of a "master" algorithm that manipulates the Linear Programming Problem and calls an "internal" non-linear programming algorithm that provides a strategy for improving the current solution. The master algorithm re-scales the Linear Programming Problem by means of a linear transformation that places the current solution at the point $e = [1, 1, \ldots, 1]'$. An "internal problem" is then defined, consisting of a non-linear potential function to be minimized in a linear space with positivity constraints, and any non-linear programming algorithm can be used to reduce the value of its objective function.

We state the complete "master" algorithm in section 2, and prove its linear convergence. In sections 3 and 4 we present the internal algorithm and details on the determination of lower bounds for the value of an optimal solution.

The Linear Programming Problem

Consider the Linear Programming Problem

$$\begin{aligned} &\textit{minimize} && c'x \\ &\textit{subject to} && Ax = 0 \\ & && a'x = 1 \\ & && x \geq 0 \end{aligned} \qquad \text{(P)}$$

where $n > m > 0$, $c, x \in R^n$, $a \in R^m$, A is an $m \times n$ matrix.

The following hypotheses must be satisfied: $a \geq 0$, $a \neq 0$, the feasible set is compact, an initial feasible solution x^0 is known as well as a lower bound v^0 for the value of an optimal solution.

The formulation is quite general, since given a problem in R^{n-1}:

$$\begin{aligned} &\textit{minimize} && c'x \\ &\textit{subject to} && \bar{A}x = b \\ & && x \geq 0 \end{aligned}$$

it is enough to set $A := [\bar{A} \; -b]$, and to introduce the variable x_n constrained by $x_n = 1$. It is also clear that the simplex constraint $e'x = 1$ is also accepted by the formulation and guarantees the compactness assumption.

We shall denote the feasible set of (P) by C, the value of an optimal solution by v, and we shall make use of Karmarkar's Potential Function $f(.)$ defined for all $x > 0$ by

$$f(x) = n \log(c'x - v) - \sum_{j=1}^{n} \log x_j \qquad (1)$$

The function cannot be evaluated, since v is unknown. The algorithm uses instead lower bounds for v, calculated at each iteration. Given the lower bound u, the problem can be restated with the objective function $c'x - u$, and with a potential function defined for all $x \geq 0$;

$$f_u(x) = n \log(c'x - u) - \sum_{j=1}^{n} \log x_j \qquad (2)$$

The objective function $c'x - u$ can be put in the standard format by using the following lemma:

1.1 *Lemma:* For all $x \in C$, $c'x - u = (c - ua)'x$.

Proof: Immediate, by noticing that for all $x \in C$, $ua'x = u$, since $a'x = 1$.

2. The master algorithm

In this section we present the complete master algorithm and prove its linear convergence as a consequence of the assumed behavior of the internal non-linear programming search. The algorithm will generate a sequence of points (x^k) and an increasing sequence of lower bounds (v^k) for v, in such a way as to guarantee a substantial decrease in $f(.)$ in each iteration.

We now present the master algorithm, to be commented immediately afterwards.

Each iteration of the master algorithm will perform the following operations:

Scaling: a linear transformation brings the current solution x^k to the point $e = (1,1,..1)$.

Projection: calculate the projection matrix onto the new linear space. This time-consuming operation is done only once for each iteration of the master algorithm.

Calculation of a new lower bound: see section 4.

Definition of the internal problem, computation of a solution with a low value and then back to the original problem.

The internal algorithm makes use of the number α, to be defined in (12), and in this section we shall assume that the potential function defined in the internal problem can be reduced by α in each iteration. The proof of this fact will be the object of sections 3 and 4.

2.1 *Algorithm*: Given a precision $\varepsilon > 0$, a lower bound $v^0 \leq v$ and $x^0 \varepsilon C$, $x^0 > 0$.
Set $k := 0$.
While $c'x^k - v^k > \varepsilon$ do
Define $D := diag(x_1^k, x_2^k, \ldots, x_n^k)$.
(scaling) Define $A^k := AD$, $a^k := Da$.
(projection) Calculate the projection matrix P onto $Null(A^k)$.
Set $c_p := PDc$, $a_p := Pa^k$.
(internal algorithm)
Calculate a lower bound u such that $v^k \leq u \leq v$.
Set $\bar{c} := c_p - ua_p$.
Define the *Internal Problem*

$$\textit{minimize} \quad g(y) := n \log \bar{c}'y - \sum_{j=1}^{n} \log y_j \qquad \text{(Pi)}$$

$$\textit{subject to} \quad A^k y = 0$$
$$y > 0$$

Use a non-linear programming algorithm to find a feasible point y^* for (Pi) such that $g(y^*) < g(e) - \alpha$.
(conical projection) calculate

