Mathematical Programming 43 (1989) 257-276
North-Holland

AN INTERIOR POINT ALGORITHM FOR SEMI-INFINITE LINEAR PROGRAMMING

M.C. FERRIS* and A.B. PHILPOTT**
Cambridge University Engineering Department, Cambridge, England

Received 3 February 1986
Revised manuscript received 22 February 1988

We consider the generalization of a variant of Karmarkar's algorithm to semi-infinite programming. The extension of interior point methods to infinite-dimensional linear programming is discussed and an algorithm is derived. An implementation of the algorithm for a class of semi-infinite linear programs is described and the results of a number of test problems are given. We pay particular attention to the problem of Chebyshev approximation. Some further results are given for an implementation of the algorithm applied to a discretization of the semi-infinite linear program, and a convergence proof is given in this case.

Key words: Semi-infinite linear programming, discretizations, Karmarkar's method, Chebyshev approximation.

1. Introduction

Much recent attention has been paid to Karmarkar's Projective Algorithm [8] for linear programming and the rescaling algorithm discovered independently by Dikin [4], Barnes [2], Cavalier and Soyster [3], and Vanderbei et al. [11]. These algorithms solve linear programs by constructing a sequence of points lying in the interior of the feasible region and converging to the optimal solution. Both algorithms make use of a transformation of variables, followed by a step in the direction of an appropriate projected gradient.

In this paper we consider the generalization of this approach to solve linear programs posed over more abstract spaces. In particular, we consider a class of linear programs posed over a particular kind of pre-Hilbert space and describe a conceptual rescaling algorithm for members of this class. In order to show that this is not solely a theoretical exercise, we show that, for a class of semi-infinite linear programs, this approach can be applied in a straightforward manner to produce a rescaling algorithm for semi-infinite linear programming.

We begin by describing a generalization of the rescaling algorithm [11] to an abstract space. Let S be any set and let W be a space of functions from S to $\mathbb{R}^q$. It

* Now at Department of Computer Sciences, University of Wisconsin, Madison, WI, USA.
** Now at Department of Theoretical and Applied Mechanics, University of Auckland, Auckland, New Zealand.

is clear that with suitable definitions of addition and scalar multiplication that W can be given a vector space structure. Let U and Z be arbitrary vector spaces. Define the space V to be the Cartesian product of U and W, i.e.

$$V = U \otimes W.$$

We assume that V is a pre-Hilbert space with an inner product denoted by $\langle \cdot, \cdot \rangle$. Members of U are denoted by u, members of W are denoted by w, and members of V are denoted by either (u, w) or (x, z). The generalization of the rescaling algorithm which we describe below requires that the elements of W are regarded at different times to have either the norm induced by the inner product (the IP norm), or the supremum norm defined by

$$\|w\|_\infty = \sup\{\|w(s)\|_\infty \mid s \in S\}.$$

For example, the set of continuous functions on $[0, 1]$ can be regarded at different times as a pre-Hilbert space with inner product defined by

$$\langle f, g \rangle = \int_0^1 f(s)g(s)\, ds$$

or as the Banach space $C[0, 1]$ with the supremum norm. We define a convex cone W_+ as follows:

$$W_+ = \{w \in W:\ w_i(s) \geq 0,\ s \in S,\ i = 1, \ldots, q\},$$

and note that its interior W_+^0 (with respect to the supremum topology) is given by

$$W_+^0 = \{w \in W:\ w_i(s) > 0,\ s \in S,\ i = 1, \ldots, q\}.$$

We let

$$V_+ = U \otimes W_+$$

and

$$V_+^0 = U \otimes W_+^0$$

and define a partial order "$\geqslant$" on V by

$$\xi \geqslant \gamma \quad \text{for } \xi, \gamma \in V \quad \text{if and only if} \quad \xi - \gamma \in V_+.$$

If we let θ be the zero element of V then it is clear that

$$\xi \in V,\ \xi \geqslant \theta \quad \text{if and only if} \quad \xi \in V_+.$$

The set V_+ is called the positive cone of V. We define $\xi > \gamma$ for $\xi, \gamma \in V$ to mean that $\xi - \gamma \in V_+^0$, and we note that $\xi > \gamma$ implies that $\xi \geqslant \gamma$. We also assume that W_+^0 is non-empty and having chosen a fixed element e of W_+^0 let any $(u, e) \in V_+^0$ be called a *preferred element* of V.

In what follows we shall confine our attention to linear programs which have the following form.

$$\begin{aligned} \text{LP} \qquad \text{minimize} \quad & \langle (c, \boldsymbol{d}), (u, w) \rangle, \\ & T((u, w)) = \boldsymbol{b}, \\ \text{subject to} \quad & (u, w) \geqslant \theta, \\ & (u, w) \in V, \end{aligned}$$

with $c \in U$, $\boldsymbol{d} \in W$, $\boldsymbol{b} \in Z$, and $T: V \to Z$ being a continuous linear operator. For this class of infinite-dimensional linear programs we can specify an algorithm which is a generalization of the rescaling algorithm of [11]. At each step of the algorithm, we define a V_+^0-invariant non-singular endomorphism of V which maps the current point $(x, z) \in V_+^0$ to a preferred point $(x, \boldsymbol{e}) \in V_+^0$ which, with respect to the supremum norm, is "far away" from the boundary of the positive cone. We then take a step in the transformed space along the direction of steepest descent of the transformed objective functional and apply the inverse of the endomorphism to give a new current point in the original space. The endomorphism is defined in terms of the current point $(x, z) > \theta$ and a preferred point $(x, \boldsymbol{e})$ as follows. Firstly, define z^{-1} by

$$z^{-1} = (z_1^{-1}, \ldots, z_q^{-1})$$

with

$$z_i^{-1}(s) = \frac{e_i(s)}{z_i(s)}, \quad \forall s \in S, \ i = 1, \ldots, q,$$

and z^* by

$$z^* = (z_1^*, \ldots, z_q^*)$$

with

$$z_i^*(s) = \frac{z_i(s)}{e_i(s)}, \quad \forall s \in S, \ i = 1, \ldots, q.$$

Further we note that z^{-1}, z^* are elements of W_+^0. For any w, $v \in W$ we define $\odot : W \times W \to W$ by

$$(w \odot v)_i(s) = w_i(s) v_i(s), \quad \forall_s \in S, \ i = 1, \ldots, q.$$

Clearly $\odot$ is commutative and associative and if w, $v \in W_+^0$ then so is $w \odot v$.

