Proc. Indian Acad. Sci. (Math. Sci.), Vol. 110, No. 1, February 2000, pp. 79–101.

A direct heuristic algorithm for linear programming

S K SEN and A RAMFUL*

Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560 012, India
*Department of Mathematics, University of Mauritius, Reduit, Mauritius

MS received 13 August 1998

Abstract. An $O(n^3)$ mathematically non-iterative heuristic procedure that needs no artificial variable is presented for solving linear programming problems. An optimality test is included. Numerical experiments depict the utility/scope of such a procedure.

Keywords. Direct heuristic algorithm for linear programming; interior-point methods; optimality test; p-inverse; revised simplex algorithm.

1. Introduction

The simplex method [6, 23] – an exponential time (non-polynomial time) algorithm – or its variation has been used and is being used to solve almost any linear programming problem (LPP) for the last four decades. In 1979, Khachiyan proposed the ellipsoid method – the first polynomial-time (interior-point) algorithm – to solve LPPs [13]. Then, in 1984, Karmarkar suggested the second polynomial time $(O(n^{3.5}))$ algorithm based on projective transformation [11, 12, 22, 24]. Unlike the ellipsoid method, Karmarkar method appears to solve very large LPPs faster than does the simplex method. However, most of the available software packages that we know of for solving LPPs are still based on the simplex algorithm (SA) or a variation of it. Both the ellipsoid method and the Karmarkar method are mathematically iterative and need many times more computing resources than does the SA certainly for small LPPs and possibly for reasonably large LPPs. Consequently, none of these are so far popular nor are they known to be used commercially extensively like the SA.

There exists no mathematically direct algorithm to solve LPPs like the ones (e.g., Gauss reduction with partial pivoting) to solve linear systems. In fact, if we view the LPP geometrically it would not be far to see why it is difficult to have a non-iterative algorithm (where the exact number of arithmetic operations is known *a priori*).

The word iteration has one meaning in computer science and a different meaning in mathematics. For example, the multiplication of two $n \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ to get the matrix $C = [c_{ij}]$, where $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$, in the usual way is iterative in computer science while it is non-iterative or, equivalently, direct (i.e., we know the exact number of operations to obtain c_{ij} beforehand) in mathematics. In fact, iteration means simply repetition in computer science.

Each linear equation in a linear system represents a hyperplane geometrically. The intersection of all the hyperplanes corresponding to the equations in the linear system will be a convex region. The portion of this region that falls [32, 19] in the first quadrant (i.e. in the non-negative region) is defined as a polytope.

In the LPP "minimise $\mathbf{c}^t\mathbf{x}$ (objective function) subject to $\mathbf{Ax} = \mathbf{b}$ (linear constraints), $\mathbf{x} \geq \mathbf{0}$ (non-negativity condition)", $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ (null column vector) define the polytope. One of the corners of the polytope is the required solution. We know *exactly* the direction of search, viz., the direction of the vector $\mathbf{c}$; but we do not know the point or location from which we should start in the direction of $\mathbf{c}$. If we already know this location then all LPPs can be solved mathematically directly just as we can solve linear equations directly. Since we do not have the knowledge of this location, we use some kind of trial and error procedure or a procedure which is implicitly a trial and error one to finally hit upon the optimal solution. For example, in the polytope-shrinking interior-point method [19] we start from a centre (middle) of the polytope (which appears quite reasonable) and proceed in the direction of $\mathbf{c}$. Since the centre does not happen to be the required location, we hit a hyperplane instead of the desired corner. This will help in deciding the next location by some means; for example, a hyperplane normal to $\mathbf{c}$ could be drawn from the point of hit such that the resulting (greatly shrunk) polytope above the hyperplane will be the one for the next search. We again start from a centre of the shrunk polytope and proceed in the direction of $\mathbf{c}$. We continue this process till either we hit the desired corner or determine this corner uniquely from the remaining hyperplanes by solving the equations corresponding to these hyperplanes.

Can we really solve LPPs directly in $O(n^3)$ operations just like the way we solve linear systems? The proposed procedure is essentially an attempt to answer this question. We have been able to find out a few problems where the procedure does not give the optimal solution. However, even if it does not, this heuristic procedure still gives a basic feasible solution quite close to the actual optimal solution with, however, a set of basic variables different from the actual ones. One can make use of this solution to obtain the optimal solution in a fewer iterations through the revised simplex procedure [35] or a variation of it.

In § 2, we describe the direct heuristic algorithm for linear programming (DHALP) and discuss a few results concerning the algorithm. We illustrate the procedure by numerical examples and state our observations including effectiveness of the procedure through numerical experiments in § 3, while in § 4 we include the conclusions and specifically demonstrate that the proposed heuristic algorithm is distinctly different from the popular SA not only in not considering the artificial variables but also in the detection of the basic variables deterministically. Also we compare the DHALP with interior-point [13, 19, 29, 36] and other methods including the inequality sorting algorithms [15, 17].

2. Direct heuristic algorithm

We first present here the LPP along with the DHALP without any comment or justification and then discuss a few results on the DHALP and on its computational/space complexity.

The LPP. Let the LPP be written in the form (without loss of generality)

$$\text{minimize } z = \mathbf{c}^t\mathbf{x} \text{ subject to } A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \tag{1}$$

where $A = [a_{ij}]$ is an $m \times n$ constraint matrix, $\mathbf{c} = [c_i]$ is an $n \times 1$ column vector, $\mathbf{b} = [b_j]$ is an $m \times 1$ column vector, t indicates the transpose, and $\mathbf{0}$ represents the $n \times 1$ null column vector.

2.1 *The DHALP with optimality test*

The inputs are $A, \mathbf{b}, \mathbf{c}$ while the outputs are the $n \times 1$ solution vector $\mathbf{x} = [x_i]$, the value of the objective function z, and the comments based on the optimality test.

S.1: Input $m, n, A = [a_{ij}], i = 1(1)m; j = 1(1)n, b = [b_j],\ j = 1(1)m, \mathbf{c} = [c_i], i = 1(1)n$, where $i = 1(1)m$ implies $i = 1, 2, 3, \ldots, m$.

S.1a: Initialize indexarray by n zeros.

S.2: Compute $\mathbf{d} = A^+\mathbf{b}$, where $\mathbf{d} = [d_i]$ is an $n \times 1$ vector and A^+ is the Moore-Penrose inverse or, equivalently, minimum norm least square inverse or, equivalently, pseudoinverse or, equivalently, p-inverse [21, 25, 26, 10, 8, 34, 27, 28, 4, 30, 31, 5, 7, 20, 14, 16] $\mathbf{e} = A\mathbf{d}$.

If $\mathbf{e} \neq \mathbf{b}$ then output 'The LPP is inconsistent or, equivalently, infeasible.' Terminate.

S.3: Compute

$$
\begin{aligned}
H &= A^+A, \\
\mathbf{c}' &= (I - H)\mathbf{c}, \\
s_k &= \min\left\{\frac{d_i}{c_i'}; c_i' > 0\right\}, \\
\mathbf{x} &= \mathbf{d} - \mathbf{c}'s_k,
\end{aligned}
$$

where $\mathbf{c}' = [c_i']$ is an $n \times 1$ vector, I is an $n \times n$ unit matrix, H is an $n \times n$ matrix. The direction vector $\mathbf{c}'s_k$ attempts to push the point (the solution vector $\mathbf{x}$) of the solution space defined by $A\mathbf{x} = \mathbf{b}$ into the polytope defined by $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ if it is not already in the polytope. The successive computation produces the point (the required solution vector $\mathbf{x}$ of the LPP) at one of the corners of the polytope.

S.4: Remove the x_i that becomes zero, remove the (corresponding) ith column vector of the constraint matrix A, and the (corresponding) element c_i from the $\mathbf{c}$-vector of the objective function, shrink A and $\mathbf{c}$ and maintain an index counter for the variable x_i that has been removed. Reduce the dimension n of A by one, n of $\mathbf{c}$ by one. Compute $\mathbf{d} = A^+\mathbf{b}$.

Remarks. The foregoing A^+ in the step S.4 is computed from the current (most recent) A^+. This computation takes $O(mn)$ operations. One could have computed A^+ from the shrunk A only, but this would have taken $O(mn^2)$ operations. If two or more x_i become zero then one has to execute the algorithm removing one x_i at a time and computing the corresponding final solution $\mathbf{x}$. The $\mathbf{x}$ that gives the minimum value of the objective function will be the required solution (output) of the DHALP. There is no easy way to decide which one of the two (or more) null x_i should leave the basis. However, such a situation is not too common.

S.5: Repeat the steps S.3 and S.4 till s_k is zero or not computable (i.e., $c_i' \leq 0$ for all i).

S.6: Compute the value of the objective function $z = \mathbf{c}^t\mathbf{x}$ where $\mathbf{x}$ and $\mathbf{c}$ are the most recent vectors and output the results.

The optimality test. Let the resulting solution vector $\mathbf{x}$ that we obtain from the DHALP, the quantities A, $\mathbf{b}$, and $\mathbf{c}$ of eq. (1) be known. Also, let B be the basis (columns of A corresponding to the basic variables x_i of the foregoing vector $\mathbf{x}$), $\mathbf{c}_B^t$ be the vector with the elements of $\mathbf{c}^t$ corresponding to the basic variables x_i, and $\mathbf{p}_j$ be the jth nonbasic column of A (i.e., the column of A whose coefficient x_j is a nonbasic variable).

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1}$ (a row vector).

S.8: Compute $z_j - c_j = \mathbf{y}^t \mathbf{p}_j - c_j$ (a scalar) for all nonbasic vectors $\mathbf{p}_j$.