$$\bar{y} := \frac{1}{a^{k\prime} y^*} y^* \qquad (3)$$

(back to the original space) Set
$x^{k+1} := Dy^*$
$v^{k+1} := u$
$k := k + 1$

Conical Projections

The internal problem (Pi) is not well defined, since for y near 0, $g(y)$ can assume any value. We shall nevertheless keep this formulation with the understanding that the desired result is a truncated sequence of points with decreasing values of the objective function. (Pi) makes no use of the constraint $a'x = 1$: its feasible set C^k is simply the intersection of $Null(A^k)$ and the interior of the first orthant. C^k is interesting for two reasons: first, because it is easy to use search methods on it, since the projection matrix is known; secondly, because it is a cone with the property that the potential function is constant on each of its rays, as we formalize in the following lemma:

2.2 *Lemma*: For all $y \varepsilon C^k = \{y > 0 \mid A^k y = 0\}$, for any $\lambda > 0$,

$$g(\lambda y) = g(y) .$$

Proof: by direct substitution in the definition of $g(.)$.

Consequently, to each feasible point x for (P), a ray $\{\lambda D^{-1}x \mid \lambda > 0\}$ of constant potential in (Pi) is associated. Conversely, given a point $y^* \varepsilon C^k$ resulting from the internal algorithm, the point $\bar{y} \varepsilon C^k$ given by the expression (3) satisfies $g(\bar{y}) = g(y^*)$ and $a^{k\prime}\bar{y} = 1$, as can be trivially verified. $\bar{y}$ is the

conical projection of y^* onto $e + Null(a^{k\prime})$, i.e. the intersection of the ray through y^* and $e + Null(a^{k\prime})$. The existence of the conical projection is guaranteed by the hypotheses $a \geq 0$, $a \neq 0$ and $y^* > 0$. The subject of conical projections will be further studied in section 5.

If $v^k = v$, then the internal problem is equivalent to problem (P), and the problem can in principle be solved by a single application of a good non-linear programming algorithm. This is unfortunately not likely to happen in practice, due to the bizarre behavior of the potential function near the boundaries of the orthant: it tends to $+\infty$ near any non-optimal boundary point $x \neq 0$, to $-\infty$ for a sequence convergent to an optimal solution, and to any number for sequences convergent to 0. Besides this, the rate of convergence of non-linear programming methods depends on well-conditioned Hessian matrices, which is definitely not guaranteed near optimal solutions. On the other hand, non-linear programming methods can be very effective in reducing the value of the potential function while far from the boundary of C^k, and this can be done at a low cost since no projection matrices must be calculated while iterating on the internal problem.

The internal algorithm (to be studied in the next section) iterates while a substantial decrease in the potential function is obtained. It will be shown that if its first iteration is a steepest descent search, then in this iteration the potential function $g(.)$ drops by at least a fixed constant $\alpha > 0.25$. The first iteration is actually equivalent to the line search done in Karmarkar's method, and α is the constant found in his work [1]. Assuming this behavior for the internal algorithm, we shall now prove the linear convergence of the master algorithm.

Proof of Linear Convergence

Our aim is to prove that the sequence of values $c'x^k$ generated by the algorithm is dominated by a sequence that decreases by a constant ratio at each iteration. The sequence $(c'x^k)$ is not necessarily monotonically decreasing, and its values can increase in the beginning or when the lower bound v^k changes.

2.3 *Lemma*: Let y and x be related by $Dy = x$ in an iteration k of the algorithm. If $A^k y = 0$ then $c_p'y = c'x$, $a_p'y = a^{k\prime}y = a'x$. Furthermore, if $a^{k\prime}y = 1$ then $\bar{c}'y = c'x - u$.

Proof: Immediate consequences of the definitions of a_p and c_p, since for any vector $z \in R^n$, for $y \in Null(A^k)$, $z'y = (Pz)'y$. The last equality is a consequence of lemma 1.1.

2.4 *Lemma*: For any $x \in C$, for $r \leq s \leq v$, $f_s(x) \leq f_r(x)$.

Proof: It is sufficient to notice that for $s \geq r$, $\log(c'x - s) \leq \log(c'x - r)$.