We now define the following mappings in terms of the current point $(x, z) > \theta$:

$$F_z((u, w)) = (u, z^{-1} \odot w),$$

$$F_z^*((u, w)) = (u, z^* \odot w).$$

It is clear that F_z^* and F_z are linear mappings from V to itself. It is easy to see that $F_z(V_+^0) \subseteq V_+^0$ and that $F_z^*(V_+^0) \subseteq V_+^0$. It also follows from the definition of $\odot$ that F_z^* and F_z are mutual inverses.

If we make the assumption that for every $(u, w) \in V$,

$$\langle (c, d), (u, w) \rangle = \langle F_z^*((c, d)), F_z((u, w)) \rangle,$$

then it follows that the solution to the problem given below has the same objective functional value as the solution of LP (the constraints are unchanged since F_z^* and F_z are mutual inverses).

$$\begin{aligned} \text{minimize} \quad & \langle F_z^*((c, d)), F_z((u, w)) \rangle, \\ & T \circ F_z^*(F_z((u, w))) = b, \\ \text{subject to} \quad & F_z^*(F_z((u, w))) \geq \theta, \\ & F_z^*(F_z((u, w))) \in V, \end{aligned}$$

where $T \circ F$ denotes the composition of T and F. Let us write y for $z^{-1} \odot w$, so that $(u, y) = F_z(u, w))$. Since V and V_+ are invariant under F_z and its inverse, it is clear that this problem may be formulated as

$$\begin{aligned} \text{SLP} \quad \text{minimize} \quad & \langle F_z^*((c, d)), (u, y) \rangle, \\ & T \circ F_z^*((u, y)) = b, \\ \text{subject to} \quad & (u, y) \geq \theta, \\ & (u, y) \in V. \end{aligned}$$

The rescaling algorithm for LP can now be described as follows. Given a current point (x, z) feasible for LP and lying in V_+^0, an iteration of the algorithm transforms V by the endomorphism F_z, mapping (x, z) to the preferred point (x, e). A step from (x, e) is then computed so as to give a new point in $F_z(V)$ with a smaller SLP objective functional value than that of (x, e); applying the inverse F_z^* of F_z to V maps this new point to a new current point in V_+^0.

In the finite-dimensional case the rationale underlying the rescaling algorithm is discussed fully in [3] and [11]; the reasoning is similar in the abstract case. The transformation F_z allows us to treat the current point as a preferred point which is chosen to be far away with respect to the supremum norm from the boundary of the positive cone. The direction of the step is given by the orthogonal projection of $-F_z^*((c, d))$ onto the kernel of $T \circ F_z^*$. A strictly positive step in this direction guarantees a strict decrease in the objective functionals of both the original problem and the scaled problem directly proportional to the length of the step. The orthogonal projection is necessary to ensure that the new current point satisfies the constraints of LP.

For the constructions described above to be possible we require a number of assumptions which we now list before proceeding to describe the steps of the algorithm explicitly.

Assumption 1. $V = U \otimes W$ is a pre-Hilbert space with inner product $\langle \cdot, \cdot \rangle$, where U is a vector space and W is a space of functions $S \to \mathbb{R}^q$. $T: V \to Z$ is a linear mapping of V to a vector space Z.

Assumption 2. There exists $\xi \in V$ with $\xi > \theta$.

Assumption 3. For every $(u, w) \in V$,

$$\langle (c, d), (u, w) \rangle = \langle F_z^*((c, d)), F_z((u, w)) \rangle.$$

Assumption 4. For any choice of $z \in W$, the kernel of the linear operator $T \circ F_z^*$ is a Hilbert space with respect to the norm induced by the inner product.

The algorithm

Step 0. Set $k = 0$.

Step 1. Take a feasible point $(x, z)^{(k)} > \theta$ and map it to a preferred point (x, e) using the map $F_{z^{(k)}}$, i.e.

$$(x, e) = F_{z^{(k)}}((x, z)^{(k)}).$$

Step 2. Project the direction of steepest descent for the transformed linear function, $-F_{z^{(k)}}^*((c, d))$, orthogonally onto the kernel of the linear operator $T \circ F_{z^{(k)}}^*$ to give (c_p, z_p).

Step 3. Find a step length $\alpha > 0$ such that $e + \alpha z_p > \mathbf{0}$, and let

$$z' = e + \alpha z_p.$$

Step 4. Invert the transformation to get $(x, z)^{(k+1)}$, i.e.

$$(x, z)^{(k+1)} = F_{z^{(k)}}^*((x + \alpha c_p, z')).$$

Step 5. Check the termination criterion and stop if it is satisfied. Otherwise set $k = k + 1$ and return to Step 1.

It is pertinent at this point to make some remarks regarding the above assumptions. When $U \otimes W = V = \mathbb{R}^n$ with the canonical inner product, the IP norm and the supremum norm are topologically equivalent, and V forms a Hilbert space with the convenient property that V_+^0 (the Cartesian product of U and the set of points in W with strictly positive components) is nonempty. We are therefore guaranteed not only the existence of a projected gradient vector by the Projection Theorem, but also the existence of a preferred element in V_+^0. When W is generalized to be a possibly infinite-dimensional space, the supremum norm and the IP norm are unfortunately no longer topologically equivalent, and the Hilbert spaces (such as $L_2[a, b]$) with which we would like to work have canonical positive cones with empty interiors with respect to the Hilbert-space norm. For this reason it is necessary to make use of the supremum topology, and make Assumption 2, which amounts to choosing W so that V_+^0 is nonempty.

In fact the condition that V be a Hilbert space is stronger than necessary. We require only that Assumption 4 above holds in order to ensure the existence of a projected direction of steepest descent. This assumption will hold in particular for

spaces W and transformations T where $\ker(T \circ F_z^*)$ is finite-dimensional for every z in W. In the next section we shall describe a class of semi-infinite linear programming problems for which this is true, and show how a rescaling algorithm to solve members of this class can be constructed along the lines described above.

