INFORMATICA, 1997, Vol. 8, No. 4, 559–582

AN EXTERIOR-POINT POLYTOPE SLIDING AND DEFORMATION ALGORITHM FOR LINEAR PROGRAMMING PROBLEMS

Hanif D. SHERALI, Gyunghyun CHOI,

Department of Industrial and Systems Engineering
Virginia Polytechnic Institute and State University, Blacksbur, Virginia 24061–0118
E-mail: hanifs@vt.edu, ghchoi@unitel.co.kr

Suvrajeet SEN

Department of Systems and Industrial Engineering, University of Arizona
Tucson, Arizona 85721
E-mail: sen@sie.arizona.edu

Abstract. In this research, we develop an algorithm for linear programming problems based on a new interpretation of Karmarkar's representation for this problem. Accordingly, we examine a suitable polytope for which the origin is an exterior point, and in order to determine an optimal solution, we need to ascertain the minimum extent by which this polytope needs to be slid along a one-dimensional axis so that the origin belongs to it. To accomplish this, we employ strongly separating hyperplanes between the origin and the polytope using a closest point routine. The algorithm is further enhanced by the generation of dual solutions which enable us to deform the polytope so that it is favorably positioned with respect to the origin and the axis of sliding motion. The overall scheme is easy to implement, requires a minimal amount of storage, and produces quick good quality lower bounds for the problem in its infinite convergence process. A switchover to the simplex method or an interior point method is also possible, using the current available solution as an advanced start. Preliminary computational results are provided along with implementation guidelines.

Key words: linear programming, exterior point method, Karmarkar's algorithm.

1. Introduction.

Consider a linear programming problem of the form

$$\text{Minimize } \{cy\colon\ Ay = b,\ y \geqslant 0\} \tag{1.1}$$

where A is an $m \times n$ matrix. The algorithm proposed below constructs a polytope based on (1.1), using both the objective function and constraints. By using a suitable sliding, rotation, and deformation of this polytope, the algorithm determines an optimal solution to (1.1). To define this polytope, let us first transform the given linear programming problem to an equivalent form having the representation used by Karmarkar (1984).

Toward this end, following Tomlin (1985), let us impose an upper bound Q on the sum of the variables in (1.1), where this constraint either artificially bounds the polyhedron in (1.1), or is implied by it. Denoting the slack variable in this constraint as y_{n+1}, and constructing a dummy variable y_0 defined to be unity in order to homogenize the constraints, we can write this linear program as

$$\begin{aligned}
\text{Minimize}\quad & cy \\
\text{subject to}\quad & Ay - by_0 = 0, \\
& \textstyle\sum_{j=1}^{n} y_j + y_{n+1} - Qy_0 = 0, \\
& \textstyle\sum_{j=1}^{n} y_j + y_{n+1} + y_0 = (Q+1), \\
& y_0, \ldots, y_{n+1} \geqslant 0,
\end{aligned}$$

where the last two equality constraints imply that $y_0 \equiv 1$. Finally, using the variable transformation $\lambda_j = y_j/(Q+1)$ for $j = 0, 1, 2, \ldots, n+1$, we may equivalently write this problem as follows, where $A = [a_1, \ldots, a_n]$.

$$\begin{aligned}
\text{Minimize}\quad & \sum_{j=1}^{n} c_j \lambda_j \\
\text{subject to}\quad & \sum_{j=1}^{n} a_j \lambda_j - b\lambda_0 = 0, \\
& \sum_{j=1}^{n} \lambda_j + \lambda_{n+1} - Q\lambda_0 = 0, \qquad (1.2a) \\
& \sum_{j=1}^{n} \lambda_j + \lambda_{n+1} + \lambda_0 = 1, \\
& \lambda_0, \ldots, \lambda_{n+1} \geqslant 0.
\end{aligned}$$

Define $\lambda = \lambda_0, \ldots, \lambda_{n+1})^t$, $C = (c_0, c_1, \ldots, c_n, c_{n+1})$ with $c_0 = c_{n+1} \equiv 0$, let $G = [g_0, \ldots, g_{n+1}]$, where

$$g_0 = \begin{bmatrix} -b \\ -Q \end{bmatrix}, \; g_j = \begin{bmatrix} a_j \\ 1 \end{bmatrix} \quad \text{for} \quad j = 1, \ldots, n, \quad \text{and} \quad g_{n+1} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix},$$

and denote by e a (row) vector of $n + 2$ ones. Then, we may write the linear program (**LP**) in (1.2a), which is the problem we intend to solve, as follows

$$\textbf{LP}\text{: Minimize } \{C\lambda\colon\ G\lambda = 0,\ e\lambda = 1,\ \lambda \geqslant 0\}. \tag{1.2b}$$

Note that our choice of the notation "λ" is to emphasize the convex combination interpretation afforded by (1.2b). To further develop this interpretation, let z^k be some known lower bound on the optimal objective value for **LP** at iteration k of some algorithmic process, and define the vectors

$$\nu_j^k = \begin{bmatrix} c_j - z^k \\ g_j \end{bmatrix} \quad \text{for} \quad j = 0, \ldots, n+1. \tag{1.3}$$

Consider the polytope

$$X(z^k) = conv\{\nu_j^k\colon\ j = 0, \ldots, n+1\}, \tag{1.4}$$

where $conv\{\cdot\}$ denotes the convex hull operation (see Fig. 1). Note that if $0 \in X(z^k)$, then 0 can be represented by some convex combination of ν_j^k, $j = 0, \ldots, n+1$, that is, there exists a vector $\lambda = \lambda^k$ which satisfies the system

$$\sum_{j=0}^{n+1} \nu_j^k \lambda_j = 0, \quad e\lambda = 1, \quad \lambda \geqslant 0. \tag{1.5a}$$

Therefore, this $\lambda = \lambda^k$ satisfies the system

$$C\lambda = z^k, \quad G\lambda = 0, \quad e\lambda = 1, \quad \lambda \geqslant 0. \tag{1.5b}$$

Since z^k is a lower bound on the optimal objective value for **LP**, we therefore have that λ is an optimal solution to **LP** of objective value z^k. On the other hand, suppose that the origin lies exterior to $X(z^k)$. The problem then becomes one of determining the minimum amount by which z^k needs to be increased for the origin to belong to the resulting $X(z^k)$, if at all possible. Noting from (1.3) and (1.4) that increasing z^k simply results in reducing the first component of each vector ν_j^k $\forall j$ by that same amount, this process is equivalent to sliding the polytope $X(z^k)$ for any z^k along the negative first axis direction, until the origin belongs to this polytope. Fig. 1. illustrates this process conceptually. For convenience, we will hereafter refer to the first axis direction as the *z*-**axis**. Note that the shape of the polytope $X(z^k)$ is invariant with respect to z^k; only its position with respect to the *z*-axis changes as z^k is increased. Furthermore, observe that if

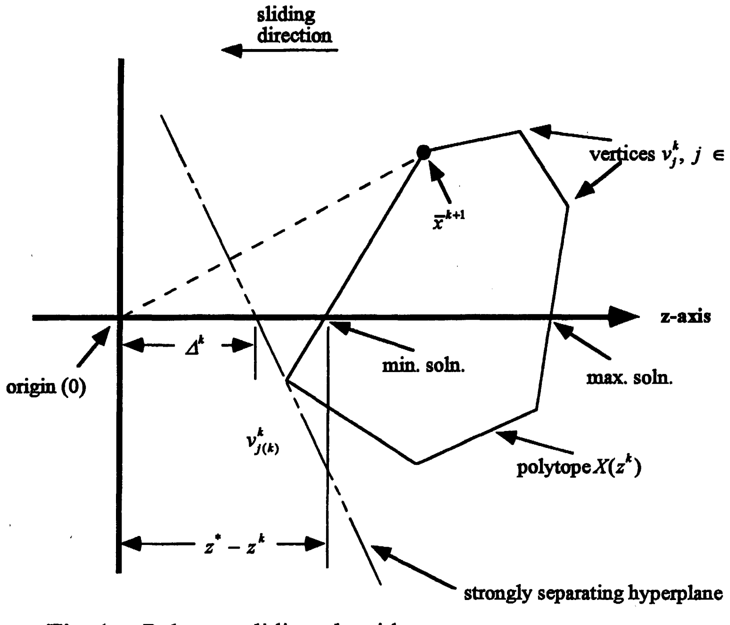

Fig. 1. Polytope sliding algorithm.