2.5 *Lemma*: At any iteration k, if $g(y^*) \leq g(e) - \alpha$ then

$$f_{v^{k+1}}(x^{k+1}) \leq f_{v^k}(x^k) - \alpha \quad . \tag{4}$$

Proof: Using the notation in the Algorithm 2.1, we first prove that for any vector $x \in C$, for $y = D^{-1}x$,

$$g(y) = f_u(x) + \log\det(D) \tag{5}$$

In fact, using lemma 2.3,

$$g(y) = n\log(\bar{c}'y) - \sum_{j=1}^{n} \log(D_{jj}^{-1}x_j)$$

$$= n\log(c'x-u) - \sum_{j=1}^{n}\log x_j + \sum_{j=1}^{n}\log D_{jj}$$

$$= f_u(x) + \log\det(D)$$

Consequently, if $g(y)$ decreases by α, so does $f_u(x)$.
The internal algorithm starts with $e = D^{-1}x^k$ and finishes with $\bar{y} = D^{-1}x^{k+1}$.
Using the assumption that $g(\bar{y}) \le g(e) - \alpha$,

$$f_u(x^{k+1}) \le f_u(x^k) - \alpha \ .$$

Using the fact that $v^{k+1} = u \ge v^k$,

$$f_{v^{k+1}}(x^{k+1}) = f_u(x^{k+1})$$
$$\le f_u(x^k) - \alpha$$
$$\le f_{v^k}(x^k) - \alpha \qquad \text{, by lemma 2.4,}$$

and the proof is complete.

This shows that the algorithm generates a sequence $(f_{v^k}(x^k))$ that decreases at each iteration by at least α. The sequence $(f(x^k))$ is dominated by it, as a direct consequence of lemma 2.4. It is generally not true that the objective function of (P) decreases at each iteration, but the following results guarantee the overall convergence.

Let $\gamma := \max\left\{ \sum_{j=1}^{n}\log x_j \mid x \varepsilon C, x>0 \right\}$. γ is well defined, since C is compact, the argument is continuous in its interior and decreases indefinitely near its frontier. If x^* is a maximizer of this expression, then it can be thought of as a "point of minimum potential in C".
Define $K = \exp\left(\frac{f_{v^0}(x^0)+\gamma}{n}\right)$.

2.6 *Theorem*: At any iteration k of the algorithm 2.1,
$c'x^k - v \le K\exp\left(-\frac{k\alpha}{n}\right)$.

Proof: Using lemma 2.5, at an iteration k, $f(x^k) \le f_{v^k}(x^k) \le f_{v^0}(x^0) - k\alpha$.
Using the definition of $f(.)$,

$$n\log(c'x^k - v) \le f_{v^0}(x^0) + \sum_{j=1}^{n}\log x_j - k\alpha$$

$$\le f_{v^0}(x^0) + \gamma - k\alpha \ ,$$

by definition of γ. Now, removing the logarithm,

$$c'x^k - v \le \exp\left(\frac{f_{v^0}(x^0)+\gamma}{n} - \frac{k\alpha}{n}\right)$$

and the final result is obtained by using the definition of K.

3. The Internal Algorithm

The Internal Algorithm performs three operations: calculation of a new lower bound, calculation of a sub-optimal solution for the Internal Problem defined in the master algorithm 2.1, and conical projection. The calculation of a new lower bound is done by a simple algorithm to be presented now; the proofs and motivation will be the subject of section 4. In all the development to follow we use the notation introduced in the master algorithm.

3.1 *Algorithm*: Calculation of a new lower bound.

If $c_p - v^k a_p$ has a non-positive component, then $u := v^k$

Else $u := \min_{j=1,2,\ldots,n} \left\{ \frac{c_{p_j}}{a_{p_j}} \mid a_{p_j} \neq 0 \right\}$

As we shall prove in the next section, this algorithm finds a lower bound for v, and guarantees that the cost vector $\bar{c}$ used by the internal algorithm has at least one non-positive component. In the present section we shall prove that this guarantees a decrease of at least $\alpha > 0.25$ for the potential function $g(.)$ from e to $\bar{y}$. We now state the Internal Algorithm.

3.2 *Algorithm*: (model) given a precision $\delta > 0$

$i := 0$; $y^0 := e$

Repeat

Calculate $P\nabla g(y^i) := \frac{n}{\bar{c}' y^i} \bar{c} - P\left[y_1^{i-1}\ y_2^{i-1} \cdots y_n^{i-1} \right]'$

Choose a descent direction $h \in Null(A^k)$ such that $h' P \nabla g(y^i) < 0$.
(in the first iteration, set $h := -P\nabla g(y^i)$)

Find $\bar{\lambda} > 0$ such that $g(y^i + \bar{\lambda} h) = \min\{ g(y^i + \lambda h) \mid \lambda > 0 , y^i + \lambda h > 0 \}$

$y^{i+1} := y^i + \bar{\lambda} h$

$i := i + 1$

Until $g(y^{i-1}) - g(y^i) < \delta$

$y^* := y^i$

The algorithm above is no more than the general model for a non-linear programming feasible directions search, for a special case in which the result of the line searches is guaranteed to lie in the interior of the feasible set. The choice of the descent directions was not specified unless for the first iteration, and can be the result of any strategy like conjugate gradients or variable metric methods [4]. The line search is well defined, since the potential function grows indefinitely near the frontier of the feasible set (unless for the lucky case of hitting an optimal solution for $u = v$). It has been proved in [2] that the potential function restricted to the line $y^i + \lambda h$ is unimodal, and reference [4] gives a method for the search.