2. Semi-infinite linear programming

2.1. Problem formulation

In this section we consider the application of the algorithm to a class of semi-infinite linear pgramming problems. These have the following form.

$$\begin{aligned} &\text{minimize} && c^{\mathrm{T}}x, \\ &\text{subject to} && \sum_{j=1}^{n} a_{1_j}(s)x_j \geqslant b_1(s) \quad \text{if } s \in [l_1, v_1], \\ & && \vdots \qquad\qquad \vdots \qquad\quad \vdots \\ & && \sum_{j=1}^{n} a_{qj}(s)x_j \geqslant b_q(s) \quad \text{if } s \in [l_q, v_q], \end{aligned}$$

where

$$a_{ij}(s),\ b_i(s) \in C^{\infty}[l_i, v_i], \quad i = 1, \ldots, q,\ j = 1, \ldots, n, \text{ and } c,\ x \in \mathbb{R}^n.$$

With a slight abuse of notation we will write this as

$$\begin{aligned} &\text{minimize} && c^{\mathrm{T}}x, \\ &\text{subject to} && A(s)x \geqslant \boldsymbol{b}(s),\ \forall s \in S, \end{aligned}$$

where

$$A(s) = \begin{pmatrix} a_{11}(s) & \cdots & a_{1n}(s) \\ \vdots & \ddots & \vdots \\ a_{q1}(s) & \cdots & a_{qn}(s) \end{pmatrix}, \qquad \boldsymbol{b}(s) = \begin{pmatrix} b_1(s) \\ \vdots \\ b_q(s) \end{pmatrix},$$

and $c,\ x \in \mathbb{R}^n$. (Here $S = [l_i, v_i]$ for the ith equation.) In order to put the above linear program into the form LP, we introduce a slack variable $z_i \in C^{\infty}[l_i, v_i]$ for each constraint in the above system. The problem then becomes

$$\begin{aligned} &\text{minimize} && c^{\mathrm{T}}x, \\ &\text{subject to} && A(s)x - z(s) = \boldsymbol{b}(s),\ \forall s \in S, \\ & && z \geqslant 0, \end{aligned}$$

where $x \in \mathbb{R}^n$, $z \in \prod_{i=1}^{q} C^{\infty}[l_i, v_i]$, which is exactly the form that we require if we set $U = \mathbb{R}^n$ and $W = Z = \prod_{i=1}^{q} C^{\infty}[l_i, v_i]$, and let $T(x, z) = A(\cdot)x - z(\cdot)$. The inner product $\langle \cdot, \cdot \rangle$ is defined on V as

$$\langle (c, \boldsymbol{d}), (x, z) \rangle = c^{\mathrm{T}}x + \sum_{i=1}^{q} \int_{l_i}^{v_i} d_i(s)z_i(s)\, \mathrm{d}s.$$

We assume that $\boldsymbol{e}$ is given by $e_i \in C^\infty[l_i, v_i]$ with $e_i(s) = 1, \forall s \in [l_i, v_i]$, $i = 1, ., \ldots, q$. We note that

$$\langle (c, \mathbf{0}), (x, z) \rangle = c^{\mathrm{T}} x,$$

and see that

$$\langle (c, \boldsymbol{d}), (u, w) \rangle = \langle F_z^*((c, \boldsymbol{d})), F_z((u, w)) \rangle,$$

since from the definition of the inner product given above we have that

$$\begin{aligned} \langle (c, \boldsymbol{d}), (u, w) \rangle &= c^{\mathrm{T}} u + \sum_{i=1}^{q} \int_{l_i}^{v_i} d_i(s) w_i(s)\, \mathrm{d}s \\ &= c^{\mathrm{T}} u + \sum_{i=1}^{q} \int_{l_i}^{v_i} d_i(s) z_i(s) (1/z_i(s)) w_i(s)\, \mathrm{d}s \\ &= \langle F_z^*((c, \boldsymbol{d})), F_z((u, w)) \rangle. \end{aligned}$$

It is also evident that $F_z^*((c, \mathbf{0})) = (c, \mathbf{0})$, so that the direction of steepest descent for the transformed linear function is $-(c, \mathbf{0})$. It remains to show how we can accomplish the orthogonal projections within this framework which is dealt with in the next section.

We note in closing this section that other forms of the semi-infinite linear program problem can be put into the LP framework with equal ease. In particular, if the elements of $\mathbb{R}^n$ are constrained in sign then we can redefine W and $\boldsymbol{e}$ in order to force these elements away from the boundary of the positive cone at each step of the algorithm.

2.2. *A description of the implementation*

Clearly, in Step 1 and Step 4 of the algorithm applied to the semi-infinite case, we are effectively dividing and multiplying a positive function by a particular positive function which has the property that it transforms the current point to the point $(x, \boldsymbol{e})$. In Step 2 of the algorithm we project a vector onto the kernel of $T \circ F_z^{*(k)}$. Observe that this is a finite-dimensional subspace of V so we may accomplish this by carrying out the following steps.

(i) Find a basis $\{\gamma_i : i = 1, \ldots, n\}$ of the kernel of $T \circ F_z^{*(k)}$ where T is defined in Section 2.1. This is accomplished easily since we can construct the elements of the basis as follows:

$$\begin{aligned} T \circ F_z^*((u, w)) &= T(u, z^* \odot w) \\ &= Au - (z^* \odot w) \\ &= Au - (z \odot w) \end{aligned}$$

since $z^* = z$ in this case. Thus, since any (u, w) in the kernel of $T \circ F_z^{*(k)}$ has

$w = Au \odot z^{-1}$, the elements of the basis are

$$\gamma_i = (k_i, r_i) \quad \text{for } i, \ldots, n,$$

where

$$k_i = (\underbrace{0, \ldots, 0, 1, 0, \ldots, 0}_{1 \text{ in } i\text{th position}}),$$

and

$$r_i(s) = (a_{1i}(s)/z_1(s), \ldots, a_{qi}(s)/z_q(s)).$$

(ii) Orthonormalize this finite basis to give

$$\{\beta_i = (g_i, t_i): i = 1, \ldots, n\}.$$

This is accomplished by the modified Gram-Schmidt orthogonalization process, which is described in Goub and Van Loan [6].

It should be noted that the major work per iteration is in the calculation of the inner products (which are required for the Gram-Schmidt procedure). Each inner product requires q integrations, and the number of inner products needed per iteration is $O(n^2)$. Thus the total number of integrations per iteration is $O(n^2 q)$.