S.9: Test the sign of $z_j - c_j$. If all $z_j - c_j \leq 0$ then the solution is optimal (minimal) else the solution is unbounded (no lower bound for the minimization problem) or the DHALP could not reach the optimal solution; use the revised SA [35] to obtain the optimal solution starting from the results of DHALP, i.e., using the DHALP solution as the initial basic feasible solution.

2.2 *Justification of the DHALP*

The step S.1 of the DHALP is simply the input step. The step S.2 checks the consistency of the constraint equation $A\mathbf{x} = \mathbf{b}$ before proceeding further. If the vector $\mathbf{e} = AA^+\mathbf{b} \neq \mathbf{b}$ then the equation $A\mathbf{x} = \mathbf{b}$ is inconsistent [28]. Hence the LPP is infeasible and we terminate the DHALP. A^+ will be termed as the p-inverse of A in the rest of the article. 'Why the term p-inverse?' is explained in the article by Lakshmikantham *et al* [16].

The general solution of the linear system $A\mathbf{x} = \mathbf{b}$, where the vector $\mathbf{b}$ may be zero or not, is $\mathbf{x} = A^+\mathbf{b} \pm P\mathbf{z}$, where $P = (I - A^+A)$ is the orthogonal projection operator that projects any arbitrary vector $\mathbf{z}$ orthogonally onto the null space of the matrix A. We are essentially computing a point (represented by a vector in an n-dimensional space) $\mathbf{c}'$ in the null space of the matrix A in the step S.3. If we write $\mathbf{x} = \mathbf{d} - \mathbf{c}'s_k = A^+\mathbf{b} - (I - A^+A)\mathbf{c}s_k$, we can easily see that the solution vector $\mathbf{x}$ is of the form $A^+\mathbf{b} - P\mathbf{z}$, where $\mathbf{c}s_k$ corresponds to the arbitrary column vector $\mathbf{z}$. The scalar s_k in the step S.3 is computed so as to (i) make one (or more) element x_i of $\mathbf{x}$ zero, i.e., to make x_i nonbasic and (ii) reduce the value of the objective function. The step S.3 has also the effect of pushing the solution vector $\mathbf{x}$ into the polytope [19, 32] if it is not already in it. Sometimes, rarely though, a true basic variable x_i may turn out to be zero and hence nonbasic. Under which necessary and sufficient condition does an actual basic variable become nonbasic or, equivalently, what would be the conditions/restrictions on A, $\mathbf{b}$ and $\mathbf{c}$ so that the DHALP becomes a regular direct algorithm? This is an open problem which needs to be explored.

In the step S.4 we remove the variable x_i whose value has become zero once for all – quite often this x_i is nonbasic; at least in our numerical experiment, in over 95% problems, x_i with zero has been nonbasic.

The step S.5 includes a stopping condition while the step S.6 is the output step in which we may include appropriate comments for degenerate, infeasible, unbounded (without a lower bound) or infinite solution cases.

The optimality test is carried out in the steps S.7, S.8, and S.9. Usually the basis B will be a nonsingular (square) matrix. Testing for the optimality of the solution is straightforward. If B is rectangular $(m < n)$ then we append one (or more) row to B and one (or more) corresponding element to $\mathbf{b}$ so that the resulting matrix B and resulting vector $\mathbf{b}$ do not change $\mathbf{x}$, and B turns out to be square nonsingular. We then carry out the optimality test as in steps S.7, S.8, and S.9.

If the test fails, i.e., if not all $z_j - c_j \leq 0$ then we may use this basis B, without wasting/ throwing away the computation due to the DHALP, in the revised SA to get the optimal solution in a comparatively few steps.

2.3 *Complexity of the DHALP*

We present here the order of computational and space complexities for the DHALP. We need, in the step S.2, (i) $O(mn^2)$ operations to compute the p-inverse of A, i.e., A^+ for the

$m \times n$ constraint matrix A [31, 10, 14], (ii) $O(mn)$ operations to compute the $n \times 1$ vector $\mathbf{d} = A^+\mathbf{b}$ and (iii) $O(mn)$ operations to compute the $m \times 1$ vector $\mathbf{e} = A\mathbf{d}$.

In step S.3, we may compute the orthogonal projection operator $P = (I - H) = (I - A^+A)$ directly (without computing A^+ or A^+A) using the concise algorithm for linear systems [18, 32, 19] However, instead, we compute the $n \times n$ matrix $H = A^+A$ in $O(mn^2)$ operations, $\mathbf{c}' = (I - H)\mathbf{c}$ in $O(n^2)$ operations, the scalar s_k in $O(n)$ operations, and the $n \times 1$ solution vector $\mathbf{x} = \mathbf{d} - \mathbf{c}'s_k$ in $O(n)$ operations.

We need, in the step S.4, $O(mn)$ operations for the removal of one column of A, one element of $\mathbf{c}$, one dimension for the number of columns n of A, and for shrinking A and $\mathbf{c}$ including keeping an index counter for the variable $\mathbf{x}_i$ that has been removed. For computing $\mathbf{d}$ in this step we need not have to compute A^+ from the original shrunk matrix A; instead, we compute A^+ from the most recently computed A^+. Thus we need $O(mn)$ operations to compute $\mathbf{d}$.

While repeating step S.5, we need not have to compute the p-inverse A^+ of the shrunk matrix A (as stated in the foregoing paragraph), that needs $O(mn^2)$ operations. We compute, instead, the new A^+ from the most recently computed A^+ in $O(mn)$ operations by the procedure PISM:

The procedure PISM: Let the current matrix A and its known p-inverse A^+ be denoted by A_{k+1} and A^+_{k+1}, respectively where $A_k = A_{k+1}$ without the qth column. We know both A_{k+1} and A^+_{k+1} as well as the column number q. Also, let $A^+_{k+1-q} = A^+_{k+1}$ without the qth row, $\mathbf{a}_q = q$th column of A_{k+1}, and $\mathbf{b}^t_q = q$th row of A^+_{k+1}. Then

$$A^+_k = A^+_{k+1-q} + \frac{1}{r}(A^+_{k+1-q}\mathbf{a}_q)\mathbf{b}^t_q, \tag{2}$$

where

$$r = 1 - \mathbf{b}^t_q\mathbf{a}_q. \tag{3}$$

Here r needs $O(m)$ operations, $A^+_{k+1-q}\mathbf{a}_q$ needs $O(nm)$ operations. The foregoing addition takes $O(mn)$ operations. Hence, the computation of each successive shrunk A^+ needs $O(mn)$ operations. The validity of eq. (2) can be easily verified by working backwards in Greville's algorithm where the qth row of $A^+_{k+1} = \mathbf{b}^t_q$ is already known [10, 14].

The computational complexity of the optimality test is as follows. Step S.7 needs $O(mn)$ operations to compute $\mathbf{y}^t$ for a nondegenerate bounded LPP – here B^{-1} is just the final A^+. Otherwise, it will take $O(m^3)$ operations. The step S.8 on the other hand, requires just $O(m)$ operations to compute $z_j - c_j$.

So far as the space complexity is concerned, we mainly need mn locations to store the constraint matrix A, m and n locations to store $\mathbf{b}$ and $\mathbf{c}$, respectively. During the successive removal of the elements of $\mathbf{x}$, we will be shrinking A as well as $\mathbf{c}$. Although this will reduce the need for the storage locations, this reduction may not be significant. We also need storage space for the program corresponding to the DHALP, which is not significant. Hence the space complexity is $O(mn)$.

3. Examples

We illustrate the DHALP by considering a few typical numerical examples. We also present an example where the DHALP has given a solution close to the optimal one (but not the optimal one), and demonstrate how to arrive at the optimal solution starting from the output (solution) of the DHALP.

Example 1 (*Nondegenerate bounded solution*). The LPP in an inequality form is

$$\min z = -1.2x_1 - 1.4x_2 \quad \text{subject to (s.t.)}$$
$$40x_1 + 25x_2 \leq 1000$$
$$35x_1 + 28x_2 \leq 980$$
$$25x_1 + 35x_2 \leq 875$$
$$x_1, x_2 \geq 0$$

We write the LPP as the one with equality constraints as follows

$$\min z = \mathbf{c}'\mathbf{x} \text{ s.t.}$$
$$A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0},$$

where

$$\mathbf{c} = \begin{bmatrix} -1.2 \\ -1.4 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, \quad A = \begin{bmatrix} 40 & 25 & 1 & 0 & 0 \\ 35 & 28 & 0 & 1 & 0 \\ 25 & 35 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix} 1000 \\ 980 \\ 875 \end{bmatrix}, \quad m = 3, n = 5.$$

S.1: The inputs are the numerical vectors **c** and **b**, and the numerical matrix A.