the z-axis does not pierce the polytope, then $z^k \rightarrow \infty$ in this sliding process, and the problem is evidently infeasible. Otherwise, if z^* denotes the optimal objective value for **LP**, then by sliding $X(z^k)$ along the negative z-axis by the amount $(z^* - z^k)$, we will obtain $X(z^*)$ with the origin belonging to this latter polytope. The convex combination weights λ^* that represent the origin in $X(z^*)$ in terms of its vertices would then yield an optimal solution to **LP**. Observe also that if we continued this sliding process further, we would trace solutions that take on all the objective values possible for **LP**, up to the maximizing solution value, beyond which the origin would again become exterior to the polytope.

The algorithm proposed here exploits this interpretation by iteratively improving the lower bound z^k via a closest point routine. This routine either verifies that $0 \in X(z^k)$, whence $z^* = z^k$, or else generates a strongly separating hyperplane between the origin and the polytope. This hyperplane permits us to increase z^k, and hence to slide the polytope by an additional amount Δ^k (see Figure 1), and the process continues until ε-optimality is achieved (finitely), for any chosen tolerance $\varepsilon > 0$. Furthermore, we also exhibit how a sequence of dual feasible solutions, approaching an optimal solution, may be generated and used algorithmically to deform the polytope at each iteration so that all its vertices have nonnegative z-axis components. Additional enhancements are also proposed to curtail computational effort, and to increase the amount of sliding at each iteration by suitably rotating the separating hyperplanes.

The proposed algorithm may be used in one of two capacities, besides as a direct algorithm for solving linear programming problems. First, because it is easy to implement, makes rapid initial progress, and has a minimal storage requirement, it offers a useful alternative for obtaining lower bounds based on linear programming relaxations for discrete optimization problems. In this context, the primal and dual solutions available can moreover be used to derive heuristic solutions for discrete problems, and to generate strong Benders' inequalities if necessary (see Parker and Rardin, 1987). Second, this method can be used to provide an advanced-start solution for the simplex method or for interior point methods. As an algorithm for linear programming, this switchover is also recommended in order to accelerate the convergence of the method.

The remainder of this paper is organized as follows. Section 2 presents a generic statement of the basic algorithm, and establishes its convergence. A technique for tightening the lower bound via a suitable rotation of the constructed separating hyperplanes is provided in Section 3, and Section 4 discusses the generation of dual solutions along with an accompanying deformation of the polytope. The overall algorithm with these modifications is stated in Section 5, and its convergence is established. Finally, Section 6 presents some computational results along with implementation guidelines.

2. Generic rudimentary algorithm. The basic polytope sliding procedure introduced in Section 1 hinges on the generation of a strongly separating hyperplane via a closest point routine (CPR) that can determine a closest point to the origin in the convex hull of a finite number of points relative to a given point. Specifically, we require a closest point routine (CPR) which is capable of accomplishing the following. (Sherali and Choi, 1993, present and test one such method, and compare it with some alternatives.)

Closest Point Routine (CPR)

Input: We are given ν_j^k, $j = 0, \ldots, n+1$ for some lower bound z^k on LP, a point $x^k \in X(z^k) \equiv conv\{\nu_j^k,\ j = 0, \ldots, n+1\}$ along with the convex combination weights λ^k such that $x^k \equiv \sum_{j=0}^{n+1} \nu_j^k \lambda_j^k$, where $e\lambda^k = 1$ and $\lambda^k \geqslant 0$, and we are also given a constant $0 < \mu \leqslant 1$, and some tolerance $\varepsilon > 0$.

Output: Case (1). If $0 \in X(z^k)$, then starting with $p^1 = x^k$, the method should generate a sequence $\{p^t\}$ in $X(z^k)$ which either terminates finitely with an element $p^T = 0$, or else $\{p^t\} \to 0$. Hence, in a finite number of iterations, we should obtain $\bar{x}^{k+1} \equiv p^T$ for some $T \geqslant 1$ such that $|\bar{x}^{k+1}|^2 \leqslant \varepsilon$.

Case (2). If $0 \notin X(z^k)$, then starting with $p^1 = x^k$, an element $\bar{x}^{k+1} \equiv p^T \in X(z^k)$ should be produced for some finite $T \geqslant 1$, such that

$$(\bar{x}^{k+1})^t \nu^k_{j(k)} \equiv \text{minimum}\{(\bar{x}^{k+1})^t \nu^k_j : j = 0, \dots, n+1\} \geqslant \mu |\bar{x}^{k+1}|^2. \quad (2.1)$$

Hence, we would then have

$$(\bar{x}^{k+1})^t (x - \nu^k_{j(k)}) \geqslant 0 \quad \text{for all} \quad x \in X(z^k), \quad (2.2)$$

while the origin would lie strictly in the opposite half-space, i.e.,

$$(\bar{x}^{k+1})^t (-\nu^k_{j(k)}) \leqslant -\mu \|\bar{x}^{k+1}\|^2 < 0.$$

Therefore, the hyperplane defining (2.2) would strongly separate the origin from $X(z^k)$. In particular, note that if it happens that $\bar{x}^{k+1}$ is the closest point in $X(z^k)$ to the origin, then $(\bar{x}^{k+1})^t(x - \bar{x}^{k+1}) \geqslant 0$ for all $x \in X(z^k)$, and so from (2.1), $(\bar{x}^{k+1})^t \nu^k_{j(k)} = \|\bar{x}^{k+1}\|^2$. This motivates our choice of $0 < \mu \leqslant 1$ in (2.1), which controls the degree of accuracy imposed on determining the closest point in $X(z^k)$ to the origin.

Moreover, in either Case (1) or Case (2), the CPR should also yield a corresponding set of convex combination weights λ^{k+1} such that $\bar{x}^{k+1} = \sum_{j=0}^{n+1} \nu^k_j \lambda^{k+1}_j$, where $e\lambda^{k+1} = 1$ and $\lambda^{k+1} \geqslant 0$. Then, using any such procedure as a subroutine, we derive the following algorithm.

Polytope Sliding Algorithm (PSA)

Initialization: Put $k = 1$, select $z^k = c_{\min} \equiv \min\{c_j \colon j = 0, \dots, n+1\}$ as a starting lower bound for LP, and define ν^k_j, $j = 0, \dots, n+1$, and $X(z^k)$ as in (1.3) and (1.4), respectively. Define

$$\lambda^k_0 = 1/(Q+1), \; \lambda^k_j = 0 \quad \text{for} \quad j = 1, \dots, m, \quad \text{and} \quad \lambda^k_{n+1} = Q/(Q+1),$$

as a starting set of convex combination weights, and accordingly, let

$$x^k = \sum_{j=0}^{n+1} \nu^k_j \lambda^k_j \equiv \begin{bmatrix} -c_{\min} \\ -b/(Q+1) \\ 0 \end{bmatrix}$$

be the corresponding point in $X(z^k)$. Select a termination tolerance $\varepsilon > 0$, a constant $0 < \mu \leqslant 1$, and proceed to Step 1.

