Neural, Parallel, and Scientific Computations 18 (2010) 479-486

CONCISE ALGORITHM FOR LINEAR PROGRAMS IN MATLAB: MONOTONIC CONVERGENCE, BASIC VARIABLES, BOUNDEDNESS

SYAMAL K. SEN AND SUJA RAMAKRISHNAN

Department of Mathematical Sciences, Florida Institute of Technology, Melbourne, FL 32901, United States
sksen@fit.edu, sramakrishna2009@fit.edu

ABSTRACT A physically concise polynomial-time iterative algorithm due to Barnes – a variation of Karmarkar projective transformation algorithm – is presented in Matlab for linear programs $Min\ c^t x\ subject\ to\ Ax = b,\ x \geq 0$. The concerned monotonic convergence of the solution vector and the consequent detection of basic variables are stated. The boundedness of the solution, multiple solutions, and no solution (inconsistency) cases are discussed. The possibility of applying Aitken's δ^2-process to accelerate convergence of the solution vector has been studied taking advantage of monotonic convergence. The effect of degeneracy of the primal linear program and/or its dual on the uniqueness of the optimal solution is mentioned. The foregoing algorithm is implemented in another way based on detection of basic variables and then solving the resulting linear system involving only the basic variables mathematically non-iteratively. The second way of implementation also includes optimality test and is coded in Matlab. It results in less number of iterations and usually more accurate optimal solution. Numerical experiments are carried out on these algorithms considering several typical linear programs and the Matlab implementation of these algorithms is found to be useful for solving many real-world problems.

1. INTRODUCTION

A linear program $Min\ c^t x\ subject\ to\ Ax = b,\ x \geq 0$ reduces to just solving a linear system mathematically non-iteratively once we know the basic variables that minimize the objective function $c^t x$. According to the fundamental theorem of linear programming, (i) if there is a feasible solution then there is a basic feasible solution and (ii) if there is an optimal feasible solution then there is an optimal basic feasible solution. Allowing the non-basic variables zero and deleting the column vectors of the coefficient matrix A corresponding to the non-basic variables we obtain a square nonsingular system $Bx = b,\quad x \geq 0$. Solving this system will readily produce the optimal basic feasible solution. The simplex method – an exterior method – computes only an optimal basic feasible solution. It does not compute an optimal non-basic feasible solution as is often produced by an interior method like the Karmarkar method and its variants [1-4]. An interior point method could produce either an optimal basic feasible solution or an optimal non-basic feasible solution, i.e., a solution that has more variables (with positive values) than the number of basic variables which are positive. In a real world situation, both have their scope. While the optimal solution need not be unique, the optimal value of the objective function will always be unique.

Here we present in Matlab a physically concise polynomial-time iterative algorithm due to Barnes for linear programs (LP), which is a variation of Karmarkar projective transformation algorithm. There are two ways of implementing the algorithm.

Received April 25, 2010

(i) One way is to continue the iterations for a sufficient number of times till we obtain the optimal solution of the LP to a desired accuracy, and

(ii) The other way is to take advantage of the monotonic convergence of the solution vector x; after some iterations when we discover that the elements of x have started converging, we discontinue the iteration, sieve out the basic variables, and solve the resulting linear system (consisting of only basic variables) mathematically non-iteratively to obtain the required optimal solution. To ensure that our sieving out of the basic variables was correct we perform an optimality test for the computed solution of the LP. In this procedure we can save significant number of iterations.

In section 2 we present the concise Barnes algorithm to solve the LP completely iteratively without making use of monotonic behavior of the successive solution vectors. Also we include in this section the second way of implementation as well as the concerned Matlab program. In this implementation, one may, however, obtain this basic solution by solving the non-singular system iteratively too. All these are coded in Matlab for the user to readily comprehend the procedure, modify the code easily if necessary, and use the code just by copying, pasting, and then executing for a given numerical LP. Section 3 comprises numerical examples while section 4 includes conclusions.

