AN ALGORITHM FOR SOLVING LINEAR PROGRAMMING PROBLEMS IN $O(n^3 L)$ OPERATIONS

by

Clovis C. Gonzaga

Memorandum No. UCB/ERL M87/10

5 March 1987

AN ALGORITHM FOR SOLVING LINEAR PROGRAMMING PROBLEMS IN $O(n^3 L)$ OPERATIONS

by

Clovis C. Gonzaga

Memorandum No. UCB/ERL M87/10

5 March 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

AN ALGORITHM FOR SOLVING LINEAR PROGRAMMING PROBLEMS IN $O(n^3 L)$ OPERATIONS

by

Clovis C. Gonzaga

Memorandum No. UCB/ERL M87/10

5 March 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

An Algorithm for Solving Linear Programming Problems in $O(n^3L)$ Operations

Clovis C. Gonzaga *
Dept. of Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720

March 5, 1987

Abstract

This paper describes a short-step penalty function algorithm that solves linear programming problems in no more than $O(n^{0.5}L)$ iterations. The total number of arithmetic operations is bounded by $O(n^3L)$, carried on with the same precision as that in Karmarkar's algorithm. Each iteration updates a penalty multiplier and solves a Newton-Raphson iteration on the traditional logarithmic barrier function using approximated Hessian matrices. The resulting sequence follows the path of optimal solutions for the penalized functions as in a predictor-corrector homotopy algorithm.

*On leave from COPPE-Federal University of Rio de Janeiro, Cx. Postal 68511, 21941 Rio de Janeiro, RJ, Brasil
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-0602, AFOSR grant 83-0361 and State of California Microelectronics Innovation and Computer Research Opportunities Program and General Electric.

1 Introduction

The complexity of linear programming problems was lowered to $O(n^{3.5}L)$ arithmetic operations by Karmarkar [10], beginning a new cycle in optimization research. The algorithm is based on solving an equivalent problem that has as objective his "logarithmic potential function", which was immediately recognized as a penalized objective function, his method being a smooth barrier function algorithm.

Each iteration of Karmarkar's algorithm is a steepest descent search with scaling, as was first noted in [15] and carefully formalized in [7]. This brought linear programming algorithms into the realm of nonlinear programming techniques, where smooth penalty function methods were then out of fashion due to the ill-conditioning that often happens near an optimal solution.

The surprisingly good properties of Karmarkar's algorithm gave rise to investigations in several directions, reviving common procedures in nonlinear programming. The simplest of all is the projected gradient algorithm with scaling, applied to linear programming in [17], with amazing computational results. The logarithmic potential function is not convex. Its antilogarithm, the multiplicative potential function, is convex and can consequently be minimized by a Newton-Raphson algorithm. This was done by Iri and Imai [9], who proved asymptotic quadratic convergence. Their algorithm is actually equivalent to Karmarkar's method, as we prove in [8], both being quadratically convergent.

Other variants of Karmarkar's algorithm were presented in [1] and [7]. A generalized algorithm using bi-directional searches combines properties of all previous projective and affine variants [8].

A second stage in this line of research led to the study of trajectories followed by several algorithms. The "yellow brick road" to an optimal solution, which will be the object of this paper, is the "center trajectory", described by Megiddo [12] and by Bayer and Lagarias [2].

The breakthrough was obtained by Renegar [14], who used an approach based on methods of centers to follow the center trajectory. He achieved for the first time a speed of convergence leading to a solution of the problem in $O(n^{0.5}L)$ iterations, but each iteration needs $O(n^3L)$ computations, with a total figure of $O(n^{3.5}L)$ computations, the same as in Karmarkar's algorithm. The same approach was followed by Vaidya [16], who proved a complexity bound equivalent to ours, obtained simultaneously and independently.

Our result is also based on following the center trajectory, but using a barrier function approach. The final result is a practical algorithm for which

we preview good computational characteristics. The algorithm follows the trajectory by a predictor–corrector scheme and has the following advantages over the existing methods: there is no need of knowing the value of an optimal solution, and no lower bounds to an optimal solution are used. No special format for the problem is needed, with the exception of the first iteration that needs a point near the center. No projective transformations are used. Finally, the algorithm is well adapted to use large steps instead of the small steps needed for the convergence proofs, which is usually not true for methods of centers.

Our algorithm follows the path of optimizers for penalized problems using the logarithmic barrier function. The same results could be obtained by using Karmarkar's potential function or the multiplicative potential: the curves would then be parameterized by the values of the lower bounds v in the expression

$$f_v(x) = n \log(c'x - v) - \sum_{i=1}^{n} \log x_i$$

We chose the barrier function because the mathematical treatment is simpler, and so are the resulting procedures.

Barrier function methods and homotopies: Logarithmic barrier function methods were introduced in [4] and systematically formalized in [3]. A small-step barrier function method tries to follow the path of optimizers for the penalized functions very closely, staying always comfortably in the region of convergence for Newton method. This characterizes the method as a path-following procedure in the homotopy approach. Specifically, the barrier method is a predictor-corrector algorithm with the simplest of all schemes: each iteration is composed of an *elevator* step (decrease in the penalty parameter) followed by a Newton correction (internal minimization).

Garcia and Zangwill have an extremely clear presentation of homotopies and path-following algorithms in [5]. They point out that the approach has been known since the nineteenth century, and that it has been rediscovered several times. We shall try not to invent the methods once more, but we shall not profit much from the existing theory. This is so because in complexity studies all *epsilons* and *deltas* must be carefully bounded in opposition to what is done when deriving differential properties of the paths.

The homotopy approach to barrier functions is also old, and the idea of following the path of optimizers is already present in [3]. An important reference for this study is the book by Lootsma [11], and recent results for linear programming problems were published by Megiddo [12]. The

logarithmic barrier function approach with large steps was already applied to linear programming in [6].

The approach to be followed in this paper can be considered either as a barrier function method or as a predictor-corrector algorithm. Instead of trying to adapt known results to our needs, we decided to write a self-contained paper, and prove all results, excepting the complexity of the projection operation needed at each iteration, proved by Karmarkar [10].

1.1 The problem

We shall use a very general format for the linear programming problem, working on a bounded feasible set in the first orthant. One additional assumption must be made with the only purpose of finding an initial penalty multiplier, and the easiest way to achieve it is to impose the presence of a simplex constraint. Adding a simplex constraint to a given problem is easy, as we comment ahead. We shall also study scaled problems, that have the same format as the original one, but without the simplex constraint.

Consider the following format for all linear programming problems to be studied in this paper:

$$\begin{array}{lrcl} \textit{minimize} & c'x & & \\ \textit{subject to} & Ax & = & b \\ & x & \geq & 0 \end{array} \tag{1}$$

where $b, c, x \in I\!R^n$, A is an $m \times n$ matrix, $m < n$. We assume that the feasible set is bounded.

Our algorithm will always work in the relative interior of the feasible set, and the following notation will be used:

$$\begin{array}{rcl} S & = & \{x \in I\!R^n | Ax = b, x > 0\} \\ Q & = & \{x \in I\!R^n | Ax = b\} \\ D & = & \{x \in I\!R^n | Ax = 0\} \\ I\!R^n_+ & = & \{x \in I\!R^n | x > 0\} \end{array} \tag{2}$$

S is the relative interior of the feasible set, Q is its affine hull and $D = Null(A)$ is the set of feasible directions from any point in S . $I\!R^n_+$ will denote the interior of the first orthant.

The problem: The problem to be solved in this paper is (1), with two additional assumptions: (i) that the simplex constraint is used, that is,

for any feasible x , $e'x = n$; (ii) $e = [1\ 1\ \ldots 1]'$ is a non-optimal feasible solution. If an initial feasible point is know, then these conditions can be easily obtained for an arbitrary problem (1) by means of a scaling, the introduction of a constraint of the form $e'x \leq M$, where M is a large number and two new variables. The complete procedure is detailed in Todd and Burrell [15].

The role of the simplex constraint: it is important to point out that the simplex constraint will be used at only one point, in lemma 4.1, to choose the initial penalty multiplier. With the exception of the first iteration of the algorithm, the simplex constraint is not used at all, and nothing like Karmarkar's "projective transformation" is needed.

Projection matrices: We shall denote by P the projection matrix onto D , or equivalently onto the feasible set. The computation of this matrix for the scaled problems that we shall use is studied in [10].