We shall then assume hereafter that the line search in 3.2 always has a unique solution $\bar{\lambda}$.

The first iteration of the algorithm follows a steepest descent direction starting at $y^0 = e$. The following results explore the properties of this first iteration, and show that it is actually the search performed by Karmarkar's algorithm.

Properties of the potential function at $y = e$

Consider the potential function defined by the internal problem:

$$g(y) = n \log \bar{c}'y - \sum_{j=1}^{n} \log y_j \quad , \tag{6}$$

where $\bar{c} \in Null(A^k)$, $\bar{c}'e > 0$.
The gradient of the potential function calculated at $y = e$ is given by

$$\nabla g(e) = \frac{n}{\bar{c}'e}\bar{c} - e = P\nabla g(e) \quad , \tag{7}$$

since $Pe = e$.

The search direction used by the first iteration of the internal algorithm is $h = -\nabla g(e)$. The next lemma establishes the connection between our approach and Karmarkar's method:

3.3 *Lemma*: The line $\lambda \geq 0 \rightarrow e + \lambda h$ lies on the unit simplex, defined by the equation $e'x = n$.

Proof: By direct verification,

$$e'(e + \lambda h) = n + \lambda\left(e'e - n\,\frac{\bar{c}'e}{\bar{c}'e}\right) = n \quad .$$

It is now easy to find conditions under which a fixed minimum decrease is guaranteed in this direction, by repeating Karmarkar's argument, or by following [2]. We shall write a new proof, with the intention of presenting a unified treatment. The results to follow are essentially reformulations of properties of the potential function shown in [1] and [2].

3.4 *Lemma*: The logarithm function $x \in R^+ \rightarrow \log x$ is strictly concave and differentiable, and for any $x > 0$, for λ such that $|\lambda| < x$,

$$\log(x+\lambda) = \log x + \frac{\lambda}{x} + o(x,\lambda) \quad , \tag{8}$$

where $\quad |o(x,\lambda)| \leq \dfrac{\lambda^2}{2x^2}\dfrac{1}{1-\dfrac{|\lambda|}{x}}$

Proof:
Since for $x > 0$, $\dfrac{d^2}{dx^2}\log x = -\dfrac{1}{x^2} < 0$, the function is strictly concave.
Now, expanding in series about x ,

$$\log(x+\lambda) = \log x + \frac{\lambda}{x} - \frac{\lambda^2}{2x^2} + \frac{\lambda^3}{3x^3} - \cdots$$

Comparing with (8), we recognize

$$o(x,\lambda) = \sum_{i=2}^{\infty}(-1)^{i-1}\frac{\lambda^i}{ix^i} \quad .$$

Taking the absolute value,

$$|o(x,\lambda)| \leq \sum_{i=2}^{\infty}\frac{1}{i}\left|\frac{\lambda}{x}\right|^i \leq \sum_{i=2}^{\infty}\frac{1}{2}\left|\frac{\lambda}{x}\right|^i$$

Since $|\lambda| < x$, this last series can be added: the result of the summation coincides with the expression in (8), and completes the proof.

In particular, for $x = 1$, we have

$$o(1,\lambda) \le \frac{\lambda^2}{2}\frac{1}{1-|\lambda|} \tag{9}$$

The next lemma resumes Karmarkar's most important result, showing that the in the first line search the potential function has a guaranteed decay that does not depend on the original problem.

3.5 *Lemma*: If $\|h\| \ge 1$ then for any $\lambda \in (0,1)$,

$$g\left(e + \lambda \frac{h}{\|h\|}\right) - g(e) \le -\lambda + \frac{\lambda^2}{2(1-\lambda)}$$

Proof: Let $g(y) = g_1(y) - g_2(y)$, where $g_1(y) := n \log \bar{c}'y$, $g_2(y) := \sum_{j=1}^{n} \log y_j$.

We have: $\nabla g_2(e) = e$, $\nabla g_1(e) = \nabla g(e) + e = -h + e$.