Step 1 (CPR): Starting with $x^k \in X(z^k)$, invoke a closest point routine CPR to produce a solution

$$\bar{x}^{k+1} \equiv \sum_{j=0}^{n+1} \nu_j^k \lambda_j^{k+1} \in X(z^k).$$

If $\|\bar{x}^{k+1}\|^2 \leqslant \varepsilon$ (Case (1) or Case (2) of CPR), go to Step 3(a). Otherwise, $\|\bar{x}^{k+1}\|^2 > \varepsilon$, and as in Case (2), we obtain $\bar{x}^{k+1} \in X(z^k)$ satisfying (2.1) and (2.2).

Step 2 (Sliding operation): If the component $\bar{x}_1^{k+1} \leqslant 0$, go to Step 3(b). Otherwise, compute the point $(\Delta^k, 0, \dots, 0)^t$ lying on the intersection of the z-axis and the hyperplane defined in (2.2). Hence, we obtain

$$\Delta^k = \frac{(\bar{x}^{k+1})^t \nu_{j(k)}^k}{\bar{x}_1^{k+1}} > 0. \tag{2.3}$$

Consequently, the polytope (and the separating hyperplane) can be slid along the negative z-axis by the amount Δ^k, and the hyperplane will remain a separating hyperplane between the origin and the polytope. In other words, we can raise the lower bound to $z^{k+1} = z^k + \Delta^k$ and accordingly, put

$$\nu_j^{k+1} = \nu_j^k + \begin{bmatrix} -\Delta^k \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{for} \quad j = 0, \dots, n+1,$$

and let

$$x^{k+1} = \bar{x}^{k+1} + \begin{bmatrix} -\Delta^k \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

be the new iterate. Note that $x^{k+1} = \sum_{j=0}^{n+1} \nu_j^{k+1} \lambda_j^{k+1} \in X(z^{k+1})$. Increment k by one and return to Step 1.

Step 3(a) (ε-optimality): Consider the convex combination vector corresponding to $\bar{x}^{k+1}$. Now we have

$$|C\lambda - z^k| = |\bar{x}_1^{k+1}| \leqslant \|\bar{x}^{k+1}\|_\infty$$

and

$$|(G\lambda)_r| = |\bar{x}^{k+1}_{r+1}| \leqslant \|\bar{x}^{k+1}\|_\infty \quad \text{for} \quad r = 1, \ldots, m+1.$$

Since $\|\bar{x}^{k+1}\|_\infty \leqslant \|\bar{x}^{k+1}\| \leqslant \sqrt{\varepsilon}$, we have

$$z^k - \sqrt{\varepsilon} \leqslant C\lambda \leqslant z^k + \sqrt{\varepsilon}, \quad \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix} \sqrt{\varepsilon} \leqslant G\lambda \leqslant \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \sqrt{\varepsilon}, \; e\lambda = 1, \; \lambda \geqslant 0.$$

That is, $\lambda \equiv \lambda^{k+1}$ is an ε-optimal solution to Problem LP. Note that with integer data, and with $\varepsilon = 2^{-2L}$, where L is the number of binary bits required to store the data, λ^{k+1} can be rounded to a feasible solution using the method of Gacs and Lovasz (1981). Alternatively, we can switchover to the simplex method for example, using suitable artificial variables and the purification technique described in Bazaraa *et al.* (1990), and hence polish λ^{k+1} to an exact optimum.

Step 3(b) (Infeasibility): Since $\bar{x}^{k+1}_1 \leqslant 0$, $\|\bar{x}^{k+1}\|^2 > \varepsilon$, and (2.1) and (2.2) hold, we now have for any point $\Phi = (\phi, 0, \ldots, 0)^t$ on the z-axis, $\phi \geqslant 0$, that

$$\begin{aligned}(\bar{x}^{k+1})^t(\Phi - \nu^k_{j(k)}) &= \phi\bar{x}^{k+1}_1 - (\bar{x}^{k+1})^t\nu^k_{j(k)} \leqslant -(\bar{x}^{k+1})^t\nu^k_{j(k)} \\ &\leqslant -\mu\|\bar{x}^{k+1}\|^2 < -\mu\varepsilon. \end{aligned} \tag{2.4}$$

Hence, the nonnegative z-axis is strongly separated from $X(z^k)$ by the hyperplane (2.2), and so we can terminate with the indication that LP is infeasible.

The above algorithm therefore produces an increasing sequence of valid lower bounds for LP, and when it terminates at Step 3, it does so with either an ε-optimal solution or with a valid indication of infeasibility. The following result establishes finite convergence of the algorithm, and addresses the issue of running the algorithm with $\varepsilon = 0$.

Theorem 1 (Convergence theorem).

(a) *Given* $\varepsilon > 0$ *and* $0 < \mu \leqslant 1$, *the Polytope Sliding Algorithm (PSA) terminates finitely in less than* $1 + (c_{\min} - c_{\min})^2/\mu\varepsilon$ *iterations, where* $c_{\max}$ *and* $c_{\min}$ *are, respectively, the largest and the smallest values of* c_j, $j = 0, \ldots, n+1$.

(b) *Suppose the PSA is run with* $\varepsilon = 0$, *for some* $0 < \mu \leqslant 1$. *Then either* (i) *CPR will generate a sequence* $\{p^t\}$ *in* $X(z^k)$ *for some* k *such that* $p^T = 0$ *for some finite* $t = T$, *or such that* $\{p^t\} \to 0$, *or* (ii) *the Algorithm PSA will terminate finitely at Step 3(b) with an infeasibility indication, or* (iii) *an infinite sequence* $\{x^k\}$ *will be generated such that* $\{x^k\} \to 0$. *If* $p^T = 0$ *occurs finitely in Case* (i), *then the convex combination weights corresponding to* p^T *solve* **LP**. *Otherwise, in the first and third cases, the limit of any convergent subsequence* *of the convex combination weights generated, corresponding to* $\{p^t\}$ *in Case* (i) *and to* $\{x^k\}$ *in Case* (iii), *solves* **LP**.

Proof. (a) Note that while termination has not occurred, we have $\|\bar{x}^{k+1}\| > \varepsilon$ and $\bar{x}_1^{k+1} > 0$. In particular, and since $\{z^k\}$ is an increasing sequence with $z^1 = c_{\min}$, we have

$$0 < \bar{x}_1^{k+1} = \sum_{j=0}^{n+1} \lambda_j^{k+1}(c_j - z^k) = \Big(\sum_{j=0}^{n+1} \lambda_j^{k+1} c_j\Big) - z^k \leqslant c_{\max} - c_{\min}. \quad (2.5a)$$

This implies that

$$c_{\min} \leqslant z^k < c_{\max}, \quad \text{and that} \quad 0 < \bar{x}_1^{k+1} \leqslant c_{\max} - c_{\min}. \quad (2.5b)$$

However, from Step 2 of PSA, we have using (2.1), (2.3), and (2.5b), that

$$\begin{aligned} z^{k+1} &= z^k + \Delta^k = z^k + \frac{(\bar{x}^{k+1})^t \nu_{j(k)}^k}{\bar{x}_1^{k+1}} \\ &\geqslant z^k + \frac{\mu\|\bar{x}^{k+1}\|^2}{\bar{x}_1^{k+1}} \geqslant z^k + \frac{\mu\varepsilon}{(c_{\max} - c_{\min})}. \end{aligned} \quad (2.6)$$

Consequently, from (2.5b) and (2.6), we deduce directly that

$$c_{\max} > z^k \geqslant z^1 + \frac{(k-1)\mu\varepsilon}{(c_{\max} - c_{\min})} = c_{\min} + \frac{(k-1)\mu\varepsilon}{(c_{\max} - c_{\min})}.$$

If $k \geqslant 1 + (c_{\max} - c_{\min})^2/\mu\varepsilon$, this gives a contradiction, hence establishing the required bound on the number of iterations.