1.2 Outline of the algorithm

A barrier method breaks the constrained problem (1) into a sequence of non-linear programming problems with the format

$$(P_k) \qquad \min_{x \in S} c'x - \epsilon_k \sum_{i=1}^{n} \log x_i \qquad (3)$$

where ϵ_k are positive *penalty multipliers* such that $\epsilon_k \to 0$. Problems (P_k) are constrained only by linear equality constraints and can thus be solved by algorithms for unconstrained minimization.

Following the methodology in Polak [13], we begin by studying properties of a *conceptual* algorithm, and then evolve into an *implementable* method, followed by a *practical* algorithm.

A conceptual barrier function algorithm assumes that an exact solution x^k can be found for each (P_k). In this case it is well known that any accumulation point of the sequence (x^k) is an optimal solution for (1). The speed of convergence is dictated by the speed with which the multipliers converge to zero.

An implementable algorithm does not assume exact solutions. Each penalized problem is approximately solved and generates a point z^k that must be near the unknown x^k . This vector z^k will be the starting point for (P_{k+1}). If subsequent points x^k are near each other (in some sense to

be explained later), then one Newton-Raphson iteration provides the good approximations that we need.

We begin by studying the conceptual algorithm and show that "short steps" are obtained by using a *fixed* sequence $\epsilon_k = (1-\sigma)^k \epsilon_0$, where ϵ_0 is an initial penalty multiplier and $\sigma = 0.005/\sqrt{n}$. At this point it is already possible to prove the bound of $O(n^{0.5}L)$ for the number of iterations. This will be done in section 3.

The next step, to be done in section 4, is to show that good approximations to the conceptual solutions are obtained by scaling each penalized problem (P_k) about z^k and solving one Newton Raphson iteration to generate z^{k+1}. A further improvement in complexity is obtained by using approximated Hessian matrices in these computations. A low complexity procedure to achieve this is presented, as well as a method to compute the initial penalty multiplier.

Before we start studying the conceptual algorithm, section 2 will be dedicated to listing some basic results on barrier functions and scalings.

All action will take place in $\mathbb{R}^n$. We shall denote vectors by lower case letters, matrices by upper case letters. The transpose of a matrix A will be denoted by A'. The vector with components x_i will also be denoted by $[x_i]$, and the letter e will be reserved for the unit vector $e = [1\,1 \ldots 1]'$. To each vector z^k, an upper case Z_k will denote the diagonal matrix $diag(z_1^k, z_2^k, \ldots, z_n^k)$, to be used in scaling operations. The norms 1,2,3 and ∞ for $\mathbb{R}^n$ will be denoted by $\|\cdot\|_1$, $\|\cdot\|$, $\|\cdot\|_3$ and $\|\cdot\|_\infty$, respectively, and other norms will be defined in section 2.

2 Scalings, barrier functions , quadratic approximations

This section concentrates basic results related to logarithmic barrier functions and their quadratic approximations. They are mostly "common knowledge" results, and our purpose is to present the notation and specialize the results to our needs.

2.1 Scalings

We shall be working with the affine set Q defined in (2) and with the relative interior of its restriction to the first orthant S. We assume that S is nonempty. Given a point $\bar{x} \in \mathbb{R}^n_+$, we define a *scaling* about $\bar{x}$ as a change of coordinates $x = \bar{X}y$, where $\bar{X} = diag(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$. Notice that

since $\bar{x} > 0$, the transformation is well defined, and it defines new norms $\|\cdot\|_{\bar{x}}$ for $\mathbb{R}^n$, such that for all $z \in \mathbb{R}^n$,

$$\begin{aligned} \|z\|_{\bar{x}} &= \|\bar{X}^{-1}z\| \\ \|z\|_{\bar{x}}^{\infty} &= \|\bar{X}^{-1}z\|_{\infty} \end{aligned} \tag{4}$$

Given a point $x \in \mathbb{R}^n_+$ and the linear programming problem (1), scaling about x produces an equivalent problem

$$\begin{aligned} \textit{minimize} \quad & (Xc)'y \\ \textit{subject to} \quad & (AX)y = b \\ & y \geq 0 \end{aligned} \tag{5}$$

For this problem $y = e$ corresponds to x . If $x \in S$, then $y = e$ is feasible for the scaled problem.

Now define $\bar{A} = AX$, $\bar{c} = Xc$. The scaled linear programming problem will then be

$$\begin{aligned} \textit{minimize} \quad & \bar{c}'y \\ \textit{subject to} \quad & \bar{A}y = b \\ & y \geq 0 \end{aligned} \tag{6}$$

This problem has the same format as (1). We shall denote by $\bar{S}$, $\bar{Q}$, $\bar{D}$ the sets corresponding respectively to S , Q , D in (1).

Dealing with the scaled problem will need the projections of the vectors $\bar{c}$ and e onto $\bar{D}$, and this is the most time-consuming operation of all necessary to algorithms based on scaling.

Scaling affects the steepest descent direction in an optimization algorithm, and it can be used to improve the performance of first-order feasible direction methods. This is the case with Karmarkar's algorithm, which computes at each iteration a steepest descent direction for his "logarithmic potential function". On the other hand, Newton-Raphson method is scale-invariant, and no change in its performance can be gained by such coordinate changes. In our approach, scaling will be used with the purpose of simplifying the mathematical treatment.

Our first lemma asserts that scalings about points "near" each other have similar effects on the scaled norms. In this lemma proximity will be measured in the norm of the sup, and is obviously valid for any other standard norm for $\mathbb{R}^n$.

Lemma 2.1 *Let $a, b \in \mathbb{R}^n_+$ be such that $\|a - b\|_a^{\infty} \leq 0.1$. Then given any $h \in \mathbb{R}^n$,*

$$0.9 \leq \frac{\|h\|_a}{\|h\|_b} \leq 1.1$$

Proof: Let

$$h_a = \left[\frac{h_i}{a_i}\right] = \left[\frac{h_i}{b_i}\frac{b_i}{a_i}\right] \quad .$$

Note that $\|h_a\| = \|h\|_a$, and similarly $\|h_b\| = \|h\|_b$. Consequently,

$$\|h_a\|^2 = \sum_{i=1}^{n} (\frac{h_i}{b_i})^2 (\frac{b_i}{a_i})^2 \geq (\frac{b_i}{a_i})^2_{min} \|\frac{h_i}{b_i}\|^2$$

where $\quad (\frac{b_i}{a_i})_{min} = \min_{i=1,\ldots,n} |\frac{b_i}{a_i}| \geq 1 - 0.1 \ , \quad$ since $|1 - \frac{b_i}{a_i}| \leq 0.1$

It follows that

$$\|h_a\| \geq 0.9 \|h\|_b .$$

The other inequality is proved similarly, using

$$\|h_a\| \leq (\frac{b_i}{a_i})_{max} \|h\|_b \ .$$

In the lemma above we used for the first time the criterion $\|z - x\|_x < \delta$ to assert that the points x and z are "near" each other. This means that scaling the problem about x, the point corresponding to z will be near e. The next lemma shows that if measured by the cost function, "near" is really near.

Lemma 2.2 *Let $x, z \in S$ be such that $\|x - z\|_x \leq \delta < 1$ in problem (1), and let v^* be the cost of an optimal solution for the problem. Then $c'z - v^* \leq (1 + \delta)(c'x - v^*)$*

Proof: Consider the problem scaled about x, as in (6). Define $z_x = X^{-1}z$. Then $\bar{c}'e = c'x$ and $\bar{c}'z_x = c'z$. Let

$$\alpha = \max_{h \in D} \{\bar{c}'h | \|h\| \leq 1\}$$

Since $\{e + h | \|h\| < 1, h \in \bar{D}\} \subset \bar{S}$,

$$v^* \ \leq \ \bar{c}'e - \alpha \ = \ c'x - \alpha \qquad , \text{or}$$

$$\alpha \leq c'x - v^* \ .$$

On the other hand,

$$c'z \ = \ \bar{c}'z_x \ \leq \ \bar{c}'e + \delta\alpha \ = \ c'x + \delta\alpha.$$

Merging the last two inequalities,

$$c^t z \leq c^t x(1+\delta) - \delta v^*$$

Subtracting v^* from both sides,

$$c^t z - v^* \leq (1+\delta)(c^t x - v^*)\,,$$

completing the proof.