S.1a: indexarray $= [0\ 0\ 0\ 0\ 0]^t$.

$$\text{S.2: } \mathbf{d} = A^+\mathbf{b} = \overbrace{\begin{bmatrix} 0.0361 & 0.0130 & -0.0361 \\ -0.0296 & -0.0035 & 0.0524 \\ 0.2966 & -0.4340 & 0.1345 \\ -0.4340 & 0.6417 & -0.2035 \\ 0.1345 & -0.2035 & 0.0682 \end{bmatrix}}^{A^+} \overbrace{\begin{bmatrix} 1000 \\ 980 \\ 875 \end{bmatrix}}^{\mathbf{b}} = \overbrace{\begin{bmatrix} 17.2657 \\ 12.8164 \\ -11.0406 \\ 16.8388 \\ -5.2187 \end{bmatrix}}^{\mathbf{d}}.$$

$$\mathbf{e} = A\mathbf{d} = [1000\ 980\ 875]^t.$$

Here $\mathbf{e} = \mathbf{b}$. (Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.)

$$\text{S.3: } H = A^+A = \overbrace{\begin{bmatrix} 0.9972 & 0.0030 & 0.0361 & 0.0130 & -0.0361 \\ 0.0030 & 0.9963 & -0.0296 & -0.0035 & 0.0524 \\ 0.0361 & -0.0296 & 0.2966 & -0.4340 & 0.1345 \\ 0.0130 & -0.0035 & -0.4340 & 0.6417 & -0.2035 \\ -0.0361 & 0.0524 & 0.1345 & -0.2035 & 0.0682 \end{bmatrix}}^{H}$$

$$\mathbf{c}' = (I - H)\mathbf{c} = [0.0009\ \ -0.0015\ \ 0.0018\ \ 0.0107\ \ 0.0300]^t$$

$$s_k = \min\left\{\frac{d_i}{c_i'} : c_i' > 0\right\} = \min\left\{\frac{17.2657}{0.0009}, \frac{-11.0406}{0.0018}, \frac{16.8388}{0.0107}, \frac{-5.2187}{0.0300}\right\}.$$

$$s_k = \min\{19486, -5997, 1567, -174\} = -5997.$$

$$\mathbf{x} = \mathbf{d} - \mathbf{c}' s_k = [22.5793 \;\; 3.8732 \;\; 0 \;\; 81.2767 \;\; 174.9568]^t.$$

S.4: We remove x_3 since it has become zero, i.e., nonbasic. Hence we remove the third column vector of A and the third element of $\mathbf{c}$, i.e., c_3. We then shrink the 3×5 matrix A and the 5×1 vector $\mathbf{c}$ to the 3×4 matrix and 4×1 vector and call them once again A and $\mathbf{c}$, respectively. The indexarray that keeps track of which element of $\mathbf{x}$ has become 0, i.e., nonbasic, now becomes $[0\ 0\ 3\ 0\ 0]^t$. Replace n by $n-1$, i.e., n is now 4.

$$\mathbf{d} = A^+\mathbf{b} = [16.6990 \;\; 13.2815 \;\; 23.6506 \;\; -7.3298]^t.$$

S.5: Now we go back to the step S.3.

$$\text{S.3: } H = \begin{bmatrix} 0.9991 & 0.0015 & -0.0092 & -0.0292 \\ 0.0015 & 0.9976 & 0.0148 & 0.0468 \\ -0.0092 & 0.0148 & 0.9094 & -0.2864 \\ -0.0292 & 0.0468 & -0.2864 & 0.0939 \end{bmatrix}$$

We compute the new A^+ from the current A^+ in $O(mn)$ operations using the procedure PISM as follows. Let the current matrix A be denoted by A_{k+1}, i.e.,

$$A_{k+1} = \begin{bmatrix} 40 & 25 & 1 & 0 & 0 \\ 35 & 28 & 0 & 1 & 0 \\ 25 & 35 & 0 & 0 & 1 \end{bmatrix}.$$

Let the current p-inverse of A be denoted by A^+_{k+1}, i.e.,

$$A^+_{k+1} = \begin{bmatrix} 0.0361 & 0.0130 & -0.0361 \\ -0.0291 & -0.0035 & 0.0524 \\ 0.2966 & -0.4340 & 0.1345 \\ -0.4340 & 0.6417 & -0.2035 \\ 0.1345 & -0.2035 & 0.0682 \end{bmatrix}.$$

Let $A_k = A_{k+1}$ without the qth column. Here $q = 3$. So we remove the third column of the original matrix A. Hence

$$A_k = \begin{bmatrix} 40 & 25 & 0 & 0 \\ 35 & 28 & 1 & 0 \\ 25 & 35 & 0 & 1 \end{bmatrix}.$$

Moreover, let $A^+_{k+1-3} = A^+_{k+1}$ without the third row, i.e.,

$$A^+_{k+1-3} = \begin{bmatrix} 0.0361 & 0.0130 & -0.0361 \\ -0.0291 & -0.0035 & 0.0524 \\ -0.4340 & 0.6417 & -0.2035 \\ 0.1345 & -0.2035 & 0.0682 \end{bmatrix}.$$

Let $\mathbf{a}_3 =$ third column of A_{k+1}, i.e., $\mathbf{a}_3 = [1 \;\; 0 \;\; 0]^t$. Let $\mathbf{b}^t_3 =$ third row of A^+_{k+1}, i.e., $\mathbf{b}^t_3 = [0.2966 \;\; -0.4340 \;\; 0.1345]$. Compute $r = 1 - \mathbf{b}^t_3\mathbf{a}_3 = 0.7034$. Compute

$$A_k^+ = A_{k+1-3}^+ + \frac{1}{r}(A_{k+1-3}^+ \mathbf{a}_3)\mathbf{b}_3^t = \begin{bmatrix} 0.0513 & -0.0092 & -0.0292 \\ -0.0421 & 0.0148 & 0.0468 \\ -0.6170 & 0.9094 & -0.2864 \\ 0.1912 & -0.2864 & 0.0939 \end{bmatrix}.$$

$\mathbf{c}' = [0.0010 \quad -0.0016 \;\; 0.0096 \;\; 0.0304]^t, s_k = -241.1290.$

$\mathbf{x} = [16.9355 \;\; 12.9032 \;\; 25.9677 \;\; 0]^t.$

We remove the last element of $\mathbf{x}$ since it has become zero. Hence we remove the last column of A and the last element of $\mathbf{c}$. We then shrink the 3×4 matrix A and the 4×1 vector $\mathbf{c}$ to the 3×3 matrix and 3×1 vector and call them once again A and $\mathbf{c}$, respectively. The indexarray now becomes $[0\ 0\ 3\ 0\ 5]^t$ which means that the elements x_3 and x_5 have become nonbasic. Replace n by $n-1$, i.e., n is now 3.

S.5: We now go back to step S.3.

S.3: $H = A^+A$ is the unit matrix of order 3.
$\mathbf{c}' = (I - H)\mathbf{c} = \mathbf{0}$ (null column vector), s_k is not computable and hence the current solution vector $\mathbf{x}$ using the information of the indexarray is

$$\mathbf{x} = [16.9355 \;\; 12.9032 \;\; 0 \;\; 25.9677 \;\; 0]^t.$$

S.6: The value of the objective function $z = \mathbf{c}^t\mathbf{x} = [-1.2 \quad -1.4 \;\; 0 \;\; 0 \;\; 0]\mathbf{x} = -38.3871.$

Optimality test for the solution x: Here the basis (i.e., the original matrix A without columns 3 and 5)

$$B = \begin{bmatrix} 40 & 25 & 0 \\ 35 & 28 & 1 \\ 25 & 35 & 0 \end{bmatrix}, \quad \mathbf{c}_B^t = \begin{bmatrix} -1.2 \\ -1.4 \\ 0 \end{bmatrix}, \quad c_3 = 0, \mathbf{p}_3 = \text{column 3 of original}$$

$$A = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{p}_5 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

where $\mathbf{p}_j$ = column j of the original matrix A. Since B is nonsingular, $B^+ = B^{-1}$ is already available from step S.3.

S.7: $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [-0.009 \;\; 0 \quad -0.0335].$

S.8: Compute $z_3 - c_3 = \mathbf{y}^t\mathbf{p}_3 - c_3 = -0.009$ since $c_3 = 0$. Similarly,

$$z_5 - c_5 = \mathbf{y}^t\mathbf{p}_5 - c_5 = -0.0335.$$

S.9: All (here two) the $z_j - c_j$ values are negative, i.e., ≤ 0. Hence the DHALP has given us the optimal solution.

Example 2 (*Infeasible LPP where* $A\boldsymbol{x} = \boldsymbol{b}$ *is inconsistent*)

$$\min z = x_1 + x_2 + x_3 \;\text{ s.t.}$$
$$5x_1 + 3x_2 + 2x_3 = 10$$
$$2x_1 + x_2 + 2x_3 = 5$$

$$4x_1 + 2x_2 + 4x_3 = 4$$
$$x_1, x_2, x_3 \geq 0$$

S.1: Input $m = 3, n = 3, A = \begin{bmatrix} 5 & 3 & 2 \\ 2 & 1 & 2 \\ 4 & 2 & 4 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 10 \\ 5 \\ 4 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$.

S.1.a: indexarray $= [0\ \ 0\ \ 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [1.6340\ \ 1.2491\ \ -0.9585]^t, \mathbf{e} = A\mathbf{d} = [10\ \ 2.6\ \ 5.2]^t \neq \mathbf{b}$. Output 'The LPP is inconsistent.' Terminate.

Example 3 (*Infeasible LPP where* $A\mathbf{x} = \mathbf{b}$ *is consistent*)

$$\min z = -20x_1 - 30x_2 \text{ s.t.}$$
$$2x_1 + 3x_2 \geq 120$$
$$x_1 + x_2 \leq 40$$
$$2x_1 + 1.5x_2 \geq 90$$
$$x_1, x_2 \geq 0$$

S.1: Input $m = 3, n = 5, A = \begin{bmatrix} 2 & 3 & -1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 2 & 1.5 & 0 & 0 & -1 \end{bmatrix}$, $\mathbf{b} = [120\ \ 40\ \ 90]^t$, $\mathbf{c} = [-20\ \ -30\ \ 0\ \ 0\ \ 0]^t$.

S.1a: indexarray $= [0\ \ 0\ \ 0\ \ 0\ \ 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [22.9231\ \ 23.0769\ \ -4.9231\ \ -6.0000\ \ -9.5385]^t$. $\mathbf{e} = A\mathbf{d} = [120\ \ 40\ \ 90]^t = \mathbf{b}$. Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [1.0769\ \ -3.0769\ \ -7.0769\ \ 2.0000\ \ -2.4615]^t, s_k = -3$. $\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4\ \ x_5]^t = [26.1538\ \ 13.8462\ \ -26.1538\ \ 0\ \ -16.9231]^t$.