It was decided to use Simpson's Rule to evaluate the integrals since it has a strong error bound and a simple implementation. In general the integrations cannot be carried out explicitly since even when the matrix $A(s)$ has polynomial entries, the integrand is a rational function. Most of the function evaluation in this process is repetitive and so the cpu time can be decreased at the expense of using extra storage. Step 3 of the algorithm requires the evaluation of a step length $\alpha > 0$ satisfying

$$e + \alpha z_p > 0.$$

This is evaluated by finding the minimum value of $z_p(s)$ in $[0, 1]$, say $z_p^{(\min)}$, and then setting

$$\alpha = \frac{-\alpha_{(\text{mul})}}{z_p^{(\min)}},$$

where $\alpha_{(\text{mul})}$ is a constant multiplier constrained to lie in the interval $(0, 1)$, and we assume that the minimum value of z_p is negative, otherwise the problem is unbounded. (The choice of $\alpha_{(\text{mul})}$ is not entirely straightforward since the likelihood of the algorithm terminating at a non optimal point increases as we increase $\alpha_{(\text{mul})}$. A further discussion of this point is made in Section 2.3.) For Step 5, the termination criterion chosen was to stop when two successive solutions differed by less than a prespecified tolerance.

We now digress slightly to consider a Phase 1 procedure for the algorithm. At Step 0 we require a feasible solution, which is not always immediately available. We therefore consider the following problem, for some vector k,

$$\begin{aligned} \text{FP} \qquad & \text{minimize} && \lambda, \\ & \text{subject to} && A(s)x + k\lambda \geq \boldsymbol{b}(s), \\ & && \lambda \geq 0. \end{aligned}$$

If we can find a solution to the above problem with $\lambda = 0$ then we have a feasible starting point for the main algorithm. It is easy to see that if we set every component of k to $\max_i(\max_{s\in S}(b_i(s)))$, then a feasible starting point to the algorithm is obtained by setting $\lambda^{(0)}$ to some value greater than 1 and setting x to 0, thus forcing the corresponding slack functions to be greater than $\mathbf{0}$. It is then possible to solve the feasibility problem, FP, by the algorithm described above if we change Step 3 to read

Step 3. (For feasibility) Find the maximum step length $\beta > \mathbf{0}$ such that

$$\boldsymbol{e} + \beta \boldsymbol{z}_p \geq \mathbf{0}.$$

Let $(c_p)_\lambda$ be the component of the projection c_p corresponding to the variable λ. $(c_p)_\lambda < 0$ since FP is bounded below. Then set

$$\boldsymbol{z}' = \boldsymbol{e} + \alpha \boldsymbol{z}_p,$$

where

$$\alpha = \begin{cases} -\lambda/(c_p)_\lambda & \text{if } \lambda + \beta(c_p)_\lambda < 0, \\ \alpha_{\text{mul}} \times \beta & \text{otherwise,} \end{cases}$$

and $\alpha_{\text{mul}} \in (0, 1)$ is a constant multiplier which ensures that the slack function remains strictly positive.

Note that this sets λ to zero at the first instance that it becomes negative and that the variable λ is assumed to lie in U and not in W. This enables the solution of Phase 1 to be easily converted into a starting point for Phase 2, without a redefinition of the positive cone.

2.3. Chebyshev approximation

We consider the problem of approximating a given function $f(s)$ with a finite set of approximating functions $\{a_i(s): i = 1, \ldots, n\}$. This problem may be formulated in the L^∞ norm,

$$\min_{x\in\mathbb{R}^n} \max_{s\in S} \left| f(s) - \sum_{i=1}^{n} x_i a_i(s) \right|,$$

or, with a change of notation,

$$\min_{x\in\mathbb{R}^n} \max_{s\in S} \left| f(s) - \boldsymbol{a}^{\mathrm{T}}(s)x \right|.$$

It is well known that this is equivalent to the following semi-infinite program.

$$\begin{aligned} \text{minimize} \quad & h, \\ \text{subject to} \quad & h + \boldsymbol{a}^{\mathrm{T}}(s)x \geq f(s), \\ & h - \boldsymbol{a}^{\mathrm{T}}(s)x \geq -f(s). \end{aligned}$$

It is clear to see that we can use the algorithm discussed in Section 1 if we set

$$W = Z = C^{\infty}[0, 1] \times C^{\infty}[0, 1],$$

and

$$U = \mathbb{R} \times \mathbb{R}^n.$$

Thus, a general element has the form $(h, x, z_1(s), z_2(s))$ where $z_i(s)$ is the slack function associated with the ith constraint.

The first problem attempted was to approximate s^6 with the functions $a_i(s) = s^{i-1}$, $i = 1, \ldots, n$. The results are given in Table 1.

Table 1

Chebyshev approximation of s^6. Tolerance of solution is 10^{-6}; True solution is 4.88×10^{-4}

$\alpha_{(\mathrm{mul})}$	Phase 1	Phase 2	Solution value	CPU (s)
0.20	1	109	4.99×10^{-4}	26.3
0.40	1	59	4.90×10^{-4}	14.5
0.60	1	45	5.63×10^{-4}	11.1
0.70	1	34	6.63×10^{-4}	8.6
0.80	1	40	5.95×10^{-4}	9.9
0.90	1	26	9.46×10^{-4}	6.7
0.95	1	26	9.25×10^{-4}	6.6
0.99	1	22	2.64×10^{-3}	5.7