2.2 Barrier functions

We shall study penalized functions for problem (1) We do not assume that the simplex constraint is present, but assume instead that the set S is bounded. This includes problem (6), and thus covers all cases to appear. The logarithmic barrier function is defined as

$$x \in I\!R^n_+ \longmapsto p(x) = -\sum_{i=1}^{n} \log x_i \tag{7}$$

Given a positive constant ϵ, a *penalized* objective function is defined by

$$x \in I\!R^n_+ \longmapsto f_\epsilon(x) = c^t x + \epsilon p(x) = c^t x - \epsilon \sum_{i=1}^{n} \log x_i \tag{8}$$

For any $\epsilon > 0$, $f_\epsilon(\cdot)$ is well defined, it is analytic and strictly convex and has a global minimum over S .

Penalized problem: Given $\epsilon > 0$, the penalized problem associated to the linear programming problem (1) is :

$$\min_{x \in S} f_\epsilon(x) \tag{9}$$

The Newton-Raphson step: Consider the penalized problem (9) and a given point $x^0 \in S$. A N-R step for this problem is a direction $h_N \in D$ that solves the problem

$$\min_{h \in D} f_N(x^0 + h)\ , \tag{10}$$

where $f_N(\cdot)$ is the quadratic approximation to $f_\epsilon(\cdot)$, given by

$$h \in D \mapsto f_N(x^0 + h) = f_\epsilon(x^0) + \nabla f_\epsilon(x^0)^t h + \frac{1}{2}\, h^t \nabla^2 f_\epsilon(x^0) h \tag{11}$$

Both (9) and (10) are well defined, since $f_\epsilon(\cdot)$ grows indefinitely near the boundary of S and $f_N(\cdot)$ is quadratic with positive definite Hessian, as we shall see in the follow.

We now have listed all problems to be studied in the paper. The LP problem (1) will be solved by a sequence of penalized problems (9). Approximate solutions for these will be found by solving (10). The last link in this chain of simplifications will be introduced later by allowing well-bounded errors in the computation of the Hessian matrices.

Lemma 2.3 *The penalized function (8) has the values and derivatives below, and $\nabla^2 f_\epsilon(x)$ is positive definite for any $x \in \mathbb{R}^n_+$.*

$$\begin{aligned} f_\epsilon(e) &= c'e & & \\ \nabla f_\epsilon(e) &= c - \epsilon e & \nabla f_\epsilon(x) &= c - \epsilon[x_i^{-1}] \\ \nabla^2 f_\epsilon(e) &= \epsilon I & \nabla^2 f_\epsilon(x) &= \epsilon X^{-2} \end{aligned} \qquad (12)$$

Proof: Straightforward.

We must now examine how well the quadratic approximation to the penalized functions approximates its actual values. The result comes as a consequence of properties of the logarithm function.

Lemma 2.4 *Given a direction $h \in \mathbb{R}^n$ such that $\|h\| < 1$, the barrier function (7) can be written as*

$$\begin{aligned} p(e+h) &= -e'h + \frac{\|h\|^2}{2} + o(h), \qquad \text{where} \qquad (13) \\ |o(h)| &\leq \frac{\|h\|^3}{3(1-\|h\|)} \end{aligned}$$

Proof: We start by writing a quadratic approximation for the logarithm on the real line:

$$\log(1+\lambda) = \lambda - \frac{\lambda^2}{2} + \frac{\lambda^3}{3} - \ldots \qquad (14)$$

The error of the quadratic approximation for $\lambda < 1$ is given by

$$|\delta(\lambda)| = \left| \sum_{i=3}^{\infty} \frac{\lambda^i (-1)^{i+1}}{i} \right| \leq \sum_{i=3}^{\infty} \frac{|\lambda|^i}{3} = \frac{|\lambda|^3}{3(1-|\lambda|)}$$

If we now consider a direction h with $\|h\| < 1$, and consequently $|h_i| < 1$ for $i = 1, 2, \ldots, n$,

$$\begin{aligned} p(e+\lambda) &= -\sum_{i=1}^{n} \log(1+h_i) \\ &= -\sum_{i=1}^{n} h_i + \sum_{i=1}^{n} \frac{h_i^2}{2} + \sum_{i=1}^{n} \delta(h_i) \\ &= -h'e + \frac{\|h\|^2}{2} + o(h) \end{aligned}$$

$$\text{where } |o(h)| \leq \sum_{i=1}^{n} \frac{|h_i|^3}{3} \frac{1}{1-|h_i|} \leq \frac{1}{1-\|h\|} \sum_{i=1}^{n} \frac{|h_i|^3}{3}$$

$$\text{But } \sum_{i=1}^{n} |h_i|^3 = \|h\|_3^3 \leq \|h\|^3$$

by the ordering of norms in $\mathbb{R}^n$, and this completes the proof.

This lemma gives us good bounds on the third order errors for the barrier function about the point e . The corresponding errors for the penalized functions (8) will then be given by $-\epsilon o(h)$. Our next step is to estimate how well a minimizer of the quadratic approximation approximates the minimizer of the penalized function.

To simplify the notation, we shall work with the function

$$x \in \mathbb{R}^n, x > 0 \longmapsto g(x) = \gamma' x + p(x) \;=\; f_\epsilon(x)/\epsilon \tag{15}$$

The function $g(\cdot)$ is equivalent to $f_\epsilon(\cdot)$ up to a multiplicative constant. Using (12), define its quadratic approximation about e as

$$h \in \mathbb{R}^n, e+h > 0 \longmapsto g_N(e+h) = \gamma'(e+h) - e'h + \frac{1}{2}\|h\|^2, \tag{16}$$

By lemma 2.4, for $e + h \in S$,

$$g(e+h) = g_N(e+h) + o(h)$$

Let the penalized problem and the N-R step be defined as usual by

$$\begin{aligned} \hat{x} = e + \hat{h} &= argmin\{g(x) | x \in S\} \\ x_N = e + h_N &= argmin\{g_N(x) | x \in Q\} \end{aligned} \tag{17}$$

The following lemma asserts that when $\hat{x}$ approaches e , the precision with which x_N estimates $\hat{x}$ increases.

Lemma 2.5 *Consider* $\hat{h}$ *and* h_N *defined as above, and let* $\alpha > 0$ *be given. Then there exists* $\gamma > 0$ *such that if either* $\|\hat{h}\| \leq \gamma$ *or* $\|h_N\| \leq \gamma$ *then* $\|\hat{h} - h_N\| \leq \alpha\gamma$

Proof: We shall study the limitation in $\|\hat{h} - h_N\|$ for small values of $\hat{h}$ and h_N.

First part: assume that $\hat{h} \leq 0.1$. Since we cannot assume that h_N is also small (although this is true), let us examine points in the segment joining $\hat{x}$ and x_N. Since $g_N(\cdot)$ is minimized at x_N and $\nabla^2 g_N(x_N) = I$, $g_N(\cdot)$ restricted to this segment is a parabola with minimum at x_N , and for any point z on this segment,

$$g_N(\hat{x}) \geq g_N(z) + \frac{1}{2}\|\hat{x} - z\|^2 \tag{18}$$

Choose such a point z satisfying $\|\hat{x} - z\| \leq \|\hat{h}\|$ and define $d = z - \hat{x}$. Then

$$\begin{aligned} g_N(\hat{x}) &\leq g(\hat{x}) + |o(\hat{h})| \\ &\leq g(z) + |o(\hat{h})| \qquad \text{by definition of } \hat{x} \\ &\leq g_N(z) + |o(\hat{h})| + |o(\hat{h} + d)| \end{aligned} \tag{19}$$

Subtracting (18) from (19),

$$\begin{aligned} \frac{1}{2}\|d\|^2 &\leq |o(\hat{h})| + |o(\hat{h} + d)| \\ &\leq \frac{\|\hat{h}\|^3}{3(1 - \|\hat{h}\|)} + \frac{\|\hat{h} + d\|^3}{3(1 - \|\hat{h} + d\|)} \end{aligned} \tag{20}$$

Now define μ by $\|d\| = \mu\|\hat{h}\|$, $\mu \leq 1$. Since the denominators in (20) are greater than 0.8 ,

$$\frac{1}{2}\mu^2 \leq \frac{1}{2.4}[1 + (1 + \mu)^3]\|\hat{h}\| \leq \frac{9}{2.4}\|\hat{h}\| ,$$

since $\mu < 1$. We conclude that μ decreases with $\sqrt{\|\hat{h}\|}$, completing the first part of the proof.

Second part: we must now consider the case in which $\|h_N\|$ is known to be small. The proof is identical , unless for the choice of z near x_N instead of near $\hat{x}$. All relations above are then satisfied for z substituting $\hat{x}$ and x_N substituting z .