2 .BARNES ALGORITHM TO SOLVE LP AND DETECT BASIC VARIABLES WITH OPTIMAL SOLUTION

Let the linear program (LP) be

$$\textit{Minimize} \quad c^t x \quad \textit{subject to} \quad Ax = b, \quad x \geq 0, \qquad (1)$$

where $A = [a_{ij}]$ is a given $m \times n$ numerical matrix of rank m, $c = [c_1 \quad c_2 \quad \cdots \quad c_n]^t$ is a given numerical cost vector, $b = [b_1 \quad b_2 \quad \cdots \quad b_m]^t$ is a specified numerical response vector, and $x = [x_1 \quad x_2 \quad \cdots \quad x_n]^t$ is the unknown solution vector to be determined. Denote by a_j the j th column of A. The Barnes algorithm [3, 4] is as follows.

Algorithm 1(Barnes algorithm)
S.1 Choose $x^0 > 0$, i.e., select, for all the elements of the vector x^0, positive numerical values such that $Ax^0 = b$.
S. 2 If the vector x^k is known, then define the diagonal matrix $D_k = diag(x_1{}^k \quad x_2{}^k \quad \cdots \quad x_n{}^k)$, and compute $x^{k+1} > 0$ using the formula

$$x^{k+1} = x^k - r_k \frac{D_k{}^2(c - A^t\lambda_k)}{\| D_k(c - A^t\lambda_k) \|},$$

where $\lambda_k = (AD_k{}^2A^t)^{-1}AD_k{}^2c$, $r_k = \min\limits_{(c_i - a_i{}^t\lambda_k) > 0} \dfrac{\| D_k(c - A^t\lambda_k) \|}{x_i{}^k(c_i - a_i{}^t\lambda_k)} - \alpha_k$

For some $\alpha_k > 0$ such that $r_k > 0$. For proof of the algorithm, see [3, 4]. The foregoing algorithm converges to a solution of the LP (1). This convergence follows from the following theorems.

Theorem 1 If the LP (1) has a bounded solution then the foregoing algorithm converges to a solution of (1). For proof see [3, 4].

Theorem 2 Let $n = m+1$ in LP (1). Then the sequence $\{x_i^k\}$ converges monotonically, where i is such that the deletion of the i the column from the matrix A produces a non-singular matrix. For proof see [3, 4]. From Theorem 2 we can see that not only the non-basic variable converges monotonically but also one more variable converges monotonically. For the proof of the general case where $n > m+1$ we need to consider all possible combinations that grow exponentially with $n-m$. Although we have not proved this mathematically, our numerical experiments within the precision of computation, did not violate this for a general case.

The second algorithm ─ a second way of implementation of Barnes algorithm ─ is a combination of (i) partial Barnes algorithm (to detect basic variables taking advantage of monotonic convergence) and (ii) non-iterative algorithm to compute the basic solution along with its optimality test. This algorithm is as follows.

Algorithm 2 (partial Barnes algorithm + non-iterative linear system solver with optimality test)

S.1 Follow the foregoing Barnes algorithm till the monotonic convergence of the solution vector x^k sets in.

S.2 Choose those variables x_i^k which tend to positive values and reject those which tend to 0 along with their corresponding columns in the coefficient matrix A and also along with their corresponding coefficients in the cost vector c. Call the resulting coefficient matrix B, the resulting cost vector c_B, and the resulting (yet-to-be computed) solution vector x_B.

S.3 Compute the (basic) solution vector x_B of the linear system $Bx_B = b$. Construct the complete solution vector x^o of the LP inserting appropriately 0 (the value of the non-basic variables) in y.

S.4 Test the optimality of the computed solution x^o of the given LP *Minimize* $c^t x$ *subject to* $Ax = b, \quad x \ge 0$ as follows.

●Compute $y^t = c_B{}^t B^{-1}$ (row vector).

●Compute $z_j - c_j = y^t p_j - c_j$ (scalar), where $p_j = j-th$ non-basic vector in A.

●If all $z_j - c_j \le 0$ then the solution is optimal [5] else the monotonic convergence of the solution vector in Barnes algorithm has not yet set in; continue further iterations of Barnes algorithm till monotonic convergence is achieved. Redo Steps S.2, S.3, and S.4.