(a) We start by examining the decrease in $g_1(.)$.
g_1 is concave, since $\log(.)$ is concave and $y \to \bar{c}'y$ is linear. Consequently,

$$g_1\left(e + \lambda \frac{h}{\|h\|}\right) \le g_1(e) + \lambda \nabla g_1'(e) \frac{h}{\|h\|}$$

Now, notice that $\nabla g_1'(e)\, h = (-h + e)'h = -\|h\|^2$, since by lemma 3.3 $e'h = 0$.
Using this result in the inequality, and assuming that $\|h\| \ge 1$,

$$g_1\left(e + \lambda \frac{h}{\|h\|}\right) - g_1(e) \le -\lambda \|h\| \le -\lambda \tag{10}$$

(b) We now examine the variation of g_2. Using lemma 3.4, for $0 \le \lambda < 1$

$$\begin{aligned} g_2\left(e + \lambda \frac{h}{\|h\|}\right) &= \sum_{j=1}^{n} \log\left(1 + \lambda \frac{h_j}{\|h\|}\right) \\ &= \sum_{j=1}^{n} \log 1 + \lambda \sum_{j=1}^{n} \frac{h_j}{\|h\|} + \sum_{j=1}^{n} o\left(1, \frac{\lambda h_j}{\|h\|}\right) \\ &= \sum_{j=1}^{n} o\left(1, \frac{\lambda h_j}{\|h\|}\right) \quad , \text{since } \sum_{j=1}^{n} h_j = e'h = 0 \ , \text{ by lemma 3.3} . \end{aligned}$$

Now, using (8),

$$\begin{aligned} \left| g_2\left(e + \lambda \frac{h}{\|h\|}\right) \right| &\le \frac{1}{2} \sum_{j=1}^{n} \frac{\lambda^2 h_j^2}{\|h\|^2} \frac{1}{1 - \lambda \frac{|h_j|}{\|h\|}} \\ &\le \frac{\lambda^2}{2} \sum_{j=1}^{n} \frac{h_j^2}{\|h\|^2} \frac{1}{1-\lambda} \quad \text{since } |h_j| \le \|h\| \ , j = 1, \ldots, n \\ &= \frac{\lambda^2}{2} \frac{1}{1-\lambda} \quad , \quad \text{since } \sum_{j=1}^{n} h_j^2 = \|h\|^2 \end{aligned} \tag{11}$$

Finally, combining (10) and (11), and noting that $g_2(e) = 0$,

$$\begin{aligned} g\left(e + \lambda \frac{h}{\|h\|}\right) - g(e) &\le g_1\left(e + \lambda \frac{h}{\|h\|}\right) - g_1(e) + \left| g_2\left(e + \lambda \frac{h}{\|h\|}\right) \right| \\ &\le -\lambda + \frac{\lambda^2}{2(1-\lambda)} \ . \end{aligned}$$

completing the proof.

The lemma above leads us to the conclusion that if in the first iteration of algorithm 3.2 $\|h\| \geq 1$, then the line search results in a reduction of the potential function g with a lower bound α that does not depend on the original problem, given by

$$\alpha = -\min_{0<\lambda<1}\left(-\lambda + \frac{\lambda^2}{2(1-\lambda)}\right) > 0.25 \qquad (12)$$

The minimum is reached for λ near 0.423, and the value is approximately -0.268.

To obtain the desired reduction in the potential function, the condition $\|h\| \geq 1$ must be guaranteed. The next lemma reduces this condition to an easier one, dependent on the value of $\bar{c}$, and consequently on the lower bound u calculated by algorithm 3.1.

3.6 *Lemma*: If $\bar{c}$ has a non-positive component, then $\|h\| \geq 1$ in the first iteration of the internal algorithm 3.2.

Proof: From (7), $h = -\frac{n}{\bar{c}'e}\bar{c} + e$.

Suppose that for some $k = 1,..,n$, $\bar{c}_k \leq 0$. Then, $h_k = 1 - \frac{n}{\bar{c}'e}\bar{c}_k \geq 1$, since $\bar{c}'e > 0$. The fact that $\|h\| \geq h_k$ completes the proof.

To establish definitely the efficiency of the internal algorithm, we must still prove that the vector $\bar{c} = c_p - u\,a_p$ with u resulting from Algorithm 3.1 has a non-positive component. This will be the object of section 4.

4. Determination of lower bounds for the value of an optimal solution

In the beginning of iteration k of the master algorithm, a feasible solution x^k is known. A linear transformation on the original problem (P) brings this point to the vector e in the new coordinates. Using the notation introduced in Algorithm 2.1, a Linear Programming Problem can be written in the new coordinates:

$$\begin{aligned} &\textit{minimize} && c_p'y && (Pk)\\ &\textit{subject to} && A^k y = 0 \\ & && a_p'y = 1 \\ & && y \geq 0 \end{aligned}$$

This problem is equivalent to (P), in the sense that each feasible point x for (P) is univocally associated to a point $y = D^{-1}x$ feasible for (Pu), and their costs are related by $c_p'y = c'x$, as was shown in lemma 2.3. The value of an optimal solution for (Pk) is then equal to the value v of an optimal solution for (P).