It is interesting to check some numerical values for expression (20). For $\|\hat{h}\|$

very small, the relationship tends to $\mu^2 \leq (4/3)\|\hat{h}\|$, or $\|d\| \leq 1.16\|\hat{h}\|^{1.5}$, obtained by taking limits. The following values will prove useful, and can be directly verified:

Lemma 2.6 *Let $\hat{x}$ and x_N be defined as in (17). Then:*
If $\|x_N - e\| \leq 0.02$ then $\|\hat{x} - x_N\| \leq 1.4\|x_N - e\|^{1.5} \leq 0.004$.
If $\|\hat{x} - e\| \leq 0.025$ then $\|\hat{x} - x_N\| \leq 0.005$.

2.3 Computation of a Newton-Raphson step

We now show how to solve for the N-R step in (10). Let P be the projection matrix onto D , and set $c_p = Pc$, $e_p = Pe$.

Lemma 2.7 *Consider the N-R step problem (10) for given $\epsilon > 0$, $x^0 \in S$. The N-R step h_N satisfies*

$$PX_0^{-2}h_N = -\frac{c_p}{\epsilon} + PX_0^{-1}e \ . \tag{21}$$

In particular, for $x^0 = e$,

$$h_N = -\frac{1}{\epsilon}P\nabla f_\epsilon(e) = -\frac{c_p}{\epsilon} + e_p \ . \tag{22}$$

Proof: As we saw before, h_N is well defined. At $x^0 + h_N$ we must have

$$P\nabla f_N(x^0 + h_N) = 0$$

or, since f_N is quadratic,

$$P(\nabla f_\epsilon(x^0) + \nabla^2 f_\epsilon(x^0)h_N) = 0 \tag{23}$$

Using (12),

$$P(c - \epsilon X_0^{-1}e + \epsilon X_0^{-2}h_N) = 0$$

This expression is equivalent to the desired result. For $x_0 = e$, $X_0 = I$, (21) becomes

$$Ph_N = -\frac{c_p}{\epsilon} + e_p \ .$$

The proof is completed by noting that since $h_N \in D$, $Ph_N = h_N$.

Expression (22) resumes what is well known in nonlinear programming: scaling about x^0 corresponds to a change of metric in which the steepest descent coincides with the N-R direction. In other words: the change of coordinates

corresponding to the scaling operation transforms the ellipsoidal level sets for $f_N(\cdot)$ into spheres.

Approximate Hessians: Our last result in this section will be an account of the effect of errors in the Hessian. We shall consider errors of the following kind: given $x^0 \in S$ and $\tilde{x}$ near x^0 ($\tilde{x}$ not necessarily in S), the N-R iteration from x^0 will be computed using the wrong values $\nabla^2 f_\epsilon(x^0) \leftarrow \epsilon \tilde{X}^{-2}$ instead of $\nabla^2 f_\epsilon(x^0) = \epsilon X_0^{-2}$. In terms of the algorithm to be studied in this paper, this amounts to not updating the Hessian matrices in every iteration: only entries with a large error will be changed. This will give rise to rank-1 updates in the solution of the N-R steps instead of full matrix inversions.

Note that saving time by approximating the Hessians is current practice in predictor-corrector algorithms, and again we are using traditional procedures.

It is easy to see that by scaling the problem, it is enough to consider the case $\tilde{x} = e$, and the error in the Hessian reduces to $\nabla^2 f_\epsilon(x^0) \leftarrow \epsilon I$. This has a slightly different interpretation: instead of rescaling the problem at every iteration about x^k , the scaling is done about a point near x^k . This will be discussed in depth in section 4.

The next lemma relates errors in the Hessian to errors in the resulting N-R step.

Lemma 2.8 *Consider the N-R step problem (10) for given $\epsilon > 0$, $x \in S$. Let h_N be the N-R step computed by (21) and define*

$$\tilde{h}_N = -\frac{1}{\epsilon} P \nabla f_\epsilon(x) = -\frac{c_p}{\epsilon} + PX^{-1}e \ . \tag{24}$$

If $\|x - e\|_\infty \leq 0.1$ then $\|h_N - \tilde{h}_N\| \leq 0.235 \|h_N\|$.

Proof: Using (23),

$$P \nabla f_\epsilon(x) + \epsilon P X^{-2} h_N = 0 \qquad \text{, or}$$

$$P X^{-2} h_N \ = \ -\frac{1}{\epsilon} \nabla f_\epsilon(x) \ = \ \tilde{h}_N$$

But by hypothesis for $i = 1, \ldots, n$

$$x_i^{-2} = \frac{1}{(1+\nu_i)^2} \qquad \text{where} \quad |\nu_i| \leq 0.1 \ ,$$

and consequently, as can easily be verified,

$$x_i^{-2} = 1 + \mu_i \qquad \text{where} \quad |\mu_i| \leq 0.235 \ .$$

Setting $M = diag(\mu_i)$, it follows that

$$\tilde{h}_N \ = \ PX^{-2}h_N \ = \ PIh_N + PMh_N \ = \ h_N + PMh_N \ ,$$

Consequently,

$$\|h_N - \tilde{h}_N\| \ = \ \|PMh_N\| \ \leq \ \|Mh_N\| \ \leq \ 0.235\|h_N\| \ ,$$

completing the proof.

To finish this section, we shall state a new version of the result above, in a format that will show useful ahead.

Lemma 2.9 *Consider the N-R step problem (10) for given $\epsilon > 0$, $z^k \in S$, and let h_N be the N-R step computed by (21). Let $\tilde{z} \in \mathbb{R}^n_+$ be a point such that $\|z^k - \tilde{z}\|^{\infty}_{\tilde{z}} \leq 0.1$, and define $\tilde{h}_N$ as the result of the N-R step problem with Hessian Z_k^{-1} approximated by $\tilde{Z}^{-1}$. Then*

$$\|h_N - \tilde{h}_N\|_{z^k} \leq 0.26\|h_N\|_{z^k} \quad .$$

Proof: Scaling about $\tilde{z}$, the situation in lemma 2.8 is reproduced. The resulting directions satisfy

$$\|h_N - \tilde{h}_N\|_{\tilde{z}} \leq 0.235\|h_N\|_{\tilde{z}}$$

To translate the result to $\|\cdot\|_{z^k}$, all we need is to use lemma 2.1, obtaining

$$\|h_N - \tilde{h}_N\|_{z^k} \leq 0.235 \times 1.1 \leq 0.26 \quad ,$$

completing the proof.

We are now well equipped to study the barrier function method and its approximations.

3 The conceptual algorithm

Consider the linear programming problem (1). The conceptual barrier function method will use penalty multipliers $\epsilon_k = (1-\sigma)^k\epsilon_0$, where $\epsilon_0 > 0$ and $\sigma \in (0,1)$ are given. Each iteration solves exactly a minimization problem with criterion $f_k(x) = c'x + \epsilon_k p(x)$. The algorithm iterates until ϵ_k falls under a given precision $\delta > 0$. Notice that this is equivalent to fixing the number of iterations to a number K such that $\epsilon_0(1-\sigma)^K < \delta$.

Algorithm 3.1 *Conceptual barrier function: given* $\epsilon_0 > 0$, $\delta > 0$, $\sigma \in (0,1)$.

$k := 0$
Repeat
Compute a solution x^k to the problem

$$\min_{x \in S}(c'x - \epsilon_k \sum_{i=1}^{n} \log x_i) \qquad (P_k)$$

$\epsilon_{k+1} := (1-\sigma)\epsilon_k$
$k := k+1$
Until $\epsilon_k < \delta$

This section is dedicated to define "small steps" for the conceptual algorithm 3.1, and to find a value of the penalty adaptation parameter σ that guarantees such small steps.

3.1 Convergence of the conceptual algorithm

Let us begin by describing properties of the conceptual algorithm. Consider problem (1), and let x^* be an optimal solution, and call its value $v^* = c'x^*$.

Lemma 3.2 *The sequence* $(x^k)_{k \in N}$ *generated by algorithm 3.1 satisfies:*
i) For any given $x \in S$, $f_k(x) \to c'x$.
ii) $f_k(x^k) \to v^*$

Proof: (i) is immediate, since for any $x \in S$, $f_k(x) = c'x + \epsilon_k p(x)$, and $\epsilon_k \to 0$. To prove (ii), assume by contradiction that for all $k \in M \subset N$, $f_k(x^k) \geq v^* + 2\delta$, $\delta > 0$, where M is an infinite set.

Since any optimal solution is in the closure of S, there exists $x \in S$ such that $c'x < v^* + \delta$. It follows that for all $k \in N$,

$$f_k(x) \geq f_k(x^k) \geq v^* + 2\delta \geq c'x + \delta \ .$$

This contradicts (i), and completes the proof.