The precise iteration in which the monotonic convergence sets in is not known a priori. Hence the test of optimality is required to be used to determine whether the solution is truly optimal or not. The computational complexity is marginally increased. This second way of implementation (Algorithm 2) still remains polynomial-time.

Determination of Initial feasible solution Consider the LP (1). The initial feasible solution x^0 can be determined at least in two ways. One way is to append one artificial variable each to an equation of the constraint linear system $Ax = b$. Set $x_i = 0, \quad i = 1(1)n$ and $x_i = b_{i-n} \quad i = n+1(1)n+m$. The resulting solution vector x will have $n+m$ elements. The artificial variables are required to check inconsistency of the system $Ax = b,\ x \geq 0$. This way is usually used in a simplex algorithm and enhances the dimension of the system $Ax = b$ significantly needing at least m^2 storage locations in a computer.

The other way is to append only one artificial variable x_{n+1} to each of the equations of the system $Ax = b$ and determine its coefficient so that the initial feasible solution x will have each of the $n+1$ elements equal to 1. This increases the need to have only m storage locations plus one more in the cost vector c. We will consider the second way for solving all our numerical LPs in both the implementations (algorithms) described here.

Detection of basic variables The monotonic convergence of the algorithms permits the detection of both basic and non-basic variables in the LP (1). The following table depicts the detection for different types of LPs.

Primal LP[1]	**Dual LP**	**Solution Type**	**Procedure for Detection of Basic Variables**
Non-degenerate	Non-degenerate	Unique	After a sufficient number of iterations, the m columns of A corresponding to m basic variables which have larger values are chosen and the resulting linear system with $m \times m$ non-singular coefficient matrix is solved. The remaining $n-m$ non-basic variables are set to 0.
Non-degenerate	Degenerate	Multiple having $k \geq m$ variables positive (non-zero)	If the algorithms converge to a solution with exactly m variables non-zero then these are basic (not the case, in general), else detection is hard [4].
Degenerate	Non-degenerate	Unique having $k < m$ variables positive (non-zero)	The basis here is not unique and may not be detectable. The k variables (basic) will produce the optimal solution allowing the remaining $n-k$ variables 0. After a sufficient number of iterations, choose m columns that include the k columns and solve the resulting $m \times m$ system allowing the values of arbitrary $m-k$ variables 0.
Degenerate	Degenerate	Multiple	Detection is hard here too [4]. For details of detection of basic variables in the foregoing two multiple solution cases, see [4].

[1] If the LP $Minimize\ c^t x\ subject\ to\ Ax \geq b,\ x \geq 0$ is considered primal then its dual is $Maximize\ b^t y \cdots subject\ to\ A^t y \leq c,\ y \geq 0.$