(b) The assertions concerning cases (i) and (ii) are readily evident. On the other hand, if cases (i) and (ii) do not occur, then an infinite sequence $\{x^k\}$ is generated such that $\bar{x}_1^{k+1} > 0$ for all k. Hence, we have (2.5) holding, and as in (2.6), we obtain

$$z^{k+1} \geqslant z^k + \frac{\mu\|\bar{x}^{k+1}\|^2}{(c_{\max} - c_{\min})} > z^k.$$

Since $\{z^k\}$ is bounded above by $c_{\max}$ from (2.5b) and is strictly increasing, this means that $\{z^k\}$ converges to some $\bar{z}$, and moreover, we must have $\{\|\bar{x}^{k+1}\|\} \to 0$ and also $\{\Delta^k\} \to 0$. But $\|x^{k+1}\| \leqslant \|\bar{x}^{k+1}\| + \Delta^k$. Hence, $\{\|x^k\|\} \to 0$, and so $\{x^k\} \to 0$. Since an optimal solution to **LP** corresponds to a set of convex combination weights that represent the origin in terms of the vertices of $X(\bar{z})$

for some lower bound $\bar{z}$, whence $\bar{z}$ must be the optimal value, this completes the proof.

Several remarks which offer insights into the operation and interpretation of the above algorithm are given below.

REMARK 1. The algorithm can be initialized using any known feasible solution to (1.1), if so desired. The starting solution suggested at the initialization step of the procedure is adequate in practice because of the rapid initial improvements made by the algorithm. Moreover, the algorithm is oblivious to any degeneracy related issues or to the rank deficiency of A in (1.1).

REMARK 2. There is an interesting interpretation for Algorithm PSA in light of Newton's algorithm (see Bazaraa *et al.*, 1993). Noting (1.3), let us define

$$f(\lambda, z) = \left\| \sum_{j=0}^{n+1} \begin{bmatrix} c_j - z \\ g_j \end{bmatrix} \lambda_j \right\|^2 = (C\lambda - z)^2 + \|G\lambda\|^2 \tag{2.7a}$$

and consider the function

$$\psi(z) = \operatorname*{Minimum}_{\lambda}\{f(\lambda, z) \colon e\lambda = 1,\ \lambda \geqslant 0\}. \tag{2.7b}$$

Note that $\psi(z)$ is the distance of the closest point in $X(z)$ to the origin, and we wish to find the smallest z for which $\psi(z) = 0$. Now, given z^k, suppose that CPR finds $\bar{x}^{k+1}$ as the actual closest point in $X(z^k)$ to the origin, where $\bar{x}_1^{k+1} > 0$. Hence, because of the uniqueness of the closest point, for any optimal solution λ^{k+1} to (2.7b), we have

$$\begin{bmatrix} C\lambda^{k+1} - z^k \\ G\lambda^{k+1} \end{bmatrix} \equiv \bar{x}^{k+1}$$

uniquely. Moreover, if $\psi(\cdot)$ is differentiable at $z = z^k$, then its derivative can be obtained as

$$\psi'(z^k) = -2(C\lambda^{k+1} - z^k) = -2\bar{x}_1^{k+1}.$$

Therefore, the first-order approximation to $\psi(\cdot)$ at $z = z^k$ is given by

$$\psi(z^k) + (z - z^k)(-2\bar{x}_1^{k+1}),$$

and this is zero when

$$z = z^k + \frac{\psi(z^k)}{2\bar{x}_1^{k+1}} = z^k + \frac{\|\bar{x}^{k+1}\|^2}{2\bar{x}_1^{k+1}} = z^k + \frac{(\bar{x}^{k+1})^t \nu_{j(k)}^k}{2\bar{x}_1^{k+1}}, \tag{2.8}$$

where we have used (2.1) at the last step. However, note that this Newton step is half the actual stepsize we take in (2.3), and hence, this method accounts for the curvature of $\psi(\cdot)$ as well, and goes beyond a Newton based scheme for finding a (smallest) root for $\psi(\cdot)$.

3. Rotation of the separating hyperplane. Recall that in the Polytope Sliding Algorithm PSA, the closest point routine either generates an element in $X(z^k)$ which is sufficiently close to the origin, or else, unless infeasibility is recognized at this step, it constructs a separating hyperplane (2.2) which permits the revision of Δ^k given by (2.3) in the lower bound. We can possibly further enhance the improvement in the lower bound beyond that given by (2.3) by suitably rotating the separating hyperplane before conducting the sliding operation, while maintaining it as a strongly separating hyperplane.

Specifically, noting (2.1), (2.2), and (2.3), consider the separation problem (SP) given below, where $\bar{x}_1^{k+1} \geqslant 0$.

$$\begin{aligned} \textbf{SP}: \quad & \text{Maximize} && \beta^t \nu_{j(k)}^k && (3.1a)\\ & \text{subject to} && \beta^t(\nu_j^k - \nu_{j(k)}^k) \geqslant 0 \quad \text{for } j = 0, \ldots, n+1, && (3.1b)\\ & && \beta_1 = \bar{x}_1^{k+1}. && (3.1c) \end{aligned}$$

Note that $\beta = \bar{x}^{k+1}$ is a feasible solution to SP with objective value $(\bar{x}^{k+1})^t \nu_{j(k)}^k \geqslant \mu \|\bar{x}^{k+1}\|^2 > \mu\varepsilon$ from (2.1). Hence, if β^* solves SP, then from (3.1b) the hyperplane $\beta^{*t}(x - \nu_{j(k)}^k) = 0$ will strongly separate the origin from $X(z^k)$. Moreover, from (3.1a), (3.1b), and (2.3), this hyperplane will permit a maximum improvement in the lower bound from among all such separating hyperplanes which support $X(z^k)$ at $\nu_{j(k)}^k$. Of course, (4.1) is itself a linear programming problem, and naturally, we do not require this to be solved optimally. Instead, we propose a Gauss-Seidel type of coordinate ascent heuristic which starts with the solution $\beta = \bar{x}^{k+1}$ and finds a feasible solution $\beta = \bar{\beta}$ with at least as good an objective value. Hence, in this case, we have

$$\overline{\beta}^t (x - \nu_{j(k)}^k) \geqslant 0 \quad \text{for all} \quad x \in X(z^k), \tag{3.2a}$$

and from (3.1a), (3.1c), and (2.3), we can revise the lower bound by

$$\overline{\Delta}^k = \frac{\overline{\beta}^t \nu_{j(k)}^k}{\overline{\beta}_1} \geqslant \Delta^k. \tag{3.2b}$$

To present this method, substitute $\beta = \bar{x}^{k+1} + \delta$ in SP in order to equivalently write this problem as

$$\text{Maximize} \quad \Big\{\delta^t \nu^k_{j(k)}\colon\ \delta^t(\nu^k_{j(k)} - \nu^k_j) \leqslant (\bar{x}^{k+1})^t(\nu^k_j - \nu^k_{j(k)})$$

$$\text{for} \quad j = 0,\dots,n+1, \quad \delta_1 \equiv 0,\ \delta_2,\dots,\delta_{m+2} \text{ unrestricted}\Big\}. \tag{3.3}$$

We present below a heuristic for approximately solving Problem (3.3), and hence determining a suitable rotation of the separating hyperplane.

Hyperplane Rotation Algorithm (HRA)

Initialization: Put $\delta = 0$ and let s_j, $j = 0,\dots,n+1$ be the corresponding nonnegative slack variable values for the inequalities in (3.3). Put row index $i = 2$, and proceed to Step 1.