Lemma 3.3 *Let* x^k *be a point generated by algorithm 3.1. Then*

$$c'x^k - v^* \leq n\epsilon_k$$

Proof: From (12), $\nabla f_k(x^k) = c - \epsilon_k X_k^{-1} e$. Since x^k solves problem (P_k), $\nabla f_k(x^k)$ is orthogonal to any feasible direction. In particular, for the direction $h = x^k - x^*$,

$$\nabla f_k(x^k)'(x^k - x^*) = c'x^k - c'x^* + \epsilon_k e' X_k^{-1} x^* - \epsilon_k e' X_k^{-1} x^k = 0$$

But $X_k^{-1} x^k = e$, and then

$$c'x^k - v^* = \epsilon_k (n - e' X_k^{-1} x^*)$$

Since $x^* \geq 0$ and $x^k > 0$, it follows that $e' X_k^{-1} x^* \geq 0$, completing the proof.

Lemma 3.3 links the improvement in cost to the evolution of the sequence (ϵ_k). This means that costs $c'x^k$ converge to v^* at the same speed with which ϵ_k approaches zero. Our task is then to reduce these multipliers as fast as possible while keeping our "short step" strategy.

3.2 Differential properties: the homotopy approach

In the algorithm above we made an option for the barrier function approach, that will be kept in the remainder of the paper. We take a little pause now to make informal comments on the alternative homotopy framework.

If we define the homotopy mapping

$$x \in S, \epsilon > 0 \mapsto H(x, \epsilon) = P \nabla_x f_\epsilon(x) \ ,$$

the path of solutions to the penalized problem is defined by the homotopy

$$H(x, \epsilon) = 0 \ , \ x \in S \ , \ \epsilon > 0 \ .$$

It is easy to derive differential properties for the solution path, following reference [5] . Considering the path parameterized by ϵ , the homotopy equation $H(x(\epsilon), \epsilon) = 0$ can be differentiated, resulting in

$$\frac{\partial H}{\partial x}\frac{dx}{d\epsilon} + \frac{\partial H}{\partial \epsilon} = 0 \quad , \quad \frac{\partial H}{\partial x} = \epsilon P X^{-2} \quad , \quad \frac{\partial H}{\partial \epsilon} = -P X^{-1} e$$

Calculating these terms for $x = e$, we obtain

$$\epsilon P \frac{dx}{d\epsilon} = e_p \quad .$$

But for $x(\epsilon) \in S$,

$$P\frac{dx}{d\epsilon} = \frac{dx}{d\epsilon} \qquad \text{, and}$$

$$\frac{dx}{d\epsilon} = \frac{1}{\epsilon} e_p \quad .$$

This can be written as

$$\Delta x = \frac{\Delta\epsilon}{\epsilon} e_p + o(\Delta\epsilon)$$

where $o(\Delta\epsilon)$ is an error. Taking norms, it follows that

$$\|\Delta x\| \leq \frac{|\Delta\epsilon|}{\epsilon}\sqrt{n} + \|o(\Delta\epsilon)\| \; . \tag{25}$$

This is a very important result. It means that if we choose a small α and set at each iteration

$$\frac{\Delta\epsilon}{\epsilon} \leq \frac{\alpha}{\sqrt{n}} \quad ,$$

then $\|\Delta x\| \leq \alpha + \|o(\Delta\epsilon)\|$.

In other words, a small variation in x is associated to a variation in the penalty parameter (and thus in the objective) that depends on $\sqrt{n}$. Small variations in x mean good precision for the N-R steps; improvements dependent on $\sqrt{n}$ will lead us to the solution of the LP problem in $0(n^{0.5}L)$ iterations.

Unfortunately, complexity studies need precise bounds on $o(\Delta\epsilon)$, not provided by the analysis above. The rest of this section will be used to derive this result again, without taking any limits. The barrier function approach seems more adapted to this purpose.

3.3 The choice of ϵ_k

We saw in last section that the precision with which a Newton-Raphson iteration can approximate the minimizer of a penalized function starting from the point e depends only on how far the minimizer is from e . This gives us the clue to the small step: two consecutive minimizers x^k, x^{k+1} will be considered as near each other if $\|x^k - x^{k+1}\|_{x^k}$ is small. In other words, we want the solution of each penalized problem to be near e , when solving the problem scaled about x^k .

Assume that in iteration k of the conceptual algorithm 3.1 we know x^k exactly, then set $\epsilon_{k+1} = (1-\sigma)\epsilon_k$ and use a N-R iteration to find an approximation to x^{k+1} .

Instead of performing this in the original space, we shall scale the problem about x^k , obtaining problem (6). In this problem the vector e is feasible and corresponds to x^k in the original coordinates.

Objective function: the penalized function for the scaled problem will be

$$y \in \bar{S} \longmapsto g(y) = \bar{c}'y - \epsilon_{k+1} \sum_{i=1}^{n} \log y_i \tag{26}$$

Lemma 3.4 *The penalized function is scale-invariant in the sense that for any* $y \in \bar{S}$, $g(y) = f_{k+1}(X_k y) + K$, *where* K *is constant with* y .

Proof: Developing $f_{k+1}(\cdot)$,

$$\begin{aligned} f_{k+1}(X_k y) &= c' X_k y - \epsilon_{k+1} \sum_{i=1}^{n} \log x_i^k y_i \\ &= (X_k c)' y - \epsilon_{k+1} \sum_{i=1}^{n} \log y_i - \epsilon_{k+1} \sum_{i=1}^{n} \log x_i^k \end{aligned}$$

But $(X_k c)' y = \bar{c}' y$, and then setting $K = -\epsilon_{k+1} \sum_{i=1}^{n} \log x_i^k$ completes the proof.

This shows that minimizing (26) in $\bar{S}$ is equivalent to minimizing $f_{k+1}(\cdot)$ in S . We now proceed with the scaled problem.

Let $r(\cdot)$ be defined as the penalized function in $\bar{S}$ with penalty multiplier ϵ_k :

$$y \in \bar{S} \longmapsto r(y) = \bar{c}' y - \epsilon_k \sum_{i=1}^{n} \log y_i \tag{27}$$

Applying lemma 3.4, this function assumes a minimum over $\bar{S}$ at e , the point corresponding to x^k .

Setting $\epsilon_{k+1} = (1 - \sigma)\epsilon_k$ and computing derivatives by (12),

$$\begin{aligned} g(e) &= r(e) = \bar{c}' e \\ \nabla r(e) &= \bar{c} - \epsilon_k e \\ \nabla g(e) &= \bar{c} - (1 - \sigma)\epsilon_k e = \nabla r(e) + \sigma \epsilon_k e \end{aligned} \tag{28}$$

We are ready for the most important result which, like (25), relates σ to the distance from e to the minimizer of $g(\cdot)$. This will be obtained in two steps: we first deal with the minimizer of $g_N(\cdot)$ and then extend the result to the minimizer of $g(\cdot)$.

Lemma 3.5 *Let* $\delta \in (0, 0.1)$ *be a given constant. If* $\sigma \leq \delta/\sqrt{n}$, *then the N-R step* h_N *for* $g(\cdot)$ *from* e *satisfies* $\|h_N\| \leq \delta$.

Proof: Setting $\epsilon = \epsilon_{k+1}$ in the expression for the N-R step (22),

$$h_N = -\frac{1}{\epsilon_{k+1}} P \nabla g(e)$$

Taking the scalar product with h_N and noting that since h_N is feasible, $h_N' P y = h_N' y$ for any vector y ,

$$\|h_N\|^2 = -\frac{1}{\epsilon_{k+1}} h_N' \nabla g(e) \tag{29}$$

Now, using (28),

$$h_N' \nabla g(e) = \sigma \epsilon_k e' h_N + h_N' \nabla r(e)$$

But e minimizes $r(\cdot)$ over $\bar{S}$, and then for the feasible direction $h_N \in \bar{D}$, $h_N' \nabla r(e) = 0$. It follows that, merging the last equation into (29),

$$\|h_N\|^2 = -\sigma \frac{\epsilon_k}{\epsilon_{k+1}} e' h_N = -\frac{\sigma}{1-\sigma} e' h_N$$

Now, using the well-known relationship between norms 1 and 2,

$$-e' h_N \leq \|h_N\|_1 \leq \sqrt{n} \|h_N\|$$

It follows that

$$\|h_N\| \leq \frac{\sigma}{1-\sigma} \sqrt{n}$$

If we choose $\sigma \leq \delta/\sqrt{n} \leq 0.08$ for $n > 1$,

$$\|h_N\| \leq 1.1 \sigma \sqrt{n} \ \leq \ \delta \ ,$$

completing the proof.