The Matlab Code We now present a Matlab program for the foregoing algorithms. Although the Matlab program computes the optimal solution using entirely the Barnes algorithm, one may stop continuing the iterations once the monotonic convergence sets in and then detect those variables which tend to positive values different from numerical zero [6]. These variables are basic variables while other variables that tend to 0 are non-basic variables. We remove those columns from the coefficient matrix A, that correspond to the non-basic variables. We also remove the non-basic variables from the solution vector x (without changing the identity of the basic variables). The resulting solution vector, also called x, is computed using the Matlab command $x = A \backslash b$. This computed solution vector along with the non-basic variables with values 0 constitutes the required optimal solution if it passes the optimality test. The following Matlab program is not completely automated and not completely general, One can make it so with some programming effort. Unlike the main-frame computing era during mid and late 20th century when many users would be using a single physically large computer in batches, one has one high speed computer (laptop/desktop) always at his disposal. Thus one can inspect the outputs/results again and again and accordingly decide on the basic variables instead of computer performing this task based on the code supplied/implemented by the user. The Matlab program *Barnes_4* below is self-explanatory, needs no formal programming knowledge to follow, and can be easily modified by the user according to his need.

```
function boolean = Barnes_4(A,b,c,epsilon,c1) %c1= parameter added to
%coeff. of artificial variable.
n1=n+1; m1=m+1; xk=zeros(n1,1); cf=c;
for j=1:m, A(j,n1)=b(j) - sum(A(j,:)); end;
c(n1)=sum(abs(c))+c1; for i=1:n1, xk(i)=1; end;
disp('The coefficient matrix A is ') , disp(A), disp('The rhs column
vector b is ') , disp(b')
disp('The cost vector c is ') , disp(c'), disp('The initial feasible
solution is '), disp(xk')
f=c'*xk; disp('The initial objective function value is '), disp(f)
xaitken(:,1)=xk; total=sum(xk); minimum=min(xk); ctr=1;
while (minimum>total*epsilon && ctr <= 2*n)
Dk=diag(xk); lambdak=(pinv(A*Dk^2*A')*A*Dk^2*c); beta=norm(Dk*(c-
A'*lambdak));
    for i=1:n1,  psi(i) = c(i) - A(:,i)'*lambdak;
        if (abs(psi(i)) < 0.001*abs(c(i))), psi(i) = (c(i)^2-
(A(:,i)'*lambdak)^2)/(c(i)+A(:,i)'*lambdak); end;
        phi(i) = xk(i)*psi(i);
    end;
    j=1; for i=1:n1, if psi(i) > 0, del(j) = beta/phi(i); j=j+1; end;
end;
    Rk=0.9*min(del); xk=xk-((Rk*Dk^2*(c-A'*lambdak))/beta);
    display('Solution xk is '), disp(xk'), total=sum(xk);
minimum=min(xk); fi=f;
    f=c'*xk; xaitken(:,ctr+1)=xk;
    if xk(n1) > 0.3 * total, disp('The problem is not feasible');
break; end; ctr=ctr+1; end;
disp('The final solution is '), disp(xk'), disp('The objective function
value is '), disp(c'*xk)
for j = 1:n1
    if xaitken(j,ctr) - ((xaitken(j,ctr) - xaitken(j,ctr-
1))^2)/((xaitken(j,ctr) - 2*xaitken(j,ctr-1) + xaitken(j,ctr-2))) > 0
```

```
       xaitken(j,ctr+1) = xaitken(j,ctr) - ((xaitken(j,ctr) -
xaitken(j,ctr-1))^2)/((xaitken(j,ctr) - 2*xaitken(j,ctr-1) +
xaitken(j,ctr-2)));
    else xaitken(j,ctr+1) = xk(j);
    end; end;
A*xaitken(:,ctr+1); xkaitken=xaitken(:,ctr+1); f=c'*xaitken(:,ctr+1);
disp('Aitkens delta-squared process results ')
disp('The final solution vector after applying Aitkens delta-squared
process is')
disp(xkaitken'),disp('Aitken Objective function value is '), disp(f)
bool = 1;
if xk(n1) > 0.001, disp('The problem is infeasible'), bool = 0; end;
if (abs((fi^2-f^2)/(fi+f)) > (0.02 * abs(f))), disp('Check solutions
for possible unboundedness'); end;
[X,Index]=sort(xk,'descend');
for i=1:m, Afinal(:,i)=A(:,Index(i)); cfinal(:,i)=c(Index(i),:); end;
if X(m1)>0.2*total/n1, disp('Check for possible multiple solutions'),
bool = 2; end;
%   Algorithm 2 begins
if   bool ~= 0,j=1; k=1;
        for i=1:n
            if xk(i) >= 0.2*total/n1,Ind(j,:)=i; B(:,j)=A(:,i);
cB(j)=c(i); j=j+1; else Ind1(k,:)=i; k=k+1; end; end
        x0=B\b; S = size(Ind); y=zeros(n,1); ctr=1;
        for j = 1:S(1), y(Ind(j,:))=x0(ctr); ctr=ctr+1; end
        if S(1) < n
            for k=1:n-S(1), Zk(k)=(cB*pinv(B))*A(:,Ind1(k,:)) -
c(Ind1(k,:)); end; end
        if(max(Zk) <= 0)
            disp('The optimal solution to the LP is '),  disp(y')
            disp('Objective function value is '), disp(cf'*y)
        end; end
```

3. NUMERICAL EXAMPLES

We have considered several LPs — general, infeasible, Beale's cycling, multiple solution, near-multiple solution, unbounded solution, cost vector perturbed. To conserve space we just provide one general LP and one multiple solution LP.