S.4: We remove x_4 since it has become zero. Hence we remove the fourth column vector of A and the fourth element of $\mathbf{c}$. Indexarray $= [0\ \ 0\ \ 0\ \ 4\ \ 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [16.9231\ \ 23.0769\ \ -16.9231\ \ -21.5385]^t.$$

S.5: We now go back to the step S.3.

S.3: The matrix A^+ is computed using the procedure PISM in $O(mn)$ operations since the current A^+ and A are known and can be used.

$$\mathbf{c}' = (I - H)\mathbf{c} = [3.0769\ \ -3.0769\ \ -3.0769\ \ 1.5385]^t,\ \ s_k = -14.0000.$$
$$\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_5]^t = [60\ \ -20\ \ -60\ \ 0]^t.$$

S.4: We remove x_5 since it has become zero. Hence we remove the last column vector of A and the last element of $\mathbf{c}$. Indexarray $= [0\ \ 0\ \ 0\ \ 4\ \ 5]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [60\ \ -20\ \ -60]^t.$$

S.5: We now go back to the step S.3.

S.3: $H = A^+A$ is the unit matrix of order 3. $\mathbf{c}' = (I - H)\mathbf{c} = [0\ \ 0\ \ 0]^t$.

Since $c_i' \leq 0$ for all i, we cannot compute s_k. The final solution is $\mathbf{x} = [x_1\ x_2\ x_3\ x_4\ x_5]^t = [60\ -20\ -60\ 0\ 0]^t$ that contains two negative elements, viz., $x_2 = -20$ and $x_3 = -60$, violating the nonnegativity condition. Hence the problem is infeasible.

Remark. It may be seen in the foregoing minimization (infeasible) problem that the value of the objective function increased; in a feasible case the value decreases for each successive removal of an element of $\mathbf{x}$.

Example 4 (*Infeasible LPP with* $A\mathbf{x} = \mathbf{b}$ *consistent*)

$$\min z = -3x_1 - 2x_2 \text{ s.t.}$$
$$2x_1 + x_2 \leq 2$$
$$3x_1 + 4x_2 \geq 12$$
$$x_1, x_2 \geq 0$$

S.1: Input $m = 2, n = 4, A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 3 & 4 & 0 & -1 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 2 \\ 12 \end{bmatrix}$, $\mathbf{c} = [-3\ -2\ 0\ 0]^t$.

S.1.a: Indexarray $= [0\ 0\ 0\ 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [0.3571\ 2.5000\ -1.2143\ -0.9286]^t$. $\mathbf{e} = A\mathbf{d} = [2\ 12]^t = \mathbf{b}$. Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.4643\ 0.2500\ 0.6786\ -0.3929]^t, s_k = -1.7895.$

$$\mathbf{x} = [x_1\ x_2\ x_3\ x_4]^t = [-0.4737\ 2.9474\ 0\ -1.6316]^t.$$

S.4: We remove x_3 since it has become zero. Hence we remove the third column vector of A and the third element of $\mathbf{c}$.
Indexarray $= [0\ 0\ 3\ 0]^t, \mathbf{d} = A^+\mathbf{b} = [-0.5333\ 3.0667\ -1.3333]^t$.

S.5: We now go back to step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [0.0333\ -0.0667\ -0.1667]^t$, $s_k = -16.0000$.

$$\mathbf{x} = [x_1\ x_2\ x_4]^t = [0\ 2\ -4]^t.$$

S.4: We remove x_1 since it has become zero. Hence we remove the first column vector of A and the first element of $\mathbf{c}$. indexarray $= [1\ 0\ 3\ 0]^t, \mathbf{d} = A^+\mathbf{b} = [2\ -4]^t$.

S.5: We now go back to step S.3.

S.3: $H = A^+A$ is now the unit matrix of order 2. $\mathbf{c}' = (I - H)\mathbf{c} = [0\ 0]^t$. Since $c_i' \leq 0$ for all i, we cannot compute s_k. The final solution is $\mathbf{x} = [x_1\ x_2\ x_3\ x_4]^t = [0\ 2\ 0\ -4]^t$ that contains one negative element, viz., $x_4 = -4$, violating the nonnegativity condition. Hence the problem is infeasible. As in the foregoing example, here also the value of the objective function increased.

Example 5 (*Degenerate solution*)

$$\min z = -3x_1 - 9x_2 \text{ s.t.}$$
$$x_1 + 4x_2 \leq 8$$
$$x_1 + 2x_2 \leq 4$$
$$x_1, x_2 \geq 0$$

S.1: Input $m = 2, n = 4, A = \begin{bmatrix} 1 & 4 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 8 \\ 4 \end{bmatrix}$, $\mathbf{c} = \begin{bmatrix} -3 \\ -9 \\ 0 \\ 0 \end{bmatrix}$.

S.1a: Indexarray $= [0\ 0\ 0\ 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [0.4444\ \ 1.7778\ \ 0.4444\ \ 0]^t \cdot \mathbf{e} = A\mathbf{d} = [8\ 4]^t = \mathbf{b}$. Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.3333\ \ -0.3333\ \ 1.6667\ \ 1.0000]^t$, $s_k = 0$.

$$\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4]^t = [0.4444\ \ 1.7778\ \ 0.4444\ 0]^t.$$

S.4: We now remove x_4 since it has become zero. Hence we remove the fourth column vector of A and fourth element of $\mathbf{c}$. Indexarray $= [0\ 0\ 0\ 4]^t$.
$\mathbf{d} = A^+\mathbf{b} = [0.4444\ \ 1.7778\ \ 0.4444]^t$.

S.5: We now go back to step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [0.6667\ \ -0.3333\ \ 0.6667]^t$, $s_k = 0.6667$.

$$\mathbf{x} = [x_1\ \ x_2\ \ x_3]^t = [0\ \ 2\ \ 0]^t.$$

S.4: Since $\mathbf{x}$ contains two zeros, we have two possibilities. (i) Keep x_1 in the basis and remove x_3 and (ii) keep x_3 in the basis and remove x_1.

(i) We remove x_3 from the basis. Thus indexarray $= [0\ \ 0\ \ 3\ \ 4]^t$. $\mathbf{d} = A^+\mathbf{b} = [0\ \ 2]^t$.

S.5: We now go back to step S.3.

S.3: $H = A^+A$ is the unit matrix of order 2. $\mathbf{c}' = (I - H)\mathbf{c} = [0\ 0]^t$. Since $c_i' \leq 0$ for all i, we cannot compute s_k.

S.6: Compute $\mathbf{z} = \mathbf{c}^t\mathbf{x} = (-3 \times 0) + (-9 \times 2) = -18$. Output

$$\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4]^t = [0\ \ 2\ \ 0\ \ 0]^t.$$

We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 1 & 4 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{c}_B = [-3\ -9]^t, \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{p}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad c_3 = 0, \quad c_4 = 0.$$

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [-1.5\ \ -1.5]^t$.

S.8: Compute $z_3 - c_3 = \mathbf{y}^t\mathbf{p}_3 - c_3 = -1.5$, and $z_5 - c_5 = \mathbf{y}^t\mathbf{p}_5 - c_5 = -1.5$.

S.9: As all the $z_j - c_j$ values are negative, the DHALP has given us the optimal solution to the degenerate problem with the basic variable $x_1 = 0$. (ii) We remove x_1 from the basis. Indexarray $= [1\ \ 0\ \ 0\ \ 4]^t$, $\mathbf{d} = A^+\mathbf{b} = [2\ \ 0]^t$.

S.5: We now go back to step S.3.

S.3: $H = A^+A$ is the unit matrix of order 2. $\mathbf{c}' = (I - H)\mathbf{c} = [0\ \ 0]^t$. Since $c_i' \leq 0$ for all i, we cannot compute s_k.

S.6: Output $z = \mathbf{c}^t\mathbf{x} = (-9 \times 2) + (0 \times 0) = -18$ and

$$\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4]^t = [0\ \ 2\ \ 0\ \ 0]^t.$$

We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 4 & 1 \\ 2 & 0 \end{bmatrix}, \quad \mathbf{c}_B = [-9\ \ 0]^t, \quad \mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{p}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad c_1 = -3, \quad c_4 = 0.$$

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [0 \ -4.5]^t$.

S.8: Compute $z_1 - c_1 = \mathbf{y}^t\mathbf{p}_1 - c_1 = -1.5$, and $z_4 - c_4 = \mathbf{y}^t\mathbf{p}_4 - c_4 = -4.5$.

S.9: As all the $z_j - c_j$ values are negative, the DHALP has given us the optimal solution to the degenerate problem with the basic variable $x_3 = 0$.

Optimal solution $\mathbf{x} = [0 \ 2 \ 0 \ 0]^t, z = -18$.

Example 6 (*Infinity of solutions*)

$$\min z = -2x_1 - 4x_2 \text{ s.t.}$$
$$x_1 + 2x_2 \leq 5$$
$$x_1 + x_2 \leq 4$$
$$x_1, x_2 \geq 0$$

S.1: Input $m = 2, n = 4, A = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$, $\mathbf{c} = [-2 \ -4 \ 0 \ 0]^t$.

S.1a: Indexarray $= [0 \ 0 \ 0 \ 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [1.3333 \ 1.6667 \ 0.3333 \ 1.0000]^t$. $\mathbf{e} = A\mathbf{d} = [5 \ 4]^t = \mathbf{b}$. Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [0.0000 \ -0.6667 \ 1.3333 \ 0.6667]^t, s_k = 0.25$.

$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4]^t = [1.3333 \ 1.8333 \ 0.0000 \ 0.8333]^t.$$

S.4: We remove x_3 since it has become zero. Hence we remove the third column vector of A and third element of $\mathbf{c}$. Indexarray $= [0 \ 0 \ 3 \ 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [1.3333 \ 1.8333 \ 0.8333]^t.$$

S.5: We now go back to the step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [0 \ 0 \ 0]^t$. Since $c_i' \leq 0$ for all i, s_k is not computable.