2.4. L_1-approximation

We present some less favourable results on the one-sided L_1-approximation problem. A complete description of this problem is given in Glashoff and Gustafson [5]. Let $\phi_1, \ldots, \phi_n$ be an extended Chebyshev system of order two. The problem is then to

find a function q in the span of the ϕ_i, $i = 1, \ldots, n$, which minimizes $\|q - f\|_1$ from above. We can write this as the following optimization problem:

$$\text{minimize} \quad \int_{s \in S} |q(s) - f(s)| w(s)\, ds$$

$$\text{subject to} \quad q(s) = \sum_{i=I}^{n} x_i \phi_i(s), \; q(s) \geq f(s), \quad \text{for all } s \in S$$

where w is a positive weighting function. The constraints enable us to rewrite the integral and after a little algebra it is clear that the problem is equivalent to

$$\text{minimize} \quad c^T x,$$

$$\text{subject to} \quad \sum_{i=I}^{n} x_i \phi_i(s) \geq f(s) \text{ for all } s \in S,$$

where $c_i = \int_{s \in S} \phi_i(s) w(s)\, ds$. For the case $S = [0, 1]$, $\phi_i(s) = s^{i-1}$ and $f(s) = -\sum_{i=0}^{4} s^{2i}$ the problem becomes

$$\text{minimize} \quad \sum_{i=1}^{7} (1/i) x_i,$$

$$\text{subject to} \quad \sum_{i=1}^{7} x_i s^{i-1} \geq -\sum_{i=0}^{4} s^{2i} \quad \text{for } s \in [0, 1].$$

Table 2 gives the results of the implementation applied to this problem which has an optimal solution of -1.78688. Note that the implementation terminates if the number of iterations exceeds 160.

It is evident from these results that the rescaling algorithm performs poorly, especially when the step length $\alpha_{(\text{mul})}$ becomes close to one. We shall attempt an explanation for this poor behaviour in what follows. Before doing this, it is instructive to discuss the relationship between the algorithm described above and the standard rescaling method applied to a discretization of the problem. This is the subject of the next section.

Table 2

L_1-approximation problem. Tolerance of solution is 10^{-6}

$\alpha_{(\text{mul})}$	Phase 1	Phase 2	Solution value	CPU (s)
0.20	0	161	−1.7859	21.4
0.40	0	129	−1.7866	17.3
0.60	0	71	−1.7863	9.4
0.70	0	62	−1.7863	8.7
0.80	0	51	−1.7862	7.2
0.90	0	37	−1.7856	5.3
0.95	0	33	−1.7845	4.5
0.99	0	31	−1.7838	4.1

3. Discretizing the index set

In order to apply the rescaling algorithm to semi-infinite linear programs as described above, a discretization of the set S must be made in order to carry out the computation of the integrals by Simpson's Rule. It is interesting to speculate on whether the performance of this rescaling method differs from that obtained by discretizing S at the outset, and solving the resulting linear program by the standard rescaling method.

This approach has been tried by Kortanek [9] who has experimented with the rescaling method applied to the dual of the discretized problem. (The reader is referred to Gustafson [7] for conditions under which a sequence of solutions to successively finer discrete approximations of a semi-infinite linear program converges to the true solution. If, in the notation of Section 2, we assume without loss of generality that $q = 1$, then these conditions amount to the following.

(i) There exists a feasible z with $z(s) > 0$ for every $s \in S$.

(ii) For $i = 1, \ldots, n$, there exist $s_i \in S$, and $\lambda_i > 0$ such that

$$\{A(s_1), \ldots, A(s_n)\}$$

is a linearly independent set and $c = \sum_{i=1}^{n} \lambda_i A(s_i)$.

It is easily verified that the examples above satisfy these conditions.)

A discretized version of the semi-infinite linear programming problem of section 2.1 is

$$\begin{array}{lll} \text{DLP} & \text{minimize} & c^{\mathrm{T}}x, \\ & \text{subject to} & A(s_h)x \geq \boldsymbol{b}(s_h), \end{array}$$

where s_h, $h = 0, 1, \ldots, N$, runs over some discretization of the set S. For simplicity of notation, we assume that A is a row vector a, so that introducing surplus variables z_h, $h = 0, 1, \ldots, N$, the constraints become

$$\begin{pmatrix} a(s^{(0)}) & & \\ \vdots & & -I \\ a(s^{(N)}) & & \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} b(s^{(0)}) \\ \vdots \\ b(s^{(N)}) \end{pmatrix}, \quad z \geq 0.$$

The rescaling transformation $F_{z^{(k)}}$ of the algorithm becomes premultiplication by a diagonal matrix D with

$$D = \begin{pmatrix} I_n & 0 \\ 0 & D_N^{-1} \end{pmatrix},$$

where $(D_N)_{hh} = z_h^{(k)}$, $h = 0, 1, \ldots, N$. The projection step simplifies to projection $\binom{-c}{0}$ onto the kernel of the matrix $(B \ -D_N)$ where

$$B = \begin{pmatrix} a(s^{(0)}) \\ \vdots \\ a(s^{(N)}) \end{pmatrix}.$$

It is easily verified that the actual projection is given by

$$\begin{pmatrix} c_p \\ z_p \end{pmatrix} = \begin{pmatrix} I_n \\ D_N^{-1} B \end{pmatrix} v,$$

where v solves

$$(I_n + B^{\mathrm{T}} D_N^{-2} B) v = -c.$$

Observe that $\binom{c_p}{z_p}$ lies in the kernel of $(B \ -D_N)$ even though v may be computed inaccurately. Adler et al. [1] have exploited this fact in developing a fast implementation of the rescaling algorithm by applying it to the dual of the linear program in standard form.

The rescaling algorithm was implemented with the projection calculation described above and applied to discretizations of the Chebyshev and L_1-approximation problems. The results were promising and a sample of them have been given in Tables 3, 4 and 5. It should be noted that the results given in Table 4 are accurate to the precision specified by the stopping rule. The corresponding results using Simpson's Rule for the integration stopped before the convergence was completed.

Table 3

Chebyshev approximation of s^6. Discretization interval 0.01.
Tolerance of solution is 10^{-6}; True solution is 4.88×10^{-4}

$\alpha_{(\mathrm{mul})}$	Phase 1	Phase 2	Solution value	CPU (s)
0.20	1	72	4.92×10^{-4}	19.9
0.40	1	34	4.90×10^{-4}	10.2
0.60	1	22	4.89×10^{-4}	6.5
0.70	1	17	4.88×10^{-4}	5.5
0.80	1	15	4.88×10^{-4}	4.9
0.90	1	13	4.88×10^{-4}	4.4
0.95	1	12	4.88×10^{-4}	4.1
0.99	1	11	4.88×10^{-4}	3.8

Table 4

Chebyshev approximation of s^n. Discretization interval 0.01.
Tolerance of solution is 10^{-6}

n	Phase 1	Phase 2	Solution value	CPU (s)
3	1	10	3.1250×10^{-2}	1.7
4	1	11	7.809×10^{-3}	2.4
5	1	11	1.950×10^{-3}	3.0
6	1	12	4.88×10^{-4}	4.1
7	1	12	1.22×10^{-4}	4.9
8	1	11	3.1×10^{-5}	5.5

Table 5

L_1-approximation problem. Discretization interval 0.01.