Step 1: If $(\nu^k_{j(k)})_i < 0$, then go to Step 2a to possibly decrease δ_i. Similarly, if $(\nu^k_{j(k)})_i > 0$, then go to Step 2b to possibly increase δ_i. Otherwise, go to Step 3.

Step 2a (Decrease δ_i): Define

$$J_1 = \Big\{j \in \{0,\dots,n+1\}\colon\ (\nu^k_{j(k)} - \nu^k_j)_i < 0\Big\}.$$

If J_1 is empty, then go to Step 4. Otherwise, put

$$\delta_i = -\text{minimum}\left\{\frac{s_j}{-(\nu^k_{j(k)} - \nu^k_j)_i}\colon\ j \in J_1\right\}. \tag{3.4a}$$

Accordingly, revise the slack values s_j to $s_j - \delta_i(\nu^k_{j(k)} - \nu^k_j)_i$ for $j = 0,\dots,n+1$. Go to Step 3.

Step 2b (Increase δ_i): Define

$$J_2 = \Big\{j \in \{0,\dots,n+1\}\colon\ (\nu^k_{j(k)} - \nu^k_j)_i > 0\Big\}.$$

If J_2 is empty, go to Step 4. Otherwise put

$$\delta_i = \text{minimum}\left\{\frac{s_j}{(\nu^k_{j(k)} - \nu^k_j)_i}\colon\ j \in J_2\right\}. \tag{3.4b}$$

Accordingly, revise the slack values s_j to $s_j - \delta_i(\nu^k_{j(k)} - \nu^k_j)_i$ for $j = 0,\dots,n+1$. Go to Step 3.

Step 3: If $i = (m + 2)$, stop with $\overline{\beta} = \overline{x}^{k+1} + \delta$. Otherwise, increment i by one and return to Step 1.

Step 4: The problem (3.3) is unbounded, and so the lower bound can be increased indefinitely. Hence, terminate with the indication that the original linear program **LP** is infeasible.

Observe that the procedure HRA holds the first component of the normal to the separating hyperplane (2.2) fixed, and examines changing one component of this normal at a time in attempting to improve the objective value in (3.1), while maintaining the separation property, and hence augmenting the permissible increase in the lower bound z^k. For example, referring to the conceptual illustration in Fig. 1, note that $(\nu^k_{j(k)})_2 < 0$. Hence, at Step 2a of the procedure HRA, holding the first component of the normal to the separating hyperplane fixed, we can decrease the second component. Doing this to the maximum extent permissible rotates the separating hyperplane clockwise while being hinged at $\nu^k_{j(k)}$, until it supports the polytope along the face containing the minimizing solution. Hence, for this example, the rotation enables us to increase the lower bound right up to the optimal value for LP.

4. Obtaining dual solutions and polytope deformation.

The Polytope Sliding Algorithm PSA also inherently generates a sequence of dual feasible solutions of increasing dual objective function values, as we show below. Moreover, the dual solutions can be used to modify the algorithm based on a "reduced cost" representation of the linear program LP, so that all the vertices of the polytope under consideration have nonnegative z-axis components. This modification results in deforming the polytope in addition to sliding it along the negative z-axis.

To present this development, consider the dual (**DLP**) to LP of (1.2b), as stated below.

$$\textbf{DLP:}\quad \text{Maximize } \left\{\pi_0 \colon \pi^t g_j + \pi_0 \leqslant c_j \quad \text{for} \quad j = 0, \ldots, n+1\right\}. \tag{4.1}$$

At the Initialization Step of Algorithm PSA, when $k = 1$, let us define $\pi^1 \equiv 0$ and $\pi^1_0 = z^1 \equiv c_{\min}$. Note that (π^1, π^1_0) is a dual feasible solution with objective value z^1 in (4.1). Now, at any iteration $k \geqslant 1$, consider the linear program.

$$\text{Minimize}\left\{\left(C - (\pi^k)^t G\right)\lambda \colon G\lambda = 0,\ e\lambda = 1,\ \lambda \geqslant 0\right\}. \tag{4.2}$$

Note that because of the constraint $G\lambda = 0$, (4.2) is precisely the same as the linear program LP defined in (1.2b). In particular, z^k is a lower bound on the optimal value for this problem, and we are interested in finding a solution to the system

$$\left(C - (\pi^k)^t G - z^k e\right)\lambda = 0, \quad G\lambda = 0, \quad e\lambda = 1, \quad \lambda \geqslant 0$$

if one exists. Accordingly, we define

$$\nu_j^k = \begin{bmatrix} c_j - (\pi^k)^t g_j - z^k \\ g_j \end{bmatrix} \tag{4.3a}$$

for $j = 0, \ldots, n+1$, and denote

$$X(z^k) = conv\left\{\nu_j^k,\ j = 0, \ldots, n+1\right\}. \tag{4.3b}$$

Note that all the points ν_j^k, $j = 0, \ldots, n+1$, and hence points in $X(z^k)$, have nonnegative z-axis components since we are given that $(\pi^k, \pi_0^k \equiv z^k)$ is feasible to the dual (4.1). Now, as before, we use the closest point routine to either find a point $\overline{x}^{k+1} \in X(z^k)$ with $\|\overline{x}^{k+1}\|^2 \leqslant \varepsilon$, or to generate a strong separating hyperplane as in (3.2a), where $\overline{\beta}$ may be $\overline{x}^{k+1}$ itself. If $\overline{\beta}_1 \leqslant 0$, then LP is infeasible. Otherwise, from (3.2a), we get $\overline{\beta}^t(\nu_j^k - \nu_{j(k)}^k) \geqslant 0$ for all $j = 0, \ldots, n+1$. Now, partition $\overline{\beta} = (\overline{\beta}_1, \overline{\beta}_2)$ where $\overline{\beta}_1$ is the first component of $\overline{\beta}$, and $\overline{\beta}_2$ represents the remaining components. Then, noting (3.2b) and that $\overline{\beta}_1 > 0$, this last inequality yields $(\overline{\beta}^t \nu_j^k)/\overline{\beta}_1 \geqslant \overline{\Delta}^k$ for all $j = 0, \ldots, n+1$. Using (4.3a) to substitute for ν_j^k, this becomes

$$\left(\pi^k - \overline{\beta}_2/\overline{\beta}_1\right)^t g_j + (z^k + \overline{\Delta}^k) \leqslant c_j \quad \text{for all} \quad j = 0, \ldots, n+1. \tag{4.4}$$

Noting (4.1), we now have a revised improved dual feasible solution (π^{k+1}, π_0^{k+1}) given by

$$\pi^{k+1} = \pi^k - \overline{\beta}_2/\overline{\beta}_1, \quad \text{and} \quad \pi_0^{k+1} = z^k + \overline{\Delta}^k > \pi_0^k. \tag{4.5}$$

Setting $z^{k+1} = \pi_0^{k+1} \equiv z^k + \overline{\Delta}^k$, we now define ν_j^{k+1}, $j = 0, \ldots, n+1$, and $X(z^{k+1})$ as in (4.3), and repeat. Note that

$$\nu_j^{k+1} = \nu_j^k + \begin{bmatrix} \left(\overline{\beta}_2/\overline{\beta}_1\right)^t g_j - \overline{\Delta}^k \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4.6}$$

for $j = 0, \ldots, n+1$. Hence, the polytope $X(z^k)$ is slid along the negative z-axis by the amount $\overline{\Delta}^k$, and is also deformed by adjusting the first components of all the points ν_j^k, $j = 0, \ldots, n+1$, so that all vertices of the new polytope $X(z^{k+1})$ have nonnegative z-axis components. In particular, since $(\overline{\beta}^t \nu_{j(k)}^k)/\overline{\beta}_1 \equiv \overline{\Delta}^k$ from (3.2b), we have (4.4) holding as an equality for $j = j(k)$ (see the inequality that leads to (4.4)), and so from (4.3a) and (4.6), the first component of $\nu_{j(k)}^{k+1}$ is zero. The following section proves convergence of the algorithm under this modification.