S.6: Compute $z = \mathbf{c}^t\mathbf{x} = (-2 \times 1.3333) + (-4 \times 1.8333) + (0 \times 0.8333) = -10$. Output $\mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4]^t = [1.3333 \ 1.8333 \ 0.0000 \ 0.8333]^t$, $z = -10$. We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{c}_B = \begin{bmatrix} -2 \\ -4 \\ 0 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad c_3 = 0.$$

When we have the number of equations less than the number of variables and the solution is bounded, we append additional rows to the rectangular basis matrix B so that the resulting B is nonsingular (square). The corresponding righthand side value of the new element of $\mathbf{b}$ is then computed using the solution vector $\mathbf{x}$. The columns $\mathbf{p}_j$ (of A) corresponding to the nonbasic variables x_j are appended by zeros so that their dimension is compatible with the order of the basis B (i.e., the number of elements in $\mathbf{p}_j$ = the order of B). This appendage (by zeros) is equivalent to appending null rows to A and corresponding zero elements to $\mathbf{b}$. The optimality test is then performed.

The basis matrix $B = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ with the appended row $[0 \ 0 \ 1]$ and the nonbasic vector $\mathbf{p}_3$ with one appended zero become

$$B = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [-2 \;\; 4 \;\; 0]B^{-1} = [-2 \;\; 0 \;\; 0]$.

S.8: Compute $z_3 - c_3 = \mathbf{y}^t \mathbf{p}_3 - c_3 = [-2 \;\; 0 \;\; 0]\mathbf{p}_3 - 0 = -2$.

S.9: As $z_3 - c_3 \leq 0$ corresponding to the only nonbasic variable x_3, the DHALP has given us the optimal solution.

Example 7 (*Unbounded case*)

$$\begin{aligned} &\min z = -100x_1 - 800x_2 \;\text{ s.t.} \\ &6x_1 + 2x_2 \geq 12 \\ &2x_1 + 2x_2 \geq 8 \\ &4x_1 + 12x_2 \geq 24 \\ &x_1, x_2 \geq 0 \end{aligned}$$

S.1: Input $m = 3, n = 5$,

$$A = \begin{bmatrix} 6 & 2 & -1 & 0 & 0 \\ 2 & 2 & 0 & -1 & 0 \\ 4 & 12 & 0 & 0 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 12 \\ 8 \\ 24 \end{bmatrix}, \quad \mathbf{c} = [-100 \;\; -800 \; 0 \; 0 \; 0]^t.$$

S.1a: Indexarray $= [0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [1.5481 \;\; 1.4962 \;\; 0.2811 \;\; -1.9114 \;\; 0.1470]^t$.
$\mathbf{e} = A\mathbf{d} = [12 \;\; 8 \;\; 24]^t = \mathbf{b}$. Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [7.7622 \;\; -8.4757 \;\; 29.6216 \;\; -1.4270 \;\; -70.6595]^t$, $s_k = 0.0095$.

$$\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5]^t = [1.4745 \;\; 1.5766 \;\; 0 \;\; -1.8978 \;\; 0.8175]^t.$$

S.4: We remove x_3 since it has become zero. Hence we remove the third column vector of A and the third element of $\mathbf{c}$. Indexarray $= [0 \;\; 0 \;\; 3 \;\; 0 \;\; 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [1.4952 \;\; 1.5143 \;\; -1.9810 \;\; 0.1524]^t.$$

S.5: We now go back to the step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [2.1905 \;\; -6.5714 \;\; -8.7619 \;\; -70.0952]^t$, $s_k = 0.6826$.

$$\mathbf{x} = [x_1 \;\; x_2 \;\; x_4 \;\; x_5]^t = [0.0000 \;\; 6.0000 \;\; 4.0000 \;\; 48.0000]^t.$$

S.4: We remove x_1 since it has become zero. Hence we remove the first column vector of A and the first element of $\mathbf{c}$. indexarray $= [1 \;\; 0 \;\; 3 \;\; 0 \;\; 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [6.0000 \;\; 4.0000 \;\; 48.0000]^t.$$

S.5: We now go back to step S.3.

S.3: $H = A^+A$ is the unit matrix of order 3. Sincc $c_i' \leq 0$ for all i, we cannot compute s_k.

S.6: Compute $z = \mathbf{c}^t\mathbf{x} = (-800 \times 6) + (0 \times 4) + (0 \times 48) = -4800$.

Output $\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5]^t = [0 \;\; 6 \;\; 0 \;\; 4 \;\; 48]^t, z = -4800$.

We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 2 & 0 & 0 \\ 2 & -1 & 0 \\ 12 & 0 & -1 \end{bmatrix}, \quad \mathbf{c}_B = \begin{bmatrix} -800 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix} 6 \\ 2 \\ 4 \end{bmatrix},$$
$$\mathbf{p}_3 = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \quad c_1 = -100, c_3 = 0.$$

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [-400 \ 0 \ 0]^t$.

S.8: Compute $z_1 - c_1 = \mathbf{y}^t\mathbf{p}_1 - c_1 = -2400$ and $z_3 - c_3 = \mathbf{y}^t\mathbf{p}_3 - c_3 = 400$.

S.9: Since $z_3 - c_3 > 0$, the optimal solution is not reached. Either the solution is unbounded or it is not optimal. We use the revised SA to proceed further, starting with the output solution $\mathbf{x}$ of the DHALP algorithm as the initial basic feasible solution.

Revised SA: From the DHALP, the nonbasic variables are x_1 and x_3.

S.10: *Determining the entering vector* $\mathbf{p}_j$: Compute $z_j - c_j$ for nonbasic $\mathbf{p}_1$ and $\mathbf{p}_3$. From the step S.8,

$z_1 - c_1 = \mathbf{y}^t\,\mathbf{p}_1 - c_1 = -2400$ and $z_3 - c_3 = \mathbf{y}^t\mathbf{p}_3 - c_3 = 400$. Since $z_3 - c_3$ is positive, x_3 now becomes a basic variable and $\mathbf{p}_3$ enters the basis.

S.11: *Determining the leaving vector* $\mathbf{p}_j$

(i) The current basic solution is

$$\mathbf{x}_B = \begin{bmatrix} x_2 \\ x_4 \\ x_5 \end{bmatrix} = B^{-1}\mathbf{b} = \begin{bmatrix} 0.5 & 0 & 0 \\ 1 & -1 & 0 \\ 6 & 0 & -1 \end{bmatrix}\begin{bmatrix} 12 \\ 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \\ 48 \end{bmatrix}.$$

(ii) The constraint coefficients of the entering variables, i.e.,

$$\alpha^{(3)} = B^{-1}\mathbf{p}_3 = \begin{bmatrix} 0.5 & 0 & 0 \\ 1 & -1 & 0 \\ 6 & 0 & -1 \end{bmatrix}\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1 \\ -6 \end{bmatrix}.$$
$$\theta = \min_k \left\{ \frac{(B^{-1}\mathbf{b})_k}{\alpha_k^{(3)}}, \alpha_k^{(3)} > 0 \right\}$$

where $(B^{-1}\mathbf{b})_k$ and $\alpha_k^{(3)}$ are the kth element of $B^{-1}\mathbf{b}$ and $\boldsymbol{\alpha}^{(3)}$. Since all the $\alpha_k^{(3)} \leq 0$, the problem has no bounded solution.

Example 8 (*Cycling in SA*). The LPP [3, 1] is

$$\begin{aligned} \min z = &-0.75x_4 + 20x_5 - 0.5x_6 + 6x_7 \ \text{ s.t.} \\ &x_1 + 0.25x_4 - 8x_5 - x_6 + 9x_7 = 0 \\ &x_2 + 0.5x_4 - 12x_5 - 0.5x_6 + 3x_7 = 0 \\ &x_3 + x_6 = 1 \\ &x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \end{aligned}$$

S.1: Input $m = 3, n = 7$,

$$A = \begin{bmatrix} 1.0000 & 0 & 0 & 0.2500 & -8.0000 & -1.0000 & 9.0000 \\ 0 & 1.0000 & 0 & 0.5000 & -12.0000 & -0.5000 & 3.0000 \\ 0 & 0 & 1.0000 & 0 & 0 & 1.0000 & 0 \end{bmatrix},$$

$$\mathbf{b} = [0\ \ 0\ \ 1]^t, \quad \mathbf{c} = [0\ \ 0\ \ 0\ \ -0.75\ \ 20\ \ -0.5\ \ 6]^t.$$

S.1a: Indexarray $= [0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [0.0063\ \ -0.0034\ \ 0.5023\ \ -0.0001\ \ -0.0095\ \ 0.4977\ \ 0.0462]^t$. $\mathbf{e} = A\mathbf{d} = [0\ \ 0\ \ 1]^t = \mathbf{b}$. Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-1.4946\ \ 2.6341\ \ 0.1612\ \ 0.1934\ \ 0.3471\ \ -0.1612\ \ 0.4513]^t$, $s_k = -0.0273$, $\mathbf{x} = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7]^t = [-0.0346\ 0.0686\ 0.5067\ 0.0052\ 0\ 0.4933\ 0.0585]^t$.