Tolerance of solution is 10^{-6}

$\alpha_{(\text{mul})}$	Phase 1	Phase 2	Solution value	CPU (s)
0.20	0	88	−1.78689	12.1
0.40	0	45	−1.78689	6.8
0.60	0	30	−1.78689	4.3
0.70	0	28	−1.78689	4.0
0.80	0	20	−1.78687	3.0
0.90	0	22	−1.78686	3.2
0.95	0	20	−1.78678	3.0
0.99	0	16	−1.78663	2.5

Since the computation of the projection $\binom{c_p}{z_p}$ using Simpson's Rule requires the values of $z(s)$ only at the points $z(s_h)$, $h = 0, 1, \ldots, N$, the algorithm described in Section 2.2 can be viewed as working only with the values of z and z_p at these points, as long as the minimum of z_p in **Step 3** is taken over $\{s_h | h = 0, 1, \ldots, N\}$. We shall assume in what follows that this is the case, which allows us to compare the algorithm from Section 2.2 and the standard rescaling algorithm applied to DLP, within the same framework.

The reason for the different performance of the two methods becomes clear when we consider the nature of the discretization of the interval $[0, 1]$ used in Simpson's Rule. Recall that Simpson's rule divides the interval $[0, 1]$ into an even number (N) of subintervals of length δ (0.01 in the above examples) and approximates the integral over $[0, 1]$ of some function f by

$$\frac{\delta}{3}[f(0)+4f(\delta)+2f(2\delta)+4f(3\delta)+2f(4\delta)+\cdots+4f((N-1)\delta)+f(N\delta)].$$

Thus the inner product of functions f and g is approximated by

$$[f(0)f(\delta)\cdots f(N\delta)]F\begin{bmatrix} g(0) \\ g(\delta) \\ \vdots \\ g(N\delta) \end{bmatrix},$$

where F is a diagonal matrix with

$$F_{hh} = \begin{cases} \delta/3 & \text{if } h = 0, N, \\ 2\delta/3 & \text{if } h = 2, 4, \ldots, N-2, \\ 4\delta/3 & \text{if } h = 1, 3, \ldots, N-1. \end{cases}$$

It follows that with exact arithmetic the projection calculation described in Section 2.2 using the modified Gram-Schmidt procedure and Simpson's Rule amounts to

calculating

$$\begin{pmatrix} c_p \\ z_p \end{pmatrix} = \begin{pmatrix} I_n \\ D_N^{-1} B \end{pmatrix} v,$$

where v now is the solution to the normal equations

$$(I_n + B^{\mathrm{T}} D_N^{-1} F D_N^{-1} B) v = -c.$$

It follows that the projected descent directions generated by the two algorithms are different. Observe that for small δ, the presence of F in the normal equations gives a vector c_p which is closer to $-c$ than that which is obtained when F is absent. This discounting of the constraints in the projection calculation gives a sequence of iterates which pass close to the boundary of the feasible region, and terminate prematurely due to roundoff error. However, the poor convergence is not due to the use of Simpson's rule, but rather the poor scaling that is implicitly produced in the inner product. A different inner product would replace F by $\delta^{-1} F$ in the normal equations given above, and then good convergence would result. This hypothesis was confirmed by experimentation.

Despite these difficulties, we may establish a theoretical convergence result which applies to both choices of discretization described above. Our approach closely follows that of Kortanek and Shi [10]. We begin by recalling the steps of the rescaling algorithm. In order to aid exposition the steps have been written in a form which is convenient for algebraic manipulation.

Step 0. Set $k = 0$, choose $(x^{(k)}, z^{(k)})$.

Step 1. Let $D = \mathrm{diag}(z^{(k)}(s_h),\ h = 0, 1, \ldots, N)$ and

$$B = \begin{pmatrix} a(s^{(0)}) \\ \vdots \\ a(s^{(N)}) \end{pmatrix}.$$

Step 2. Compute

$$\begin{bmatrix} c_p^{(k)} \\ z_p^{(k)} \end{bmatrix} = \begin{bmatrix} I \\ D^{-1} B \end{bmatrix} (I + B^{\mathrm{T}} D^{-1} G D^{-1} B)^{-1} (-c).$$

Step 3. Compute

$$\frac{1}{\gamma_k} = -\min\{z_p^{(k)}(s_h) \mid h = 0, 1, \ldots, N\}.$$

Step 4.

$$\begin{bmatrix} x^{(k+1)} \\ z^{(k+1)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ z^{(k)} \end{bmatrix} + \alpha \gamma_k \begin{bmatrix} c_p^{(k)} \\ D z_p^{(k)} \end{bmatrix}.$$

Step 5. Set $k = k+1$, return to Step 1.

The algorithm stops in Step 2 if

$$\begin{bmatrix} c_p^{(k)} \\ z_p^{(k)} \end{bmatrix} = 0,$$

in which case we declare optimality, or in Step 3 if $\gamma_k \leq 0$ which indicates unboundedness. Here G is any diagonal positive definite matrix. Note that if $G = I$, then the algorithm corresponds to a standard discretization, and if $G = F$ as defined in the previous section, then the algorithm is that which uses Simpson's Rule. Observe that G is independent of k, whereas D is not; its dependence on k has been suppressed for notational convenience. It is useful to define at step k the vector

$$y^{(k)} = [DG^{-1}D + BB^{\mathrm{T}}]^{-1}Bc.$$

This vector will be shown to converge to the optimal solution for the dual problem to DLP which can be formulated as follows:

$$\begin{array}{lll} \text{DLP}^* & \text{maximize} & \sum_{h=0}^{N} b(s_h)y_h, \\ & \text{subject to} & B^{\mathrm{T}}y = c, \\ & & y \geq 0. \end{array}$$

It is easy to demonstrate the following lemma.