REMARK 3. Note that Algorithm PSA as stated in Section 2 also produces a sequence of dual feasible solutions. At iteration $k = 1$, we have $(\pi^k, \pi_0^k) = (0, \ldots, 0, c_{\min})$ as a feasible solution to (4.1). Then, for any iteration $k \geqslant 1$, having determined the separating hyperplane (3.2a) with $\overline{\beta}_1 > 0$, we have $\overline{\beta}^t(\nu_j^k - \nu_{j(k)}^k) \geqslant 0$ for all $j = 0, \ldots, n+1$. Using (1.3) and (3.2b), we obtain as in (4.4) that

$$(-\overline{\beta}_2/\overline{\beta}_1)^t g_j + (z^k + \overline{\Delta}^k) \leqslant c_j$$

for all $j = 0, \ldots, n+1$, and so,

$$\begin{pmatrix} \pi^{k+1} \\ \pi_0^{k+1} \end{pmatrix} = \begin{pmatrix} -\overline{\beta}_2/\overline{\beta}_1 \\ z^k + \overline{\Delta}^k \end{pmatrix}$$

is a revised dual feasible solution with an improved objective value of $z^k + \overline{\Delta}^k$. However, the modification proposed above based on a reduced cost representation of LP helps enhance the computational performance of the algorithm by favorably positioning the polytope with respect to the nonnegative z-axis.

5. Overall algorithm and its convergence.

The overall Polytope Sliding, Hyperplane Rotation, and Deformation Algorithm (PSDA) may be summarized as follows.

Algorithm PSDA

Initialization. Put $k = 1$, $z^k = c_{\min}$, $\pi^k = (0, \ldots, 0)$, $\pi_0^k \equiv c_{\min}$, $\nu_j^k = \begin{bmatrix} c_j - c_{\min} \\ g_j \end{bmatrix}$ for $j = 0, \ldots, n+1$, $\lambda_0^k = 1/(Q+1)$, $\lambda_j^k = 0$ for $j = 1, \ldots, n$, $\lambda_{n+1}^k = Q/(Q+1)$, and

$$x^k = \sum_{j=0}^{n+1} \nu_j^k \lambda_j^k = \left[-c_{\min}, -b^t/(Q+1), 0\right]^t.$$

Note that $x^k \in X(z^k) \equiv conv\{\nu_j^k\colon j = 0, \ldots, n+1\}$. Pick termination tolerances $\varepsilon > 0$ and $\delta > 0$, and a constant $0 < \mu \leqslant 1$. Proceed to Step 1.

Step 1 (Subroutine CPR): Invoke a closest point routine CPR of Section 2, and let it produce a solution

$$\overline{x}^{k+1} \equiv p^T \equiv \sum_{j=0}^{n+1} \nu_j^k \lambda_j^{k+1} \in X(z^k). \tag{5.1}$$

If $\overline{x}_1^{k+1} \leqslant 0$, go to Step 4(b). If $\|\overline{x}^{k+1}\|^2 \leqslant \varepsilon$, go to Step 4(a). Otherwise, CPR also produces a $\nu_{j(k)}^k \equiv \nu_{j(T)}$ such that (2.1) and (2.2) hold true. In this case, proceed to Step 2.

Step 2 (Rotation of the separating hyperplane): Invoke Algorithm HRA of Section 4. If $J_1 = \varnothing$ or $J_2 = \varnothing$ at any step of this procedure, go to Step 4(b). Otherwise, let this algorithm produce a vector $\overline{\beta}$ such that $\overline{\beta}_1 = \overline{x}_1^{k+1}$, and $\overline{\beta}^t(x - \nu_{j(k)}^k) \geqslant 0$ for all $x \in X(z^k)$. Accordingly, as in (3.2b), find $\overline{\Delta}^k = (\overline{\beta}^t \nu_{j(k)}^k)/\overline{\beta}_1$ as a permissible increase in the lower bound z^k. If $\overline{\Delta}^k < \delta$, go to Step 4(a). Otherwise, proceed to Step 3.

Step 3 (Polytope deformation and sliding): Put $z^{k+1} = z^k + \overline{\Delta}^k$. If $z^{k+1} > c_{\max}$, go to Step 4(b). Otherwise, compute (π^{k+1}, π_0^{k+1}) as in (4.5), compute ν_j^{k+1} for $j = 0, \ldots, n+1$ as in (4.6), and using (4.6) and (4.1), let

$$x^{k+1} \equiv \sum_{j=0}^{n+1} \nu_j^{k+1} \lambda_j^{k+1} = \overline{x}^{k+1} + \begin{bmatrix} (\overline{\beta}_2/\overline{\beta}_1)^t \overline{x}_2^{k+1} - \overline{\Delta}^k \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{5.2}$$

where $\overline{\beta} \equiv (\overline{\beta}_1/\overline{\beta}_2)$ and $\overline{x}^{k+1} \equiv (\overline{x}_1^{k+1}, \overline{x}_2^{k+1})$. Increment k by one and return to Step 1.

Step 4(a) (Near-optimality): Terminate the algorithm with a (near)-optimal solution $\lambda^* = \lambda^k$. (In case $\overline{\Delta}^k < \delta$ at Step 2, this termination is due to insufficient progress. Also we can switchover to the simplex method as in Section 2 at this step.)

Step 4(b) (Infeasibility): Terminate with the indication that LP is infeasible.

Theorem 2 (Overall convergence result). *Suppose that Algorithm PSDA is run for some* $0 < \mu \leqslant 1$, *but with* $\varepsilon = 0$ *and* $\delta = 0$. *Then either*

(i) *CPR will generate a sequence* $\{p^t\}$ *in* $X(z^k)$ *for some* k *such that* $\overline{x}^{k+1} = p^T = 0$ *for some finite* $t = T$ *or such that* $\{p^t\} \to 0$, *or*

(ii) *the Algorithm PSDA will terminate finitely at Step* 4(b) *with an infeasibility indication, or*

(iii) *an infinite sequence* $\{\overline{x}^k\}$ *will be generated,* $k \geqslant 2$, *such that* $\{\overline{x}^k\} \to 0$. *If* $\overline{x}^k = p^T = 0$ *finitely in case* (i), *then the convex combination vector* λ^{k+1} *corresponding to* p^T *solves LP. Otherwise, in the first and the third cases, the limit point of any convergent subsequence of the convex combination weights generated, corresponding to* $\{p^t\}$ *in case* (i) *and to* $\{\overline{x}^{k+1}\}$ *in case* (iii), *solves LP.*

Proof. In the event case (i) occurs, let $\overline{\lambda}$ be the convex combination weight vector corresponding to p^T if $p^T = 0$ for some finite T, or let $\overline{\lambda}$ be the limit of a convergent subsequence of the convex combination weights corresponding to the sequence $\{p^t\}$ generated by CPR. In either case, we have

$$e\overline{\lambda} = 1, \quad \overline{\lambda} \geqslant 0, \quad \text{and} \quad \sum_{j=0}^{n+1} \nu_j^k \overline{\lambda}_j = 0.$$

From (4.3), this yields

$$G\overline{\lambda} = 0, \quad \text{and} \quad 0 = C\overline{\lambda} - (\pi^k)^t G\overline{\lambda} - z^k = C\overline{\lambda} - z^k.$$

Hence, we have

$$G\overline{\lambda} = 0, \quad e\overline{\lambda} = 1, \quad \overline{\lambda} \geqslant 0, \quad \text{and} \quad C\overline{\lambda} = z^k,$$

where z^k is a lower bound for LP. Therefore, $\overline{\lambda}$ solves LP. The event leading to termination at case (ii) implies that the dual (4.1) is unbounded or that LP is infeasible. This includes the event when $z^k > c_{\max}$ for any k, by noting the objective function of LP, and that the constraints include $e\lambda = 1$ and $\lambda \geqslant 0$.