S.4: We remove x_5 since it has become zero. Hence we remove the fifth column vector of A and the fifth element of $\mathbf{c}$. indexarray $= [0\ \ 0\ \ 0\ \ 0\ \ 5\ \ 0\ \ 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [-0.0155\ \ 0.0649\ \ 0.5085\ \ 0.0286\ \ 0.4915\ \ 0.0555]^t.$$

S.5: We now go back to step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.6996\ 0.1350\ \ -0.0660\ \ -0.8570\ 0.0660\ 0.1089]^t, s_k = 0.4803.$

$$\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4\ \ x_6\ \ x_7]^t = [-0.3206\ \ 0\ \ 0.5402\ \ 0.4404\ \ 0.4598\ \ 0.0032]^t.$$

S.4: We remove x_2 since it has become zero. Hence we remove the second column vector of A and the second element of $\mathbf{c}$. Indexarray $= [0\ 2\ 0\ 0\ 5\ 0\ 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [-0.0875\ \ 0.5268\ \ 0.1193\ \ 0.4732\ \ 0.0590]^t.$$

S.5: We now go back to the step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.8497\ \ -0.0279\ \ -0.6686\ \ 0.0279\ \ 0.1161]^t$, $s_k = 0.5082$.

$$\mathbf{x} = [x_1\ \ x_3\ \ x_4\ \ x_6\ \ x_7]^t = [0.3443\ \ 0.5410\ \ 0.4590\ \ 0.4590\ \ 0]^t.$$

S.4: We remove x_7 since it has become zero. Hence we remove the last column vector of A and the last element of $\mathbf{c}$. indexarray $= [0\ \ 2\ \ 0\ \ 0\ \ 5\ \ 0\ \ 7]^t$.

$$\mathbf{d} = A^+\,\mathbf{d} = [0.2105\ \ 0.7193\ \ 0.2807\ \ 0.2807]^t.$$

S.5: We now go back to the step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.2632\ 0.3509\ \ -0.3509\ \ -0.3509]^t$, $s_k = 2.05$.

$$\mathbf{x} = [x_1\ \ x_3\ \ x_4\ \ x_6]^t = [0.75\ \ 0\ \ 1\ \ 1]^t.$$

S.4: We remove x_3 since it has become zero. Hence we remove the second column vector of A and the second element of $\mathbf{c}$. indexarray $= [0\ \ 2\ \ 3\ \ 0\ \ 5\ \ 0\ \ 7]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [0.75\ \ 1\ \ 1]^t.$$

S.5: We now go back to step S.3.

S.3: $H = A^+A$ is the unit matrix of order 3. $\mathbf{c}' = [0\ \ 0\ \ 0]^t$. Since $c_i' \leq 0$ for all i, we cannot compute s_k.

S.6: Compute $z = \mathbf{c}^t\mathbf{x} = (0 \times 0.75) + (-0.75 \times 1) + (-0.5 \times 1) = -1.25$.
Output $\mathbf{x} = [0.75\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0]^t,\ \ z = -1.25$.
We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 1 & 0.25 & -1 \\ 0 & 0.5 & -0.5 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{c}_B = \begin{bmatrix} 0 \\ -0.75 \\ -0.5 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix},$$

$$\mathbf{p}_5 = \begin{bmatrix} -8 \\ -12 \\ 0 \end{bmatrix}, \quad \mathbf{p}_7 = \begin{bmatrix} 9 \\ 3 \\ 0 \end{bmatrix}, \quad c_2 = 0, \quad c_3 = 0, \quad c_5 = 20, \quad c_7 = 6.$$

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [0\ \ -1.5\ \ -1.25]$.

Note: Since B is nonsingular, $B^+ = B^{-1}$ is already available from step S.3.
S.8: Compute $z_2 - c_2 = \mathbf{y}^t\mathbf{p}_2 - c_2 = -1.5, z_3 - c_3 = \mathbf{y}^t\mathbf{p}_3 - c_3 = -1.25$. Similarly $z_5 - c_5 = -2$ and $z_7 - c_7 = -10.5$.

S.9: Since all the values of $z_j - c_j \leq 0$, the optimal solution is reached and is given by $\mathbf{x}_B = [x_1\ \ x_4\ \ x_6]^t = B^{-1}\mathbf{b} = [0.75\ \ 1\ \ 1]^t$ or $\mathbf{x} = [0.75\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0]^t$ and $z_B = \mathbf{c}_B^t\mathbf{x}_B = (0 \times 0.75) + (-0.75 \times 1) + (-0.5 \times 1) = -1.25$.

Example 9 (*Two x_i becoming zero simultaneously*)

$$\begin{aligned} &\min z = 2x_1 + 7x_2 - 2x_3 \ \text{ s.t.} \\ &x_1 + 2x_2 + x_3 \leq 1 \\ &-4x_1 - 2x_2 + 3x_3 \leq 2 \\ &x_1, x_2, x_3 \geq 0 \end{aligned}$$

S.1: Input

$$m = 2,\ \ n = 5,\ \ A = \begin{bmatrix} 1 & 2 & 1 & 1 & 0 \\ -4 & -2 & 3 & 0 & 1 \end{bmatrix},\ \ \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix},\ \ \mathbf{c} = [2\ \ 7\ \ -2\ \ 0\ \ 0]^t.$$

S.1a: Indexarray $= [0\ \ 0\ \ 0\ \ 0\ \ 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [-0.1946\ \ 0.2270\ \ 0.5243\ \ 0.2162\ \ 0.1027]^t, \mathbf{e} = A\mathbf{d} = [1\ \ 2]^t = \mathbf{b}$.
Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-2.2378\ \ 2.6108\ \ -1.4703\ \ -1.5135\ \ 0.6811]^t,\ \ s_k = 0.0870$.

$$\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4\ \ x_5]^t = [0\ \ 0\ \ 0.6522\ \ 0.3478\ \ 0.0435]^t.$$

S.4: We now solve the problem removing (i) x_1 only from the basis as well as (ii) keeping x_1 in the basis and removing x_2 from the basis.

Case (i). We remove x_1 from the basis. Hence the first column vector of A as well as the first element of $\mathbf{c}$ are removed.
Indexarray $= [1\ \ 0\ \ 0\ \ 0\ \ 0]^t, \mathbf{d} = A^+\mathbf{b} = [0.0723\ \ 0.6627\ \ 0.1928\ \ 0.1566]^t$.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [0.8313\ \ 0.1205\ \ -1.7831\ \ 1.3012]^t,\ \ s_k = 0.0870$.

$$\mathbf{x} = [x_2\ \ x_3\ \ x_4\ \ x_5]^t = [0\ \ 0.6522\ \ 0.3478\ \ 0.0435]^t.$$

S.4: We remove x_2 since it has become zero. Hence we remove the first column vector of A and the first element of $\mathbf{c}$. Indexarray $= [1\ \ 2\ \ 0\ \ 0\ \ 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [0.6364\ \ 0.3636\ \ 0.0909]^t.$$

S.5: We now go back to step S3,

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.1818\ \ 0.1818\ \ 0.5455]^t,\ \ s_k = 0.1667.$

$$\mathbf{x} = [x_3\ \ x_4\ \ x_5]^t = [0.6667\ \ 0.3333\ \ 0]^t.$$

S.4: We remove x_5 since it has become zero. Hence we remove the last column vector of A and the last element of $\mathbf{c}$. indexarray $= [1\ \ 2\ \ 0\ \ 0\ \ 5]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [0.6667\ \ 0.3333]^t.$$

S.5: We now go back to step S3.

S.3: $H = A^+A$ is the unit matrix of order 2. $\mathbf{c}' = (I - H)\mathbf{c} = [0\ \ 0]^t$. Since $c_i' \leq 0$ for all i, we cannot compute s_k.

S.6: Compute $z = \mathbf{c}^t\mathbf{x} = (-2 \times 0.6667) + (0 \times 0.3333) = -1.3333$.
Output $\mathbf{x} = [x_1\ \ x_2\ \ x_3\ \ x_4\ \ x_5]^t = [0\ \ 0\ \ 0.6667\ \ 0.3333\ \ 0]^t,\ \ z = -1.3333$.
We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 1 & 1 \\ 3 & 0 \end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix} 1 \\ -4 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \quad \mathbf{p}_5 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$\mathbf{c}_B^t = [-2\ \ 0], \quad c_1 = 2, \quad c_2 = 7, \quad c_5 = 0$$

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [0\ \ 0.6667]$.

Note: Since B is nonsingular, $B^+ = B^{-1}$ is already available from step S.3.

Compute $z_1 - c_1 = \mathbf{y}^t\mathbf{p}_1 - c_1 = [0\ \ -0.6667]\begin{bmatrix} 1 \\ -4 \end{bmatrix} - 2 = 0.6667.$

Similarly, $z_2 - c_2 = -5.6667$ and $z_5 - c_5 = -0.6667$. Since $z_1 - c_1 \geq 0$, the solution is not optimal.

Case (ii). Unlike case (i), here we remove x_2 keeping x_1 in the basis.

Indexarray $= [0\ \ 2\ \ 0\ \ 0\ \ 0]^t, \mathbf{d} = A^+\mathbf{b} = [0\ \ 0.6364\ \ 0.3636\ \ 0.0909]^t$.

S.5: We now go back to step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [0.0000\ \ -0.1818\ \ 0.1818\ \ 0.5455]^t,\ \ s_k = 0.1667.$

$$\mathbf{x} = [x_1\ \ x_3\ \ x_4\ \ x_5]^t = [0\ \ 0.6667\ \ 0.3333\ \ 0]^t.$$

Again we are confronted to a situation where there are two zeros in $\mathbf{x}$. We apply the same procedure as above, i.e., (a) keep x_5 in the basis and remove x_1 and (b) keep x_1 in the basis and remove x_5.

Subcase (a): We remove x_1 from the basis. Hence the first column vector of A as well as the first element of $\mathbf{c}$ are removed. indexarray $= [1\ \ 2\ \ 0\ \ 0\ \ 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [0.6364\ \ 0.3636\ \ 0.0909]^t.$$

S.5: We now go back to the step S3.

S.3: $\mathbf{c}' - (I - H)\mathbf{c} = [-0.1818 \;\; 0.1818 \;\; 0.5455]^t$, $s_k = 0.1667$.