Lemma 1. *If DLP has an optimal solution and the algorithm does not terminate then*

$$\lim_{k\to\infty} \begin{bmatrix} c_p^{(k)} \\ z_p^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Proof. For each k, we have by virtue of the definition of

$$\begin{bmatrix} c_p^{(k)} \\ z_p^{(k)} \end{bmatrix}$$

that $z_p^{(k)} = D^{-1}Bc_p^{(k)}$ and $-c = c_p^{(k)} + B^{\mathrm{T}}D^{-1}Gz_p^{(k)}$. It now follows from the definition of $x^{(k+1)}$ that

$$c^{\mathrm{T}}x^{(k+1)} = c^{\mathrm{T}}x^{(k)} - \alpha\gamma_k \left[\left\| c_p^{(k)} \right\|_2^2 + (z_p^{(k)})^{\mathrm{T}} G z_p^{(k)} \right].$$

Since the algorithm does not terminate, $\gamma_k > 0$ for every k, it is clear that $c^{\mathrm{T}}x^{(k)}$ is a decreasing sequence bounded below by the optimal value of DLP. Thus $c^{\mathrm{T}}x^{(k)}$ converges, and

$$\lim_{k\to\infty} \gamma_k \left[\left\| c_p^{(k)} \right\|_2^2 + (z_p^{(k)})^{\mathrm{T}} G z_p^{(k)} \right] = 0. \tag{1}$$

Let $\varepsilon = \min\{G_{hh} | h = 0, 1, \ldots, N\}$, so that

$$(z_p^{(k)})^{\mathrm{T}} G z_p^{(k)} \geq \varepsilon \| z_p^{(k)} \|_2^2 > 0$$

which implies that

$$\sqrt{\| c_p^{(k)} \|_2^2 + (z_p^{(k)})^{\mathrm{T}} G z_p^{(k)}} \geq \sqrt{\varepsilon}\, \| z_p^{(k)} \|_2. \tag{2}$$

However,

$$\| z_p^{(k)} \|_2 \geq -\min\{z_p^{(k)}(s_h) | h = 0, 1, \ldots, N\} = \frac{1}{\gamma_k}.$$

Multiplying both sides of (2) by $\sqrt{\| c_p^{(k)} \|^2 + (z_p^{(k)})^{\mathrm{T}} G z_p^{(k)}}$ we find

$$\gamma_k [\| c_p^{(k)} \|_2^2 + (z_p^{(k)})^{\mathrm{T}} G z_p^{(k)}] \geq \sqrt{\varepsilon}\, \sqrt{\| c_p^{(k)} \|_2^2 + (z_p^{(k)})^{\mathrm{T}} G z_p^{(k)}}.$$

It follows immediately from (1) that

$$\lim_{k \to \infty} \begin{bmatrix} c_p^{(k)} \\ z_p^{(k)} \end{bmatrix} = 0. \qquad \square$$

In order to demonstrate the convergence of $\begin{bmatrix} x^{(k)} \\ z^{(k)} \end{bmatrix}$ and $y^{(k)}$ we first prove the key result:

Lemma 2. *If DLP has an optimal solution and the algorithm does not terminate then, for each* $h = 0, 1, \ldots, N$,

$$\liminf D_{hh}^{-1} G_{hh} z_p^{(k)}(s_h) \leq 0.$$

Proof. Choose h, and let $v_k = D_{hh}^{-1} G_{hh} z_p^k(s_h)$. Let $u = \liminf v_k$ and suppose $u > 0$. Then there exists K such that for each $k \geq K$

$$|\lim\{v_k, v_{k+1}, \ldots\} - u| < \frac{u}{2}.$$

Thus, for each $k \geq K$, $v_k > u/2 > 0$. Now since $D^{-1}G$ is a positive definite diagonal matrix, it follows that $z_p^{(k)}(s_h) > 0$, for $k \geq K$. Thus, by the definition of D, for $k \geq K$,

$$z^{(k+1)}(s_h) = z^{(k)}(s_h)(1 + \alpha \gamma_k z_p^{(k)}(s_h)) > z^{(k)}(s_h),$$

implying that $\liminf z^{(k)}(s_h) > 0$. It follows immedaitely from Lemma 1 that v_k converges to zero which contradicts the assumption $u > 0$. $\square$

We now proceed to the main convergence theorem.

Theorem 1. *Suppose that DLP has an optimal solution, and that the rescaling algorithm does not terminate. Let* $H = \{h | \liminf z^{(k)}(s_h) > 0\}$ *where* $\begin{bmatrix} x^{(k)} \\ z^{(k)} \end{bmatrix}$ *is the kth iterate of the algorithm, and let M be the matrix of corresponding columns of the* $(N+1) \times (N+1)$ *identity matrix. If* $(B - M))$ *has rank* $N+1$, *then any limit point* $\begin{bmatrix} \bar{x} \\ \bar{z} \end{bmatrix}$ *of* $\{\begin{bmatrix} x^{(k)} \\ z^{(k)} \end{bmatrix}\}_k$ *solves DLP and* $y^{(k)}$ *converges to a solution of DLP*.*

Proof. Using the definition of $c_p^{(k)}$, $z_p^{(k)}$ and $y^{(k)}$ it is easy to derive the relationship

$$\begin{bmatrix} B^T \\ -I \end{bmatrix} y^{(k)} - \begin{bmatrix} c \\ 0 \end{bmatrix} = \begin{bmatrix} c_p^{(k)} \\ D^{-1} G z_p^{(k)} \end{bmatrix}. \tag{3}$$

Furthermore, for $h \in H$, $\liminf D_{hh}^{(k)} > 0$, whence Lemmas 1 and 2 imply that

$$\lim_{k \to \infty} D_{hh}^{-1} G_{hh} z_p^{(k)}(s_h) = 0.$$

Thus, since $\lim_{k\to\infty} c_p^{(k)} = 0$, it follows from (3) that

$$\lim_{k \to \infty} \begin{bmatrix} B^T \\ -M^T \end{bmatrix} y^{(k)} = \begin{bmatrix} c \\ 0 \end{bmatrix}.$$

We can now invoke the standard argument of Kortanek and Shi [10] to show that $y^{(k)}$ converges to some vector $\bar{y}$. Formally, $y^{(k)}$ is bounded, since if not, then

$$\left\{ \frac{y^{(k)}}{\|y^{(k)}\|_2} \right\}_k$$

is a bounded sequence having a limit point u satisfying

$$\begin{bmatrix} B^T \\ -M^T \end{bmatrix} u = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

which contradicts the full rank assumption. Thus $y^{(k)}$ has limit points. Moreover, if y_1 and y_2 are two such limit points then

$$\begin{bmatrix} B^T \\ -M^T \end{bmatrix} y_1 = \begin{bmatrix} c \\ 0 \end{bmatrix} = \begin{bmatrix} B^T \\ -M^T \end{bmatrix} y_2,$$

which implies $y_1 = y_2$, by the full rank assumption. Thus $y^{(k)}$ converges to $\bar{y}$.