Lemma 3.5 is almost what we need. To relate σ to the minimizer $\hat{y}$ of $g(\cdot)$ over $\bar{S}$ all we have to do is merge lemmas 3.5 and 2.6. We shall do that for numerical values that will prove convenient later: it does not make sense to try to push the stepsize to its maximum, since it does not affect the speed of convergence and the maximum stepsize is small anyway.

Theorem 3.6 *Let the penalty multipliers in 3.1 be adapted by* $\epsilon_{k+1} = (1 - \sigma)\epsilon_k$, *with* $\sigma = \delta/\sqrt{n}$. *If* $\delta \leq 0.005$, *then for all* $k \in N$, $\|x^k - x^{k+1}\|_{x^k} \leq 0.006$.

Proof: Consider the problem scaled about x^k as above. Assuming $\sigma \leq 0.005/\sqrt{n}$, then by lemma 3.5, $\|h_N\| \leq 0.005$.

Now, lemma 2.6 deals with the same scaled problem (with an objective function equivalent up to a product by a constant), and then

$$\|\hat{y} - e\| \leq \|h_N\| + \|\hat{y} - y_N\| \leq \|h_N\| + 1.4\|h_N\|^{1.5} \leq 0.006\,,$$

and the proof is complete.

Complexity of the conceptual algorithm: At this point we can advance that it will be possible to choose an initial penalty multiplier ϵ_0 bounded by 2^L. It is then immediate to compute the number of iterations necessary to obtain a reduction $\epsilon_k \leq \epsilon_0 2^{-L}$.

Lemma 3.7 *If the conceptual barrier function algorithm 3.1 uses an adaptation parameter $\sigma = \delta/\sqrt{n}$, then for $k \geq \sqrt{n}L/\delta$, $\epsilon_k \leq \epsilon_0 2^{-L}$.*

Proof: Let $k \geq \sqrt{n}L/\delta$. Taking logarithms,

$$\begin{aligned} \log \epsilon_k &= \log \epsilon_0 + k \log(1 - \delta/\sqrt{n}) \\ &\leq \log \epsilon_0 - k\delta/\sqrt{n} \qquad \text{by (14)} \\ &\leq \log \epsilon_0 - L \end{aligned}$$

it follows that $\epsilon_k \leq \epsilon_0 \exp(-L) \leq \epsilon_0 2^{-L}$, completing the proof.

In the next section we shall prove how to obtain the same speed of convergence with approximate computations: the total number of operations will then be given by the effort per iteration multiplied by $O(\sqrt{n}L)$.

4 The implementable algorithm

The implementable algorithm uses approximate Newton-Raphson searches instead of exact minimizations for the penalized functions. The points x^k will never be computed: a sequence (z^k) will be computed instead, and we shall make sure that each z^k is near the corresponding x^k. A convenient definition of "near" will be $\|z^k - x^k\|_{x^k} \leq 0.015$.

We shall begin by showing how to compute an initial penalty parameter ϵ_0 such that e (the initial point for the implementable algorithm) is near x^0 (the initial point for the conceptual algorithm). Then we develop a simplified algorithm model without specifying how to approximate the Hessian matrix and prove its efficiency in solving the linear programming problem. The

final step will be to specify the details on Hessian approximations. We shall use the notation described in sections 1 and 2.

As we pointed out before, scaling operations simplify the mathematical treatment, but are not necessary at all. The algorithm model will use scalings for this reason. It is easy to modify the algorithm so that it does not rely on scalings, and we shall indicate how to do it immediately after presenting the algorithm.

The initial penalty multiplier: Consider problem (1) and the penalized problems introduced in (3) and studied in section 2. Assume that the simplex constraint is enforced and that e is feasible. Note that this is the only point in the paper in which this constraint is used. It can also be of interest to note that we only need to know the point of minimum penalty in S, trivially equal to e if the simplex constraint is enforced.

Lemma 4.1 *Consider problem* (P_0)*, in the first iteration of the conceptual algorithm 3.1. If* $\epsilon_0 \geq \|c\|/0.01$ *then* $\|e - x^0\|_{x^0} \leq 0.015$.

Proof: In the first iteration, the N-R direction from e is computed by (22)

$$h_N = -\frac{c_p}{\epsilon_0} + e_p$$

But in the original linear programming problem the simplex constraint is in use and then $e_p = Pe = 0$. Eliminating this term and taking the scalar product with h_N ,

$$\|h_N\|^2 = -\frac{h_N' c_p}{\epsilon_0} \leq \frac{\|c_p\|}{\epsilon_0}\|h_N\|$$

Since $\|c_p\| \leq \|c\|$, it follows that

$$\|h_N\| \leq \frac{\|c\|}{\epsilon_0}$$

Using now $\epsilon_0 \geq \|c\|/0.01$,

$$\|h_N\| \leq 0.01$$

Using lemma 2.6, with $\hat{x} = x^0$ and $x_N = e + h_N$,

$$\|x^0 - e\| \leq \|h_N\| + \|x^0 - x_N\| \leq 0.01 + 0.002 \leq 0.012$$

Since $\|e - x^0\| \leq 0.09$, we can use lemma 2.1, obtaining

$$\|x^0 - e\|_{x^0} \leq 1.1\|x^0 - e\|_e \leq 0.015$$

since $\|x^0 - e\|_e = \|x^0 - e\|$, and this completes the proof.

The algorithm model: Consider the linear programming problem (1). Each iteration will perform a scaling operation, define a penalized function and perform a Newton-Raphson search. Instead of computing the exact N-R step, a point $\tilde{z}$ near the present iterate z^k will be *chosen* (this is the reason to call this a model: the determination of $\tilde{z}$ will be done later), and the Hessian at z^k will be approximated by the Hessian at $\tilde{z}$. As we saw in lemma 2.9, this is equivalent to a steepest descent step for the problem scaled about $\tilde{z}$ instead of about z^k.

Algorithm 4.2 *Model : given* $\sigma = 0.005/\sqrt{n}$, $\delta > 0$.

$k := 0$, $z^0 := e$
$\epsilon_0 := \|c\|/0.01$

Repeat

$\epsilon_{k+1} = (1 - \sigma)\epsilon_k$

Choose $\tilde{z}$ such that $\|\tilde{z} - z^k\|_{\tilde{z}}^{\infty} \leq 0.1$

Scale the problem about $\tilde{z}$ obtaining problem (6), and define the penalized problem

$$\min_{y \in S}(\bar{c}'y - \epsilon_{k+1} \sum_{i=1}^{n} \log y_i)$$

Compute the projection matrix $\bar{P}$ onto the feasible set and set

$$y := \tilde{Z}^{-1} z^k \qquad , \qquad Y := diag(y_i)$$

Compute an approximated N-R step from y , obtaining by (24)

$$\tilde{d}_N = -\frac{\bar{P}\bar{c}}{\epsilon_{k+1}} + \bar{P}Y^{-1}e$$

$\tilde{y}_N := y + \tilde{h}_N$

Return to the original space with $z^{k+1} = \tilde{Z}\tilde{y}_N$

$k := k + 1$

Until $\epsilon_k < \delta$

The algorithm can be modified not to mention scaling operations simply by saying 'Compute an approximated N-R step from z^k with Hessian approximated by $\tilde{Z}^{-2}$ '. In this case the rank-1 updates in the N-R step equations must be worked out. We shall not do it.

Now consider the sequence (x^k) that would be generated by the conceptual algorithm with the same penalty multipliers. The theorem to follow is the main result in this section.