$\mathbf{x} = [x_3 \;\; x_4 \;\; x_5]^t = [0.6667 \;\; 0.3333 \;\; 0]^t$.

S.4: We remove x_5 since it has become zero. Hence we remove the last column vector of A and the last element of $\mathbf{c}$. indexarray $= [1 \;\; 2 \;\; 0 \;\; 0 \;\; 5]^t$.

After this step, the calculation proceeds in the same way as in case (i).

Subcase (b): Unlike subcase (a), here we remove x_5, keeping x_1 in the basis.

Indexarray $= [0 \;\; 2 \;\; 0 \;\; 0 \;\; 5]^t$, $\mathbf{d} = A^+\mathbf{b} = [-0.0135 \;\; 0.6486 \;\; 0.3649]^t$.

S.5: We now go back to the step S3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.0811 \;\; -0.1081 \;\; 0.1892]^t$, $s_k = 1.9286$.

$\mathbf{x} = [x_1 \;\; x_3 \;\; x_4]^t = [0.1429 \;\; 0.8571 \;\; 0]^t$.

S.4: We remove x_4 since it has become zero. Hence we remove the last column vector of A and the last element of $\mathbf{c}$. indexarray $= [0 \;\; 2 \;\; 0 \;\; 4 \;\; 5]^t$.

$\mathbf{d} = A^+\mathbf{b} = [0.1429 \;\; 0.8571]^t$.

S.5: We now go back to the step S3.

S.3: $H = A^+A$ is the unit matrix of order 2. $\mathbf{c}' = (I - H)\mathbf{c} = [0 \;\; 0]^t$. Since $c_i' \leq 0$ for all i, we cannot compute s_k.

S.6: Compute $z = \mathbf{c}^t\mathbf{x} = (2 \times 0.1429) + (-2 \times 0.8571) = -1.4286$.

Output $\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5]^t = [0.1429 \;\; 0 \;\; 0.8571 \;\; 0 \;\; 0]^t$, $z = -1.4286$.

We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 1 & 1 \\ -4 & 3 \end{bmatrix}, \quad \mathbf{p}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \quad \mathbf{p}_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{p}_5 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$\mathbf{c}_B^t = [2 \;\; -2], \quad c_2 = 7, \quad c_4 = 0, \quad c_5 = 0.$$

Note: Since B is nonsingular, $B^+ = B^{-1}$ is already available from step S.3.

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [-0.2857 \;\; -0.5714]$.

S.8: Compute $z_2 - c_2 = \mathbf{y}^t\mathbf{p}_2 - c_2 = -6.4286$, $z_4 - c_4 = \mathbf{y}^t\mathbf{p}_4 - c_4 = -0.2857$ and $z_5 - c_5 = \mathbf{y}^t\mathbf{p}_5 - c_5 = -0.5714$. Since all the values of $z_j - c_j \leq 0$, the optimal solution is reached. Therefore, the required optimal solution is $\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_4 \;\; x_5]^t = [0.1429 \;\; 0 \;\; 0.8571 \;\; 0 \;\; 0]^t$ and the objective function value is $z = -1.4286$.

Example 10 (*Optimal solution not reached – DHALP failed*)

$$\min z = -3x_1 - 2x_2 \text{ s.t.}$$
$$2x_1 + x_2 \leq 2$$
$$3x_1 + 4x_2 \geq 2$$
$$x_1, x_2 \geq 0.$$

S.1: Input $m = 2, n = 4, A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 3 & 4 & 0 & -1 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $\mathbf{c} = [-3 \;\; -2 \;\; 0 \;\; 0]^t$.

S.1a: Indexarray $= [0 \;\; 0 \;\; 0 \;\; 0]^t$.

S.2: $\mathbf{d} = A^+\mathbf{b} = [0.7143 \;\; 0.0000 \;\; 0.5714 \;\; 0.1429]^t$, $\mathbf{e} = A\mathbf{d} = [2 \;\; 2]^t = \mathbf{b}$. Hence the equation $A\mathbf{x} = \mathbf{b}$ is consistent.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.4643 \;\; 0.2500 \;\; 0.6786 \;\; -0.3929]^t, \;\; s_k = 0.$

$$\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^t = [0.7143 \;\; 0 \;\; 0.5714 \;\; 0.1429]^t.$$

S.4: We remove x_2 since it has become zero. Hence we remove the second column vector of A and the second element of $\mathbf{c}$. indexarray $= [0 \;\; 2 \;\; 0 \;\; 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [0.7143 \;\; 0.5714 \;\; 0.1429]^t.$$

S.5: We now go back to step S.3.

S.3: $\mathbf{c}' = (I - H)\mathbf{c} = [-0.2143 \;\; 0.4286 \;\; -0.6429]^t, \;\; s_k = 1.3333.$

$$\mathbf{x} = [x_1 \;\; x_3 \;\; x_4]^t = [1.0000 \;\; 0.0000 \;\; 1.0000]^t.$$

S.4: We remove x_3 since it has become zero. Hence we remove the second column vector of A and the second element of $\mathbf{c}$. indexarray $= [0 \;\; 2 \;\; 3 \;\; 0]^t$.

$$\mathbf{d} = A^+\mathbf{b} = [1 \;\; 1]^t.$$

S.5: We now go back to step S.3.

S.3: $H = A^+A$ is the unit matrix of order 2. $\mathbf{c}' = (I - H)\mathbf{c} = [0 \;\; 0]^t$. Since $c_i \leq 0$ for all i, we cannot compute s_k.

S.6: Compute $z = \mathbf{c}^t\mathbf{x} = (-3 \times 1) + (0 \times 1) = -3$.
Output $\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^t = [1 \;\; 0 \;\; 0 \;\; 1]^t, \;\; z = -3$.
We subject the solution (basic) to the optimality test. Here

$$B = \begin{bmatrix} 2 & 0 \\ 3 & -1 \end{bmatrix}, \;\; \mathbf{p}_2 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \;\; \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \;\; \mathbf{c}_B^t = [-3 \quad 0], \;\; c_2 = -2, \;\; c_3 = 0.$$

S.7: Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [-1.5 \;\; 0]$.

S.8: Compute $z_2 - c_2 = \mathbf{y}^t\mathbf{p}_2 - c_2 = 0.5, \;\; z_3 - c_3 = \mathbf{y}^t\mathbf{p}_3 - c_3 = -1.5$.

Since $z_2 - c_2 > 0$, the solution is not optimal: either the solution is unbounded or the DHALP could not reach the optimal solution. We use the revised SA to proceed further, starting with the output solution $\mathbf{x}$ of the DHALP algorithm as the initial basic feasible solution.

Revised S.4: From the DHALP, the nonbasic variables are x_2 and x_3.

Step A. Determining the entering vector, $\mathbf{p}_j$
Compute $z_j - c_j$ for nonbasic $\mathbf{p}_2$ and $\mathbf{p}_3$. From the step S.8, $z_2 - c_2 = 0.5$ and $z_3 - c_3 = -1.5$. Since $z_2 - c_2$ is positive, x_2 now becomes a basic variable and $\mathbf{p}_2$ enters the basis.

Step B. Determining the leaving vector $\mathbf{p}_j$
The current basic solution is

(i) $$\mathbf{x}_B = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = B^{-1}\mathbf{b} = \begin{bmatrix} 0.5 & 0 \\ 1.5 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

(ii) The constraint coefficients of the entering variables, i.e.,

$$\alpha^{(2)} = B^{-1}\mathbf{p}_2 = [0.5 - 2.5]^t.$$

$\theta = \min\{1/0.5, -\} = 2$. Thus x_1 leaves the basis. We go back to the step A.

Step A.

$$B = \begin{bmatrix} 1 & 0 \\ 4 & -1 \end{bmatrix}, \quad \mathbf{p}_1 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$
$$\mathbf{c}_B^t = [-2 \;\; 0], \quad c_1 = -3, \quad c_3 = 0.$$

Compute $\mathbf{y}^t = \mathbf{c}_B^t B^{-1} = [-2 \;\; 0]$. Compute $z_j - c_j$ for nonbasic $\mathbf{p}_1$ and $\mathbf{p}_3$.

$$z_1 - c_1 = \mathbf{y}^t \mathbf{p}_1 - c_1 = -1 \text{ and } z_3 - c_3 = -2.$$

Since all the values of $z_j - c_j \leq 0$, the optimal solution has been reached and is given by $\mathbf{x}_B = [x_2 \;\; x_4]^t = B^{-1}\mathbf{b} = [2 \;\; 6]^t$ or $\mathbf{x} = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^t = [0 \;\; 2 \;\; 0 \;\; 6]^t$ and $z_B = \mathbf{c}_B^t \mathbf{x}_B = (-2 \times 2) + (0 \times 6) = -4$.

4. Conclusions

Versus simplex algorithm: The direct heuristic algorithm (DHALP) and the most popular and most widely used simplex method (SA) have the following differences.

If the SA does not enter into an infinite loop [1, 3, 40], i.e. cycling – a situation mostly not encountered in practice – then it will certainly provide the required solution of the LPP. The DHALP may not certainly provide the solution although in most (over 95% of the problems solved by us) problems it does. Even if it does not provide the optimal solution, it will usually provide one close to the optimal one. One may obtain the optimal one from this solution using the revised SA in a fewer steps.

The bottom-most row, viz., the $-c_j$ row in the SA tells us whether the optimal solution is reached or not while, in the DHALP, the optimality test based on using the coefficients of nonbasic variables checks the optimality.

Each next-tableau of the SA is computed by the elementary row/column operations similar to those in Gauss reduction method for solving linear systems. Each tableau needs almost the same amount of computation. The coefficient matrix A in the DHALP looses in each step one column corresponding to the variable x_i whose value becomes zero resulting in successive reduction in computations. In addition, a minimum norm least squares inverse (p-inverse) of the shrunk matrix A is calculated from the current A^+ and the current A in $O(mn)$ operations and not from the current A alone in $O(mn^2)$ operations.