Now, if cases (i) or (ii) do not occur, then as in case (iii), the Algorithm PSDA will generate an infinite sequence $\{\overline{x}^k\}$, $k \geqslant 2$, where $\overline{x}^k \in X(z^{k-1})$ and $x_1^k > 0$ for all $k \geqslant 2$. Noting (5.1), let $\{\lambda^k\}_K$, indexed by the set K, be a convergent subsequence of the corresponding convex combination weights generated, and suppose that $\{\lambda^k\}_K \to \overline{\lambda}$. Hence, $e\overline{\lambda} = 1$ and $\overline{\lambda} \geqslant 0$.

Now, from (2.1), (2.3) and (3.2b), we have

$$\begin{aligned} z^{k+1} &= z^k + \overline{\Delta}^k = z^k + \frac{\overline{\beta}^t \nu_{j(k)}^k}{\overline{\beta}_1} \geqslant z^k + \frac{(\overline{x}^{k+1})^t \nu_{j(k)}^k}{\overline{x}_1^{k+1}} \\ &\geqslant z^k + \frac{\mu \|\overline{x}^{k+1}\|^2}{\overline{x}_1^{k+1}} \geqslant z^k + \mu \overline{x}_1^{k+1} > z^k. \end{aligned} \tag{5.3}$$

Hence, $\{z^k\}$ is a monotone increasing sequence bounded above by $c_{\max}$ (see Step 3), and so $\{z^k\} \to \bar{z} \leqslant c_{\max}$. Moreover, this further implies from (5.3) that $\{\bar{x}_1^{k+1}\} \to 0$ and that $\{\|\bar{x}^{k+1}\|^2/\bar{x}_1^{k+1}\} \to 0$, and so $\{\bar{x}^k\} \to \underline{0}$. From (4.3) and (5.1), this in turn implies that $\{\bar{x}_2^k\} \equiv \{G\lambda^k\} \to 0$, and so, $G\bar{\lambda} = 0$.

Furthermore, from (4.3) and (5.1), since $\{\bar{x}_1^k\} \to 0$, we have

$$\left\{ \sum_{j=0}^{n+1} \lambda_j^k \left(c_j - (\pi^{k-1})^t g_j - z^{k-1}\right) \right\} \to 0. \tag{5.4}$$

Define $J = \{j \in \{0,\ldots,n+1\}\colon \bar{\lambda}_j > 0\}$. By dual feasibility of $(\pi^{k-1}, \pi_0^{k-1} \equiv z^{k-1})$ in (4.1), we know that $\left(c_j - (\pi^{k-1})^t g_j - z^{k-1}\right) \geqslant 0$ for all $j = 0,\ldots,n+1$ and for all $k \geqslant 2$. Hence, from (5.4), this implies that as $k \to \infty$, $k \in K$, we have

$$\left\{\left(c_j - (\pi^{k-1})^t g_j - z^{k-1}\right)\right\}_{k \in K} \to 0 \quad \text{for all} \quad j \in J. \tag{5.5}$$

Now, since $\sum_{j\in J} \bar{\lambda}_j = 1$, $\bar{\lambda}_j = 0$ for $j \notin J$, and $G\bar{\lambda} = 0$, we have

$$\sum_{j\in J} \bar{\lambda}_j \left(c_j - (\pi^{k-1})^t g_j - z^{k-1}\right) = \left(C\bar{\lambda} - z^{k-1}\right) \quad \text{for all} \quad k. \tag{5.6}$$

Taking limits in (5.6) as $k \to \infty$, $k \in K$, we have from (5.5) that $C\bar{\lambda} = \bar{z}$. Since $G\bar{\lambda} = 0$, $e\bar{\lambda} = 1$, $C\bar{\lambda} = \bar{z}$, where $\bar{z}$ is a lower bound for LP, it follows that $\bar{\lambda}$ solves LP, and this completes the proof.

COROLLARY 1. Algorithm PSDA is finitely convergent for any $\varepsilon > 0$ and $\delta > 0$.

6. Computational experience. We conclude this paper by providing some preliminary computational results for the proposed Algorithm PSDA. The test problems, denoted by *(m, n)* where *m* is the number of constraints and *n* is the number of variables, were generated in the form (1.1) following the approach in Rosen and Suzuki (1965) and Sherali and Myers (1988). These problems therefore have known primal and dual optimal solutions. The components of the optimal solution y^* were generated uniformly on [0, 10], while controlling the number of nonzeros to be less than *m* in order to inject degeneracy. The components of the optimal dual solution w^* were generated uniformly on $[-10, 10]$, and the coefficients of the $m \times n$ matrix A were generated uniformly on the interval $[-10, 10]$ having density 0.1. Accordingly, we then set $b = Ay^*$, and we generated the objective function coefficients by letting $c_j = w^{*t} a_j$ if $y_j^* > 0$, and

$c_j = w^{*t}a_j + \zeta$ if $y_j^* = 0$, where ζ was generated uniformly on [0, 10]. The value of Q employed in (1.2a) was taken as twice the sum of all the primal variables at optimality.

Note that the subroutine CPR is the most expensive step of Algorithm PSDA. We found that the number of iterations of the algorithm tends to be small as in interior point methods, provided that the closest point subproblems are solved fairly accurately ($\mu \geqslant 0.8$). We attempted the algorithm of Sherali and Choi (1993), Wolfe (1975), and a specialization of the first-order reduced gradient method, noting that this closest point problem minimizes a quadratic function over a simplex. Of these, the algorithm of Sherali and Choi (1993) worked best, and was adopted in the preliminary results reported below. However, a second-order conjugate gradient or quasi-Newton reduced gradient type of approach might serve to enhance the performance of Algorithm PSDA.

Tables 1 and 2 present some computational results for Algorithm PSDA, using two different values of μ, namely, 0.75 and 0.80 (denoted by PSDA (0.75) and PSDA (0.80), respectively). Also, for comparison, we provide computational results obtained using the Revised Simplex method, implemented with an efficiency on par with Algorithm PSDA. The algorithms were coded in FORTRAN VS2 and run on an IBM 3090–300E computer. For each problem, we specify its size (m, n) and the optimal value z^* obtained by the simplex method when terminated with reduced costs greater than or equal to -0.01. In all runs, we use $\varepsilon = 0.01$ and $\delta = 0.5$. The test problems of Table 2 have more zeros at the optimal solution than the problems of Table 1, implying the possibility of a higher degree of degeneracy. Indeed, this is borne out by comparing the number of iterations taken by the Simplex method for the problems in the two tables. On the other hand, note that the proposed algorithm is relatively unaffected by degeneracy. Also, the relative advantage of PSDA improves with an increase in problem size (m and n), and also with an increase in n for any given value of m.