Theorem 4.3 *At any iteration k of algorithm 4.2 , the following relations are satisfied:*

$$\begin{aligned} \|z^k - x^k\|_{x^k} &\leq 0.015 \\ \|z^k - z^{k+1}\|_{x^k} &\leq 0.04 \end{aligned}$$

Proof: The proof is done by induction. By lemma 4.1 ,

$$\|z^0 - x^0\|_{x^0} \leq 0.015.$$

Assume that at iteration k

$$\|z^k - x^k\|_{x^k} \leq 0.015\,.$$

By theorem 3.6, since $\sigma = 0.005/\sqrt{n}$,

$$\|x^{k+1} - x^k\|_{x^k} \leq 0.006$$

Adding the two last inequalities,

$$\|z^k - x^{k+1}\|_{x^k} \leq 0.021$$

By lemma 2.1,

$$\|z^k - x^{k+1}\|_{x^k} \leq 0.024 \qquad (30)$$

Now consider the (unknown) exact solution of a N-R step $\hat{z} = z_k + h_N$ and the approximate solution $z^{k+1} = z^k + \tilde{h}_N$ found by the algorithm. By lemma 2.6,

$$\|\hat{z} - x^{k+1}\|_{x^k} \leq 0.005 \qquad (31)$$

Adding the two last inequalities,

$$\|h_N\|_{x^k} \leq \|\hat{z} - x^{k+1}\|_{x^k} + \|x^{k+1} - z^k\|_{x^k} \leq 0.03$$

We can now use lemma 2.9 to obtain

$$\|z^{k+1} - \hat{z}\|_{x^k} = \|h_N - \tilde{h}_N\|_{x^k} \leq 0.25 \times 0.03 \leq 0.008$$

Adding this inequality to (31),

$$\|z^{k+1} - x^{k+1}\|_{x^k} \leq 0.013 \tag{32}$$

And finally, using again lemma 2.1,

$$\|z^{k+1} - x^{k+1}\|_{x^{k+1}} \leq 0.015 \quad ,$$

This proves the first relationship. The second one immediately obtained by adding (30) and (32), completing the proof.

The implementable algorithm generates a sequence (z^k) , each z^k near the corresponding conceptual x^k . By lemma 2.2, the costs of these pairs of vectors differ by a negligible amount. We are finally ready to extend the property found for the conceptual algorithm in section 3 to any algorithm in this model.

Theorem 4.4 *Consider algorithm 4.2 applied to the linear programming problem defined in section 1, with $\delta \leq 2^{-L}/1.015n$, where L is the total length of the input data. Then the algorithm terminates in $O(\sqrt{n}\,L)$ iterations with a feasible solution z^k such that $c^t z^k - v^* \leq 2^{-L}$*

Proof: The initial penalty multiplier satisfies $\epsilon_0 = \|c\|/0.01 \leq 100 \times 2^L$. The algorithm stops at iteration k such that $\epsilon_k \leq \delta$, or

$$\frac{\epsilon_k}{\epsilon_0} \leq \frac{2^{-2L}}{1.015n}$$

This reduction is of the order $O(2^L)$, and can be achieved in $O(\sqrt{n}\,L)$ iterations, by lemma 3.7.

Applying lemmas 2.2 and 3.3 to the final solution generated by the algorithm,

$$c^t z^k - v^* \;\leq\; 1.015(c^t x^k - v^*) \;\leq\; 1.015n\epsilon_k \;\leq\; 1.015n\delta \;\leq\; 2^{-L} \,.$$

completing the proof.

The choice of $\tilde{z}$: An obvious choice for $\tilde{z}$ is $\tilde{z} = z^k$. This choice causes one projection computation per iteration, with a bound of $O(n^3)$ operations. The algorithm then terminates with a bound of $O(n^{3.5}L)$ operations, the same as in the methods by Karmarkar and Renegar.

We shall lower this bound by saving in the projection computations. The intuitive argument is as follows: each iteration causes a move (in the scaled

problem) of $\|\delta\|$; each approximation for the Hessian is good in a "cubic" ball $\|w - y\|_\infty \leq 0.1$. Since the volume of a ball in the sup norm is related to the volume of a ball in euclidean norm by a factor of $\sqrt{n}$, it is reasonable to expect that this will be the relationship between the number of iterations and the number of complete recalculations of the projection matrix. This feature will be explored in the procedure below, in which a component of $\tilde{z}$ is changed only when it is too far from z^k .

In the first iteration, set $\tilde{z} := e$ and compute the projection matrix P by any method. For $k > 0$, use the algorithm below.

Algorithm 4.5 *updates of* $\tilde{z}$ *and* $\bar{P}$

Set

$$J := \{j = 1, \ldots, n \mid \frac{|z_j^k - \tilde{z}_j|}{\tilde{z}_j} \geq 0.1\}$$

For all $j \in J$ set

$$\tilde{z}_j := z_j^k$$

Compute the projection matrix $\bar{P}$ from the former one by $|J|$ rank-1 updates.

The projection matrix for each scaled problem is calculated by $\bar{P} = I - \tilde{Z}A'(A\tilde{Z}^2A')^{-1}A\tilde{Z}$. The main effort is in computing $(A\tilde{Z}^2A')^{-1}$. If only one entry of $\tilde{z}$ changes, this computation can be performed in $O(n^2)$ operations by a rank-1 update, as was established in [10]. This fact will be used in the next section to study the complexity of the algorithm.

5 Complexity of the Algorithm

Consider algorithm 4.2 with $\tilde{z}$ chosen as in 4.5. We shall now establish an upper bound on the total number of rank-1 updates computed by the algorithm. The updates are similar to the ones in Karmarkar's algorithm, but our proofs follow a different path.

Consider the sequence $(z^k)_{k=0,\ldots,K+1}$ generated by the algorithm, where K is the index of the last iteration, and the associated real sequences (z_j^k) for each component $j = 1, \ldots, n$. We begin by analysing a single component j .

To simplify the notation, let us denote $w \equiv z_j$, and deal with the sequence $(w^k) \equiv (z_j^k)$. Define the subsequence $(w^{k_i})_{i \in T_j}$ corresponding to the iterations in which $\tilde{z}_j$ is updated. T_j is the set of indices of such iterations,

and its cardinality $|T_j|$, the total number of updates, is the object of our study.

Consider now two consecutive indices k_i, k_{i+1}, with $i, i+1 \in T_j$. The following facts are true by construction:

$$\left| \frac{w^{k_{i+1}}}{w^{k_i}} - 1 \right| \geq 0.1 \quad ,$$

since at iteration k_{i+1} the procedure 4.5 updates $\tilde{z}_j$. Between two updates,

$$\text{for } k_i \leq k < k_{i+1} \quad , \quad \left| \frac{w^k}{w^{k_i}} - 1 \right| \leq 0.1 \quad .$$

Taking logarithms and defining $\mu = \min\{\log 1.1, -\log 0.9\} > 0.09$, we obtain for any two consecutive indices in T_j,

$$|\log w^{k_i} - \log w^{k_{i+1}}| \geq \mu \tag{33}$$

The next lemma is preparatory for the main result to follow.

Lemma 5.1 *For the sequences constructed above, for a component* $j = 1, \ldots, n$,

$$|T_j| \leq \frac{1.1}{\mu} \sum_{k=0}^{K} \frac{|w^{k+1} - w^k|}{w^k} \tag{34}$$

Proof: Consider initially any two indices $0 \leq k_1 < k_2 \leq K$. Then , manipulating the integral of the logarithmic function,

$$\begin{aligned} |\log w^{k_1} - \log w^{k_2}| &= \left| \int_{w^{k_1}}^{w^{k_2}} \frac{dw}{w} \right| \\ &\leq \sum_{k=k_1}^{k_2-1} \max \left\{ \frac{|\Delta w^k|}{w^k}, \frac{|\Delta w^k|}{w^{k+1}} \right\} \end{aligned}$$

where $\Delta w^k = w^{k+1} - w^k$.
The first equality is trivial, and the second is an immediate consequence of the definition of Riemann integral. Applying now lemma 2.1, since by theorem 4.3 $|\Delta w^k| \leq \|z^{k+1} - z^k\| < 0.1$,

$$\frac{|\Delta w^k|}{w^{k+1}} \leq 1.1 \frac{|\Delta w^k|}{w^k}$$

It follows that

$$|\log w^{k_1} - \log w^{k_2}| \leq 1.1 \sum_{k=k_1}^{k_2-1} \frac{|\Delta w^k|}{w^k}$$

In particular, for $k_1 = k_i$ and $k_2 = k_{i+1}$, with $i, i+1 \in T_j$, (33) gives

$$\mu \leq |\log w^{k_i} - \log w^{k_{i+1}}| \leq 1.1 \sum_{k=k_i}^{k_{i+1}-1} \frac{|\Delta w^k|}{w^k}$$

Finally, the summation of the complete sequence (which is composed of $|T_j|$ partial summations as above) gives

$$|T_j|\mu \leq 1.1 \sum_{k=0}^{K} \frac{|\Delta w^k|}{w^k} \quad ,$$

completing the proof.