A variable in the SA may enter into the basis and may go out; this may happen a number of times. In the DHALP, once a variable that leaves the basis will never enter into the basis.

The SA is exponential time (in the worst case) while the DHALP is polynomial-time direct and needs $O(mn^2)$ operations like the algorithms for solving linear systems using, for example, the Gauss reduction method.

While the SA is iterative, and the precise amount of computation in it is not known *a priori*, the DHALP is noniterative and the precise amount of computation in it is almost always known beforehand.

Artificial variables for 'equal to' and 'greater than or equal to' constraints are needed in the SA (besides surplus/slack variables) for consistency check. No artificial variable is needed in the DHALP – consistency check is in-built.

Initial basic feasible solution is known in SA while it is unknown in the DHALP. *Obtaining a (basic feasible) nonnegative solution of linear systems*: If the DHALP fails to provide an optimal solution for an LPP, it will almost always provide at least a nonnegative solution, i.e., a point inside the polytope defined by $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ – which is

a very good basic feasible solution from which optimal solution could be obtained within a few iterations of the revised SA.

Versus interior-point methods: Several interior-point methods [13, 12, 36, 19] have been proposed during the last two decades for the LPP. All these methods find a point (a basic feasible solution) inside the polytope $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ and then proceed in the direction of the **c**-vector in search of the corner (of the polytope) that represents the optimal solution. In the SA, however, we go along a hyperplane to a corner until an optimal corner is found. All these interior-point methods and the SA are iterative. The DHALP attempts to go inside the polytope $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ and then tries to hit upon the optimal solution mathematically noniteratively.

Versus other algorithms: The inequality sorting algorithm [15, 17] and the Barnes algorithm for detection of basic variables [2, 33] are distinctly different from the DHALP in that these are iterative and deterministic.

Error-free computation with DHALP. The DHALP involves only the basic arithmetic operations, viz., add, subtract, multiply, and divide operations. It does not need square-rooting or other operations that produce irrational numbers. A bound [9, 37–39] on the solution vector **x** for the LPP can be easily deduced. So the error-free computation using multiple-modulus residue arithmetic or p-adic arithmetic can be implemented on the DHALP. This implementation will appear elsewhere. Most interior-point methods are not amenable to error-free computations as they involve square-rooting operations.

Degenerate, unbounded, infeasible LPPs: We have considered such LPPs in the numerical examples (§ 3) and necessary discussions are included there. However, the DHALP has performed well in all these LPPs.

Small problems: We have attempted to solve small problems in which the constraint matrix A has order not greater than 20. This is mainly because of the limitation of the available computing resources including the limitation of the accuracy of computation in a PC-MATLAB at our disposal. However, a double- or multiple-precision high-level language may be used to solve reasonably large problems with a fair amount of accuracy using the DHALP. We will attempt encoding the DHALP into a high-level program (which is a significant computational problem) for large sparse LPPs in a future article.

Open problem: The problem of finding the necessary and sufficient condition under which the DHALP will produce an optimal solution of the LPP (or, equivalently, the DHALP becomes a direct algorithm) is open. A visualization of the problem geometrically may, however, provide a meaningful sufficient condition without much difficulty. Such a condition could be of some practical use but the necessary and sufficient condition, if found, will be of significant practical importance.

Acknowledgment

The authors wish to thank the authority of the University of Mauritius for providing necessary computing and office facilities for this research while the second author thanks the Tertiary Education Commission for a scholarship.

References

[1] Bazaraa M S, Jarvis J J and Sheraldi H D, Linear Programming and Network Flows, 2nd ed. (Singapore: John Wiley and sons, Inc.) (1990) pp. 165–167

[2] Barnes E R, A variation of Karmarkar's algorithm for solving linear programming problems, *Math. Prog.* **36** (1986) 174–182

[3] Beale E M L, Cycling in the dual simplex algorithm. *Naval Research Logistics Quarterly* **2** (1955) 269–275
[4] Ben Israel A and Greville T N E, Generalized Inverses: Theory and Applications (New York: Wiley) (1974)
[5] Campbell S L, On the continuity of the Moore-Penrose and Drazkin generalized inverses, *Linear algebra and its applications* **18** (1977) 53–57
[6] Dantzig G, Linear Programming and Extensions (New Jersey: Princeton University Press, Princeton) (1963)
[7] Dongarra J J, Du Croz J J, Hammerling S J and Hanson R J, An extended set of Fortran basis linear algebra subprograms, *ACM Trans. Math. Software* **14(1)** (1988) 1–17
[8] Golub G and Kahan W, Calculating the singular values and the pseudoinverse of a matrix, *SIAM J. Numer. Anal*. **B-2** (1965) 205–224
[9] Gregory R T and Krishnamurthy E V, Methods and Applications of Error-free Computation (New York: Springer Verlag) (1984)
[10] Greville T N E, The pseudoinverse of a rectangular or singular matrix and its application to the solution of systems of linear equations, *SIAM Rev.* **1** (1959) 38–43
[11] Hooker J N, Karmarkar's Linear Programming Algorithm. *Interfaces* **16(4)** (1986) 75–90
[12] Karmarkar N, A new polynomial-time algorithm for linear programming, *Tech. Rep* (New Jersey: AT&T Bell Labs.) (1984)
[13] Khachiyan L G, A polynomial algorithm in linear programming, *Doklady Akad. Nauk SSSR* **S244** (1979) 1093–1096
[14] Krishnamurthy E V and Sen S K, Numerical Algorithms: Computations in Science and Engineering (New Delhi: Affiliated East-West Press) (1993)
[15] Lakshmikantham V, Sen S K and Sivasundaram S, An inequality sorting algorithm for a class of linear programming problems, *J. Math. Anal. Appl*. **174** (1993) 450–460
[16] Lakshmikantham V, Sen S K and Howell G, Vectors versus matrices: p-inversion, cryptographic application, and vector implementation, *Neural, Parallel, and Scientific Computations* **4** (1996) 129–140
[17] Lakshmikantham V, Maulloo A K, Sen S K and Sivasundaram S, Solving Linear Programming Problems Exactly, *Appl. Maths. Comput.* **81** (1997) 69–87
[18] Lord E A, Sen S K and Venkaiah V Ch, A concise algorithm to solve under over determined linear systems, *Simulation* **54** (1990) 239–240
[19] Lord E A, Venkaiah V Ch and Sen S K, A shrinking polytope method for linear programming, *Neural, Parallel and Scientific Computations* **4** (1996) 325–340
[20] Natick M A, *MATLAB User's Guide* (The Math Works Inc.) (1992)
[21] Moore E H, On the reciprocal of general algebraic matrix (abs.), *Bull Am. Math. Soc*. **26** (1920) 394–395
[22] Murty K G, Linear Complementarity, Linear and Nonlinear Programming, (Germany: Heldermann Verlag, Berlin) (1989)
[23] Nazareth J L, Computer Solutions of Linear Programs (Monographs on Numerical Analysis) (Oxford: Oxford University Press) (1987) pp. 39–40
[24] Parker G and Rardin R, Discrete Optimization, (San Diego: Academic Press) (1988)
[25] Penrose R, A generalized inverse for matrices, *Proc. Chamb. Phil. Soc.* **51** (1955) 406–413
[26] Penrose R, On best approximate solutions of linear matrix equations, *Proc. Chamb. Phil. Soc.* **52** (1956) 17–19
[27] Peters G and Wilkinson J H, The least squares problem and pseudo-inverses, *The Computer J.* **13** (1970) 309–316
[28] Rao C R and Mitra S K, Generalized Inverse of Matrices and Its Applications (New York: Wiley) (1971)
[29] Renegar J, A polynomial-time algorithm, based on Newton's method for linear programming, *Math. Prog.* **40** (1988) 59–93
[30] Sen S K and Krishnamurthy E V, Rank-augmented LU-algorithm for computing generalized matrix inverses, *IEEE Trans. Comput.* **C-23** (1974) 199–201
[31] Sen S K and Prabhu S S, Optimal iterative schemes for computing Moore-Penrose matrix inverse, *Int. J. Systems Sci.* **8** (1976) 748–753
[32] Sen S K, Hongwei Du and Fausett D W, A center of a polytope: an expository review and a parallel implementation. *Internat. J. Math. Math. Sci.* **16** (2), (1993) 209–224

[33] Sen S K, Sivasundaram S and Venkaiah Ch, Barnes algorithm for linear programming: on detection of basic variables. *Nonlinear World*, **2** (1995)

[34] Stewart G W, On the continuity of the generalized inverse, *SIAM J. Appl. Math.* **17** (1969) 35–45

[35] Taha H A, Operations Research – An Introduction, 4th ed. (New York: Macmilan Publishing Company) (1989) 254–259

[36] Vaidya P M, An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations, *Proc. ACM Annual Symposium on Theory of Computing* (1987) pp 29–38; also *Math. Prog.* **47** (1990) 175–201

[37] Venkaiah V Ch and Sen S K, A floating-point-like modular arithmetic for polynomials with application to rational matrix processors, *Advances in Modelling and Simulation* **9(1)** (1987) 1–12

[38] Venkaiah V Ch and Sen S K, Computing a matrix symmetrizer exactly using modified multiple modulus residue arithmetic, *J. Comput. Appl. Math.* **21** (1988) 27–40

[39] Venkaiah V Ch and Sen S K, Error-free matrix symmetrizers and equivalent symmetric matrices, *Acta. Appl. Math.* **21** (1990) 291–313

[40] Wagner H M, Principles of Operations Research 2nd ed. (USA: Prentice Hall, Inc.) (1969) pp. 115–116