To summarize, we have presented in this paper a polytope sliding and deformation algorithm based on a convex hull representation interpretation of Karmarkar's form of writing any given linear program. We have also shown how a sequence of dual feasible solutions of increasing objective value can be recovered in the process. This may prompt the application of the proposed method to the dual linear program, instead of the primal. Refinements of the algorithm based on a rotation of the separating hyperplane, along with a deformation using the dual solution have been proposed and implemented. The principal advantage of the algorithm is that it is easy to implement, requires a minimal amount of storage, and is not prone to accumulated round-off error problems, thereby requiring no sophisticated numerical safeguards. For the test problems run, it usually produces

Table 1. Computational experience on low-moderately degenerate problems problems

Problem (m, n) z^*	Algorithm	Iters (K)	LB (z^K)	CPU Time (secs.)	*OPT(%)
(40, 120)	SIMPLEX	238		2.64	
−1.50	PSDA (0.75)	24	−1.64	3.53	91.39
	PSDA (0.80)	23	−1.52	6.49	98.34
(40, 300)	SIMPLEX	291		6.37	
4.26	PSDA (0.75)	25	4.17	4.77	98.02
	PSDA (0.80)	23	4.22	5.50	99.44
(50, 150)	SIMPLEX	238		4.30	
−6.53	PSDA (0.75)	29	−6.59	9.80	99.17
	PSDA (0.80)	27	−6.56	11.42	99.51
(50, 500)	SIMPLEX	432		17.96	
4.36	PSDA (0.75)	28	4.27	9.89	98.00
	PSDA (0.80)	21	4.32	10.66	99.00
(75, 250)	SIMPLEX	598		23.65	
13.25	PSDA (0.75)	36	13.08	23.70	98.73
	PSDA (0.80)	33	13.17	29.71	99.38
(75, 500)	SIMPLEX	538		36.18	
3.81	PSDA (0.75)	31	3.72	21.54	97.58
	PSDA (0.80)	28	3.72	22.07	97.62
(100, 600)	SIMPLEX	1108		112.53	
3.15	PSDA (0.75)	39	3.10	49.80	98.25
	PSDA (0.80)	36	3.12	56.53	98.99
(100, 600)	SIMPLEX	930		97.15	
−2.40	PSDA (0.75)	41	−2.47	79.75	97.03
	PSDA (0.80)	37	−2.53	87.52	94.70

*OPT(%)= $\left[1 - \frac{(z^* - LB)}{|z^*|}\right] \times 100(\%)$

Table 2. Computational experience on more degenerate problems

Problem (m, n) z^*	Algorithm	Iters (K)	LB (z^K)	CPU Time (secs.)	*OPT(%)
(40, 120)	SIMPLEX	209		2.40	
−2.80	PSDA (0.75)	25	−2.93	7.75	95.55
	PSDA (0.80)	23	−2.91	9.03	96.40
(40, 300)	SIMPLEX	290		6.45	
−8.30	PSDA (0.75)	30	−8.41	6.76	98.64
	PSDA (0.80)	28	−8.34	7.56	99.57
(50, 150)	SIMPLEX	499		8.06	
−16.33	PSDA (0.75)	33	−16.50	15.84	98.98
	PSDA (0.80)	30	−16.40	21.31	99.58
(50, 500)	SIMPLEX	417		17.80	
−8.58	PSDA (0.75)	27	−8.64	13.15	99.37
	PSDA (0.80)	25	−8.59	17.59	99.89
(75, 250)	SIMPLEX	701		27.12	
14.67	PSDA (0.75)	38	14.53	44.39	99.01
	PSDA (0.80)	34	14.59	55.61	99.42
(75, 500)	SIMPLEX	966		60.62	
33.95	PSDA (0.75)	31	33.91	26.52	99.86
	PSDA (0.80)	28	33.88	26.12	99.77
(100, 600)	SIMPLEX	1562		153.60	
4.56	PSDA (0.75)	42	4.43	73.99	97.14
	PSDA (0.80)	38	4.42	82.70	97.08
(100, 600)	SIMPLEX	1274		127.38	
−2.12	PSDA (0.75)	42	−2.22	69.25	95.30
	PSDA (0.80)	38	−2.15	87.80	98.56

*OPT(%)= $\left[1 - \frac{(z^* - LB)}{|z^*|}\right] \times 100(\%)$

a quick, acceptable near optimal solution (95–99%) within a limit of 18–45 iterations (depending on m and n). Further enhancements are possible by using a more efficient closest point subroutine. As another point of research interest, one can investigate an appropriate switchover point when using this procedure to provide an advanced-start solution for the simplex algorithm. These issues and additional tests on various structured problems are recommended for future research.

REFERENCES

Bazaraa, M.S., J.J. Jarvis and H.D. Sherali (1990). *Linear Programming and Network Flows*, Second edition. John Wiley and Sons, Inc., New York, NY.

Bazaraa, M.S., H.D. Sherali and C.M. Shetty (1993). *Nonlinear Programming: Theory and Algorithms*, Second edition. John Wiley and Sons, Inc., New York, NY.

Gacs, P., and L. Lovasz (1981). Khachian's algorithm for linear programming. *Mathematical Programming Study*, **14**, 61–68.

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

Parker, G.R., and R.L. Rardin (1988). *Discrete Optimization*. Academic Press, Inc., San Diego, CA 92101.

Rosen, J., and S. Suzuki (1965). Construction of nonlinear programming test problems. *Communications of the ACM*, **8**.

Sherali, H.D., and G. Choi (1993). Finding the closest point to the origin in the convex hull of a discrete set of points. *Computers and Operations Research*, **20**(4), 363–370.

Sherali, H.D., and D.L. Myers (1988). Dual formulations and subgradient optimization strategies for linear programming relaxations of mixed-integer programs. *Discrete Applied Mathematics*, **20**, 51–68.

Tomlin, J.A. (1985). An Experimental Approach to Karmarkar's Projective Method for Linear Programming. Ketron, Inc., Mountain View, CA 94040.

Wolfe, P. (1976). Finding the nearest point in a polytope. *Mathematical Programming*, **11**, 128–149.

Received September 1997

H.D. Sherali obtained his Ph.D. from the Georgia Institute of Technology in 1979. He is currently the Charles O. Gordon Professor of Industrial and Systems Engineering at Virginia Polytechnic Institute and State Universtiy. His areas of research interests are in convex and nonconvex (discrete and continuous) optimization with applications to production, distribution, and engineering design problems.

G. Choi obtained his Ph.D. from Virginia Polytechnic Institute and State University in 1994. He is currently Assistant Professor of Industrial Engineering at Hanyang University, Seoul, Korea. His research interests are in nondifferentiable (linear and nonlinear) optimization, logistics, and applications to information systems.

S. Sen is Professor of Systems and Industrial Engineering at the University of Arizona. He has been with the University since 1982, the year he graduated from Virginia Tech with a Ph.D. in Operations Research. His research covers a wide spectrum of optimization theory and its applications, and his papers have appeared in journals like JOTA, Mathematical Programming, Mathematics of Operations Research, Management Science and Operations Research.

IŠORINIO TAŠKO SLANKAUS IR DEFORMUOJAMO POLITOPO ALGORITMAS TIESINIO PROGRAMAVIMO UŽDAVINIAMS SPRĘSTI

Hanif D. SHERALI, Gyunghyun CHOI, Suvrajeet SEN

Šiame straipsnyje mes vystome tiesinio programavimo algoritmą naujai interpretuodami Karmarkaro pateiktą šio uždavinio traktavimą. Tam nagrinėjame politopą, kurio atžvilgiu pradinis taškas yra išorinis taškas. Siekdami optimalaus sprendinio nustatome minimalų poslinkį, kuriuo reikia paslinkti politopą vienmatės poslinkio ašies kryptimi taip, kad pradinis taškas priklausytų tam politopui. Tam, naudodami artimiausio taško procedūrą, pravedame hiperplokštumas tarp pradinio taško ir politopo. Algoritmas pagerėja įvedus dualių sprendinių generavimą. Šie sprendiniai leidžia deformuoti politopą taip, kad jo padėtis būtų patogi pradinio taško ir poslinkio ašies atžvilgiu. Bendra algoritmo schema yra lengvai realizuojama, reikalauja mažai atminties. Ji greitai duoda kokybiškus sprendinio įverčius. Yra galimybė pereiti į simplekso ar vidinio taško metodą, naudojant jau gautą sprendinį kaip pradinį. Pateikiami preliminarūs skaičiavimų rezultatai.