In the beginning of iteration k a value $v^k \leq v$ is known. If $\bar{c} = c_p - v^k a_p$ has a non-positive component, then lemmas 3.6 and 3.5 guarantee a decrease of at least α for the potential function, and the condition for linear convergence (theorem 2.6) is fulfilled at this iteration. If this is not the case, a new lower bound for v must be found, so that the new $\bar{c}$ has a non-positive component: in this section we show that this is always possible, and it is accomplished by algorithm 3.1, repeated below:

Algorithm: Calculation of a new lower bound.

If $c_p - v^k a_p$ has a non-positive component, then $u := v^k$

$$\text{Else } u := \min_{j=1,2,\ldots,n} \left\{ \frac{c_{p_j}}{a_{p_j}} \mid a_{p_j} \neq 0 \right\} \tag{13}$$

4.3 *Theorem*: Algorithm 3.1 generates a lower bound u for v, and $c_p - u a_p$ has a non-positive component.

Proof: If $c_p - v^k a_p$ has a non-positive component, then the result is trivial. Suppose then that $c_p - v^k a_p > 0$.

Consider the following relaxed version of (Pk):

$$\begin{aligned} &\textit{minimize} && c_p' y \\ &\textit{subject to} && a_p' y = 1 \\ & && y \geq 0 \end{aligned} \tag{14}$$

The dual of this Linear Programming problem is given by:

$$\begin{aligned} &\textit{maximize} && z \qquad , \quad z \in R \\ &\textit{subject to} && z\, a_p \leq c_p \end{aligned} \tag{15}$$

By hypothesis, $v^k a_p < c_p$, and consequently (15) is feasible. It follows that (14) has an optimal solution y^*, (15) has an optimal solution $u > v^k$, and $u = c_p' y^*$.

Since (14) was obtained by relaxing (Pk), u is a lower bound for v.

(15) can be rewritten as

$$\textit{maximize} \quad z$$

$$\textit{subject to} \quad z \leq \frac{c_{p_j}}{a_{p_j}} \qquad j = 1,\ldots,n \qquad \textit{such that} \quad a_{p_j} \neq 0$$

The solution for this problem is given by (13), and for some index k, $u = \dfrac{c_{p_k}}{a_{p_k}}$.

It follows that $c_{p_k} - u\, a_{p_k} = 0$, completing the proof.

5. Conical Projections : working in the original space

Each iteration of the internal algorithm starts with a point y^i and performs a line search along a direction h, resulting in a point $y^* = y^i + \bar{\lambda} h$. In particular, the first iteration works on the line $\{ e + \lambda \bar{h} \mid \lambda \geq 0 \}$, where $\bar{h} = -\dfrac{n}{\bar{c}'e}\bar{c} + e$. This line lies on the unit simplex, and the line search results in a guaranteed decrease greater than 0.25 : this seems to attach a great importance to the unit simplex.

We will show that the property above is by no means connected to the unit simplex. Due to the 0-degree homogeneity of $g(.)$, there exists a cone of directions leading to the same decrease as $\bar{h}$, and the minimizers for all these directions lie on the same ray.

Consider a set S in the first orthant of R^n. The *cone generated by* S is defined as $K(S) = \{ \alpha y \in R^n \mid \alpha > 0, y \in S \}$.

Since the potential function $g(.)$ is 0-degree homogeneous, i.e, $g(\lambda y) = g(y)$ for any $\lambda > 0$, it follows that

$$\inf_{y \in S} g(y) = \inf_{y \in K(S)} g(y) \qquad (16)$$

If two sets S, S_1 in the first orthant generate the same cone, then $\inf_{y \in S} g(y) = \inf_{y \in S_1} g(y)$.

This shows that the minimization of $g(.)$ along two different lines lead to the same optimal value whenever both lines generate the same cone.

We shall say that two directions h_1, h_2 are *equivalent from* $y > 0$, whenever minimizers y_1, y_2 of $g(.)$ respectively along h_1 and h_2 in the first orthant exist and lie on the same ray (and consequently $g(y_1) = g(y_2)$). Some facts are straightforward:

5.1 *Lemma*: Let $\bar{h}$ be such that $y^i + \bar{h} > 0$ in some iteration of the internal algorithm, and that the line search from y_i results in $\bar{\lambda} < 1$. Then for any $\alpha > 0$, the direction $h = \alpha(y^i + \bar{h}) - y^i$ is equivalent to $\bar{h}$ from y^i.