Theorem 5.2 *Consider the sequence (z^k) generated by the algorithm and let $T = \sum_{j=1}^{n} |T_j|$ be the total number of rank-1 updates computed by the algorithm until iteration K. Then*

$$T \leq \sqrt{n}\, K$$

Proof: Applying lemma 5.1,

$$T = \sum_{j=1}^{n} |T_j| \leq \frac{1.1}{\mu} \sum_{j=1}^{n} \sum_{k=0}^{K} \left| \frac{\Delta z_j^k}{z_j^k} \right|$$

Inverting the order of the summations,

$$T \leq \frac{1.1}{\mu} \sum_{k=0}^{K} \left\| \left[\frac{\Delta z_j^k}{z_j^k} \right] \right\|_1$$

But for any $y \in \mathbb{R}^n$, norms 1 and 2 are related by $\|y\|_1 \leq \sqrt{n}\, \|y\|$, and consequently

$$T \leq \frac{1.1}{\mu} \sqrt{n} \sum_{k=0}^{K} \|\Delta z^k\|_{z^k}$$

Using now theorem 4.3, for any $k = 0, \ldots, K$, $\|\Delta z^k\|_{z^k} \leq 0.04$ and it follows that

$$T \leq \frac{0.044}{\mu} \sqrt{n}\, K \quad .$$

It is now enough to remember that $\mu > 0.09$ to complete the proof.

We can now prove the main complexity result.

Theorem 5.3 *Algorithm 4.2 with the updating procedure 4.5 solves the linear programming problem (1) in no more than $O(n^3L)$ arithmetic operations.*

Proof: We showed in theorem 4.4 that the algorithm terminates with an optimal solution in $K = O(\sqrt{n}L)$ iterations. Each iteration updates one projection matrix and computes a fixed number of matrix products.

The total number of operations needed per iteration excepting the projection matrix computation is then of the order $O(n^2)$, with a total figure of $O(n^{2.5}L)$ in K iterations.

The total number of operations needed for the projection updates is given by $T \times O(n^2)$, where T is the total number of rank-1 updates. By theorem 5.2, $T \leq n^{0.5}K$, and consequently T is bounded by $O(nL)$. This gives a total figure of $O(n^3L)$, completing the proof.

The linear algebra computations are identical to the ones in Karmarkar's method [10], and can be carried with L bits of precision, as is proved in [16]. This leads to a bound $O(n^3L^2)$ for the bit operations in our algorithm.

6 Conclusion

Towards a practical algorithm: To develop a practical algorithm, the short steps must be made adaptive. This can be easily done by examining the N-R step computations:

$$h = -\frac{c_p}{\epsilon_{k+1}} + d \qquad \text{where} \quad d = \bar{P}Y^{-1}e$$

To make this step adaptive, two things should be done:

(i) Use a line search along direction h instead of a fixed step, that is, solve

$$\min_{\alpha>0}\{f_{\epsilon_{k+1}}(y+\alpha h) | y + \alpha h \in \bar{S}\}$$

(ii) Instead of a fixed penalty multipler, let $\epsilon_{k+1} = \epsilon_k - \sigma$ and make σ vary in a "trust region" Γ .

Putting together these two procedures, we end with a bi-directional search

$$\min_{\alpha,\sigma>0}\{f_{\epsilon_k-\sigma}(y - \frac{\alpha c_p}{\epsilon_k - \sigma} + \alpha d) | y + \alpha h \in \bar{S}, \sigma \in \Gamma\}$$

The region Γ must be such as to guarantee that the quadratic approximations are good. Several criteria may be used, and the best seems to be the

following: a value of σ is acceptable if the search in α for this σ results in a value near 1 .

Euler's method: Since our method is a predictor-corrector algorithm, Euler's method immediately rings a bell. Instead of the simple elevator step (update of ϵ), we may follow a tangent to the path of optimizers. It is easy to see from section 3.2 that a tangent direction for $x = e$ is given by

$$\frac{dx}{d\epsilon} = \frac{1}{\epsilon} e_p \quad .$$

At the point y,

$$\frac{dy}{d\epsilon} = \frac{1}{\epsilon_k} d$$

and associating to each value of $\Delta\epsilon = -\sigma$

$$y_\sigma = y - \frac{\sigma}{\epsilon_k} d \quad ,$$

the bi-directional search becomes

$$\min_{\alpha,\sigma>0} \{ f_{\epsilon_k - \sigma}(y_\sigma + \alpha h_\sigma) | y_\sigma + \alpha h_\sigma \in \bar{S}, \sigma \in \Gamma \}$$

where h_σ can be computed as before, $h_\sigma = h$, or by a new projection (with the same projection matrix):

$$h_\sigma = -\frac{c_p}{\epsilon_k - \sigma} + d_\sigma \qquad \text{where} \quad d_\sigma = \bar{P} Y_\sigma^{-1} e$$

Notice that Euler's method does not introduce any new direction if h is kept unchanged, and no improvement in the result of a bi-directional search can be expected. The only modification to the basic bidirectional algorithm suggested by this procedure would be to recalculate d whenever the search procedure generates points far from the initial one.

This last observation falls into the idea developed in [7]: if the projection matrix is available at low computational cost, it is always advisable to perform steepest descent searches while they lead to good improvements in the function to be minimized. Summing up these observations, the best path towards a practical algorithm is in our opinion (and this must still be verified) the bidirectional search proposed above with recalculations of the vector d .

This bi-directional search is a variant of the general pattern followed by all conical projection algorithms, and will be the object of a forthcoming

paper [8].

All these new algorithms boil down to a clever rearrangement of old techniques. Karmarkar showed that Linear Programming is indeed a particular case of Nonlinear Programming, and that nice results can be obtained by specializing its methods to linear functions. Nothing is new, and all is new. Like the fresh meats and garden vegetables rethought by the French nouvelle cuisinne.

Acknowlegement: I would like to thank Prof. E. Polak, my host in Berkeley, for his advice and encouragement.

References

[1] K. Anstreicher. *A Monotonic Projective Algorithm for Fractional Linear Programming.* Manuscript, Yale School of Organization and Management, New Haven, CT, November 1985.

[2] K. Bayer and J. C. Lagarias. *The Non-linear Geometry of Linear Programming, I. Affine and Projective Scaling Trajectories, II. Legendre Transform Coordinates, III. Central Trajectories.* preprints, AT&T Bell Laboratories, Murray Hill, NJ, 1986.

[3] A. Fiacco and G. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, New York, 1955.

[4] K. R. Frisch. *The Logarithmic Potential Method of Convex Programming.* Memorandum, University Institute of Economics, Oslo, Norway, May 1955.

[5] C. B. Garcia and W. I. Zangwill. *Pathways to Solutions, Fixed Points, and Equilibria.* Prentice-Hall, Inc., New Jersey, 1981.

[6] P. Gill, W. Murray, M. Saunders, J. Tomlin, and M. Wright. *On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method.* Report SOL 85-11, Systems Optimization Laboratory, Dept. of Operations Research, Stanford University, Stanford, 1985.

[7] C. Gonzaga. *A Conical Projection Algorithm for Linear Programming.* Memorandum UCB/ERL M85/61, Electronics Research Laboratory, University of California, Berkeley, CA, July 1985.

[8] C. Gonzaga. *Generality of the Conical Projection Algorithm for Linear Programming.* In preparation, University of California, Berkeley, CA, 1987.

[9] M. Iri and H. Imai. A multiplicative penalty function method for linear programming - another "new and fast" algorithm. In *Proc. of the 6th. Mathematical Programming Symposium*, Tokio, Japan, November 1985.

[10] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[11] F. A. Lootsma. *Numerical Methods for Nonlinear Optimization.* Academic Press, New York, 1972.

[12] N. Megiddo. *Pathways to the Optimal Set in Linear Programming.* Research Report , IBM Almaden Research Center, S. Jose, California, 1986.

[13] E. Polak. *Computational Method in Optimization.* Academic Press, New York, 1971.

[14] James Renegar. *A Polynomial-time Algorithm Based on Newton's Method for Linear Programming.* Report MSRI 07 118 – 86, Mathematical Sciences Research Institute, Berkeley, California, June 1986.

[15] M. Todd and 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 1985.

[16] Pravin M. Vaidya. *An Algorithm for Linear Programming which Requires* $O(((m+n)n^2+(m+n)^{1.5}n)L)$ *Arithmetic Operations.* preprint, AT&T Bell Laboratories, Murray Hill, NJ, 1987.

[17] R. J. Vanderbei, M. J. Meketon, and B. A. Freedman. A modification of Karmarkar's linear programming algorithm. *Algorithmica*, 1, 1987. to appear.