It is now sufficient to examine (3) to see that $D^{-1} G z_p^{(k)}$ converges to $-\bar{y}$. By Lemma 2, $\bar{y} \geq 0$, and so $\bar{y}$ is feasible for DLP*.

Returning to the primal problem, we consider the sequence $\{[\begin{smallmatrix} x^{(k)} \\ z^{(k)} \end{smallmatrix}]\}_k$ and let $\{[\begin{smallmatrix} x^{(k(m))} \\ z^{(k(m))} \end{smallmatrix}]\}_m$ be a subsequence converging to the limit point $[\begin{smallmatrix} \bar{x} \\ \bar{z} \end{smallmatrix}]$. If, for any h, $\bar{z}(s_h) > 0$, then, by Lemma 1,

$$\lim_{m \to \infty} D_{hh}^{-1} G_{hh} z_p^{(k(m))}(s_h) = 0,$$

which gives $\lim_{m\to\infty} y_h^{(k(m))} = 0$, implying that $\bar{y}_h = 0$. By standard complementary slackness arguments, namely

$$\begin{aligned} c^T \bar{x} = \bar{y}^T B \bar{x} &= \sum_{h=0}^{N} \bar{y}_h (b(s_h) + \bar{z}(s_h)) \\ &= \sum_{h=0}^{N} \bar{y}_h b(s_h), \end{aligned}$$

it follows that $[\begin{smallmatrix} \bar{x} \\ \bar{z} \end{smallmatrix}]$ and $\bar{y}$ solve DLP and its dual respectively. □

The above theorem shows that under suitable conditions the rescaling algorithm using Simpson's Rule to carry out the integrations will converge. In practice, however, choosing $G = F$ gives poorer performance than choosing $G = I$. For coarser discretizations (for example, $N = 10$) when both algorithms converge we have observed that choosing $G = F$ takes approximately three times as many iterations as choosing $G = I$. As remarked above, when the discretization is finer then rounding error gives termination at a non optimal point.

Kortanek [9] has proved that the (non optimal) iterates $x^{(k)}$ of the rescaling algorithm when applied to a problem with optimal solution $\bar{x}$ satisfy

$$\frac{c^{\mathrm{T}}x^{(k+1)} - c^{\mathrm{T}}\bar{x}}{c^{\mathrm{T}}x^{(k)} - c^{\mathrm{T}}\bar{x}} \leq 1 - \frac{\alpha\gamma_k \|c_p^{(k)}\|}{2\sqrt{n}}$$

for sufficiently large k. Here n is the number of variables in the problem and $c_p^{(k)}$ is the current projection of Dc. The proof relies on choosing k large enough to guarantee that the current estimate of the dual solution is close to the optimum. A similar proof for the algorithm described in Section 2.2 gives

$$\frac{c^{\mathrm{T}}x^{(k+1)} - c^{\mathrm{T}}\bar{x}}{c^{\mathrm{T}}x^{(k)} - c^{\mathrm{T}}\bar{x}} \leq 1 - \frac{\alpha\gamma_k \|z_p^{(k)}\| \varepsilon}{2\sqrt{N+1}\mu} \tag{4}$$

where $\varepsilon = \min\{G_{hh} \mid h = 0, 1, \ldots, N\}$ and $\mu = \max\{G_{hh} \mid h = 0, 1, \ldots, N\}$.

It is tempting to suppose that the ratio ε/μ, which equals 0.25 when $G = F$, and 1.0 when $G = I$ is responsible for the slower convergence of the rescaling method using Simpson's Rule. In fact, this is not the case, and experiments verified that scaling the matrix F by $1/\delta$ improved the convergence to a level similar to that obtained when $G = I$.

Further experimentation confirmed that the decrease expected in $c^{\mathrm{T}}x$ by virtue of (4) was often not obtained in the course of the algorithm. Examination of the estimates for the dual variables showed that these were quite different from their optimal values which indicates that the convergence rate obtained in (4) is accurate only in the immediate vicinity of the optimal solution.

Acknowledgement

The authors are grateful to a referee for suggestions concerning the discretization of the problem.

References

[1] I. Adler, N.K. Karmarkar, M.G.C. Resende and G. Veiga. "An implementation of Karmarkar's algorithm for linear programming," Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720.

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

[3] T.M. Cavalier and A.L. Soyster, "Some Computational Experience and a Modification of the Karmarkar Algorithm," ISME Working Paper 85-105, Pennsylvania State University, Pennsylvania (1985).

[4] I.I. Dikin, "Iterative solution of problems of linear and quadratic programming," *Soviet Mathematics Doklady* 8(3) (1967) 674-675.

[5] K. Glashoff and S.-Å. Gustafson, *Linear Optimization and Approximation* (Springer-Verlag, New York, 1983).

[6] G.H. Golub and C.F. Van Loan, *Matrix Computations* (The John Hopkins University Press, Baltimore, Maryland, 1983).

[7] S.-Å. Gustafson. "On numerical analysis in semi-infinite programming," in: R. Hettich, ed., *Semi-Infinite Programming* (Springer-Verlag, Berlin, 1979).

[8] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica* 4 (1984) 373-395.

[9] K.O. Kortanek. "Vector-Supercomputer Experiments with the Linear Programming Scaling Algorithm," Working Paper 87-2, College of Business Adminstration, University of Iowa, Iowa City (1987).

[10] K.O. Kortanek and M. Shi. "Convergence results and numerical experiments on a linear programming hybrid algorithm," *European Journal of Operational Research* 32 (1987) 47-61.

[11] R.J. Vanderbei, M.S. Meketon and B.A. Freedman, "A modification of Karmarkar's algorithm," *Algorithmica* 1 (1986) 395-407.