Proof: It is sufficient to see that the line segments $\{y^i + \lambda\bar{h} \mid \lambda \in [0,1]\}$ and $\{y^i + \lambda h \mid \lambda \in [0,1]\}$ generate the same cone, defined by the extreme rays $\{\delta y^i \mid \delta > 0\}$ and $\{\delta(y^i + h) \mid \delta > 0\}$.

5.2 *Lemma*: If two directions are equivalent from $y > 0$, then positive multiples of these directions are also equivalent from y.

Proof: Straightforward, since scaling the directions does not change the result of the line search.

5.3 *Lemma*: Let $\bar{h}$ be as in lemma 5.1. Then for any $\mu \in (-\infty, 1)$, the direction $h = \bar{h} + \mu y^i$ is equivalent to $\bar{h}$ from y^i.

Proof: By lemma 5.1, for any $\alpha > 0$ the direction $\alpha\bar{h} + (\alpha - 1)y^i$ is equivalent to $\bar{h}$ from y^i. By lemma 5.2, we can multiply this direction by α^{-1}, obtaining $h = \bar{h} + \alpha - \frac{1}{\alpha} y^i$, $\alpha > 0$. Setting $\mu = 1 - \frac{1}{\alpha}$, the directions are defined by $h = \bar{h} + \mu y^i$, $\mu \in (-\infty, 1)$, completing the proof.

In particular, in the first iteration, any direction of the form

$$-\frac{n}{\bar{c}'e}\bar{c} + e - \mu e = \frac{n}{\bar{c}'e}\bar{c} + \nu e \quad , \quad \nu \in (-\infty, 2) \quad , \quad \text{is equivalent to } \bar{h}$$

This set of directions includes for instance $h = -\bar{c}$, which consequently is as good as the gradient direction as a search direction. The only advantage that seems to arise from the fact that $\bar{h}$ lies on the unit simplex is in the ease with which the guaranteed decay of $g(.)$ is proved: direct proofs for other equivalent directions may not be so elegant.

The discussion above is useful for two reasons: first because it indicates that no improvement in the convergence rates can be obtained by trying other directions generated by linear combinations of $\bar{c}$ and e (or $\bar{h}$ and $[(y_j^i)^{-1}]$ in the general case); the second reason will be explored below, and is more appealing: the line searches can be executed in the original space, by projecting the search direction *conically* on $y^i + Null\,(a^{k'})$.

Working on $y^i + Null\,(a^{k\prime})$.

Consider a vector $\gamma \in R^n$, $\gamma \neq 0$, and a point $y \neq 0$.

5.4 *Definition*: Given a point $z > 0$, the *conical projection* $K_{y,\gamma}(z)$ of z onto $y + Null\,(a^{k\prime})$ is defined as the intersection of $y + Null\,(a^{k\prime})$ and the ray $\{\alpha z \mid \alpha > 0\}$, if the intersection exists.

If $\gamma \geq 0$, $\gamma \neq 0$ and $y > 0$, then the conical projection of any vector $z > 0$ exists, and is given by

$$K_{y,\gamma}(z) = \frac{\gamma' y}{\gamma' z} z \qquad . \tag{17}$$

In fact, we obtain

$$\gamma' K_{y,\gamma}(z) = \frac{\gamma' y}{\gamma' z} \gamma' z = \gamma' y \quad ,$$

and hence $K_{y,\gamma}(z) \in y + Null\,(\gamma')$

In particular,

$$\text{if} \quad \gamma' y = 1 \quad , \quad \text{then} \quad K_{y,\gamma}(z) = \frac{z}{\gamma' z} \tag{18}$$

5.5 *Definition*: Given a direction $h \in R^n$, the conical projection $H_{y,\gamma}(h)$ of the direction h onto $y + Null\,(\gamma')$ is the direction $K_{y,\gamma}(y+h) - y$, if it exists.

Consider the potential function $g(.)$ at some iteration of the internal algorithm.

5.6 *Lemma*: Let h be such that $y + h > 0$ and $\min\{g(y+\lambda h) \mid \lambda \geq 0 \ , \ y + \lambda h > 0\}$ exists. Then for any $\gamma > 0$, $\gamma \neq 0$, the directions h and $H_{y,\gamma}(h)$ are equivalent from y .

Proof: Let $\bar{\lambda}$ be a solution for the line search along h . If $\bar{\lambda} < 1$, then the result is an immediate consequence of lemma 5.1, since the projected direction is given by

$$H_{y,\gamma}(h) = \frac{\gamma' y}{\gamma'(y+h)} (y+h) - y \quad , \text{ and } \quad \frac{\gamma' y}{\gamma'(y+h)} > 0 \quad .$$

If $\bar{\lambda} \geq 1$, then the result follows from an application of lemma 5.2, since it is possible to scale h so that the hypotheses of lemma 5.1 are satisfied, and the resulting conical projection will obviously be a multiple of $H_{y,\gamma}(h)$. This completes the proof.

We now see that it is possible to work always on $\{y \mid a^{k\prime} y = 1\}$, by introducing the following modification in the internal algorithm (3.2):

5.7 Line search along conically projected directions:

Choose a descent direction $\bar{h} \in Null(A^k)$ such that $\bar{h}' P \nabla g(y^i) < 0$.

Scale $\bar{h}$ so that $y^i + \bar{h} > 0$

Set h equal to the conical projection of $\bar{h}$ onto $y^i + Null\,(a^{k\prime})$

The scaling of $\bar{h}$ can be done either by setting $\bar{h} := \frac{\|y^i\|}{\|\bar{h}\|} \bar{h}$, or by a com-

mand like

while *not* $(y^i + \bar{h} > 0)$ do $\bar{h} = 0.5\bar{h}$.

The remaining steps of the algorithm are not changed. The conical projection at the end of the internal algorithm is no more needed.

Nothing seems to be gained by working on $y + Null\,(a^{k+1})$. On the other hand, much can be lost, since the conical projections prevent us from using non-linear programming methods like conjugate gradients. The only possible advantage of this approach lies on the possibility of working in the original space, by mapping the search directions back to original coordinates and carrying the line search in that space.

Working in the original space

The internal algorithm can now be rewritten so that line searches are performed in the original space, simply by setting at each iteration $h_x := D h_y$, where h_y is the direction found in 5.7 . The line search to be done in this direction must minimize $f_u(.)$. We shall not rewrite the algorithm, since it is straightforward: notice that the projection matrix must only be recalculated whenever the decrease in a line search is small.

Again, there is no advantage in applying this procedure to regular LP problems, for two reasons: first, powerful non-linear programming algorithms cannot be directly applied; second, the repeated conical projections and changes of coordinates require extra computations and spread errors.

On the other hand, this can be interesting if the LP problem has been obtained by transforming or approximating some parent problem (e.g. in internal or external linearizations, or in dealing with inequality constraints): in these cases it may be possible to further transform the search directions so as to perform the line searches in the feasible set of the parent problem.

6. Conclusion

In his original work [1], Karmarkar needed the unit simplex as the set on which the action takes place. This can be easily understood, since the orthogonal projection of the inverted cost vector $-\bar{c}$ on the simplex coincides with $-\nabla g(e)$, and also with the conical projection of $-\nabla g(e)$. The same behavior is not observed at any other point y , and no substantial decrease of $g(.)$ should be expected by projecting $-\nabla g(y)$ *orthogonally* on the simplex. The study of conical projections clears the subject and sets us free from the unit simplex.

The computational aspects must now be studied in two lines: efficient calculation of the projection matrices, and efficient algorithms for the solution of the internal problem.

Much work has already been done in the first direction, based on the calculation of an LR representation for the projection matrix [6],[2],[3],[5]. These results are directly applicable to the present method.

The second direction consists in studying an unconstrained non-linear programming problem with the specific objective function $g(.)$. The best techniques are still to be detected, since $g(.)$ is not convex, but it is pseudo-convex, in the sense that its level sets are convex.

Acknowlegement

I would like to express my thanks to Prof. E. Polak for his friendly support and encouragement.

Research partly sponsored by CNPq - Brazilian National Council for Scientific and Technological Development, by National Science Foundation grant ECS-8121149, Office of Naval Research contract N00014-83-K-0502, and AFOSR grant 83-0361.

References

[1] N. Karmarkar, "A New Polynomial Time Algorithm for Linear Programming", AT&T Bell Labs, Murray Hill, NJ, 1984

[2] M. Todd, B. Burrell, "An extension of Karmarkar's Algorithm for Linear Programming Using Dual Variables", Technical Report 648, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, January 85.

[3] T. Cavalier, A. Soyster, "Some Computational Experience and a Modification of the Karmarkar Algorithm", Working Paper 85-105, Dept. of Industrial and Management System Eng., Pennsylvania State Univ., Feb. 85.

[4] P. Gill, W. Murray, M. Wright, Practical Optimization , Princeton University Press, New York, NY, 1981.

[5] J. Tomlin, "An Experimental Approach to Karmarkar's Projection Method for Linear Programming", Working Paper, Ketron, Inc., Mountain View, CA, 1985.

[6] M. Heath, "Some Extensions of an Algorithm for Sparse Linear Least Squares Problems", SIAM Journal on Scientific and Statistical Computing 3 (1982) 223-237.