(i) *General LP* [7]

$$Min\ 2x_1 + 7x_2 - 2x_3\ subject\ to\ x_1 + 2x_2 + x_3 + x_4 = 1, -4x_1 - 2x_2 + 3x_3 + x_5 = 2,\ x_i \geq 0\ \forall\ i$$

The command for the Matlab program Barnes_4 is

>>Barnes_4([1 2 1 1 0;-4 -2 3 0 1],[1 2]',[2 7 -2 0 0]', 0.5*10^-7, 10)

After 11 iterations we obtain the solution $x_1 = .1428, x_2 = 0, x_3 = .8571, x_4 = 0, x_5 = 0,$ objective function (ofv) value is -1.4285. The exact solution of the LP is $x_1 = 1/7, x_2 = 0, x_3 = 6/7, x_4 = 0, x_5 = 0, ofv = -10/7$.

(ii) Multiple solution LP [4]

$Min\ -2x_1 - x_2\ subject\ to\ 2x_1 - 2x_2 + x_3 = 1,\ 2x_1 - 3x_2 + x_4 = 1,\ 2x_1 + x_2 + x_5 = 2,\ x_i \geq 0\ \forall i$

The command for the Matlab program Barnes_4 is

```
>> Barnes_4([2 -2 1 0 0;2 -3 0 1 0;2 1 0 0 1], [1 1 2]', [-2 -1 0 0 0]', 0.5*10^-7, 10)
```

After 10 iterations we get the solution (a non-basic variables)
$x_1 = .7118,\ x_2 = .5763,\ x_3 = .7289,\ x_4 = 1.2052,\ ofv = -2$.

4. CONCLUSIONS

It is not necessary to know a priori whether the primal LP is degenerate or not. We exit at some iteration based on the accuracy required or, equivalently, the relative error permitted or, equivalently, the numerical zero defined in this context [6]. Call this exit parameter ε. One may choose ε as $[0.5\times10^{-7}\times\sum_{i=1}^{n+1} x_i^{\ k}]/(n+1)$. We have, however, chosen ε just as the foregoing expression without denominator $(n+1)$, i.e., allowing the denominator to be equal to 1. The iterations is continued till the value of any one of the variables including the artificial variable is less than ε or till a specified number of iterations, chosen here as $2n$, whichever is satisfied earlier. For near degenerate cases, the precision of the computer comes into effect. The numerical zero or, equivalently, the exit parameter ε needs to be redefined as a smaller value and also the specified number of iterations needs to be increased appropriately. It may, however, not be effective if the precision of computation is not sufficiently large or, in other words, if the LP is too near-degenerate (with respect to the precision).

Error-free arithmetic such as the multiple modulus residue arithmetic and p-adic arithmetic cannot be implemented in these algorithms since these involve not only square-rooting operations but also no a priori known fixed number of arithmetic operations [6, 8-10].

Unlike the simplex algorithm (an exterior point method), Barnes algorithm (an interior point method) ─ a variation of Karmarkar algorithm ─ may obtain a non-basic optimal solution (more variables>0 than number of equations in $Ax = b$).; see for instance Example (ii): multiple solution LP.

It can be seen that the Barnes algorithm needs much less number of iterations and much less computer storage due to much smaller dimension of the LP (in Barnes method) than those in Karmarkar algorithm.

For infeasible LPs, the artificial variable will occur in the final solution as a positive value. The infeasibility is, in fact, known from the value of the artificial variable, which does not vanish indicating that the LP is inconsistent (and hence infeasible).

For LPs with unbounded solution, the elements of the solution vector will become very large in magnitude.

Aitken's δ^2-process [7] does not seem to help much. However, we are in the process of exploring complete potential of the δ^2 process.

REFERENCES

1. S.K. Sen and Sagar Sen, Karmarkar form of linear program and algorithm: Precise presentation, Proc. 45th International Congress of the Indian Society of Theoretical and Applied Mathematics, Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu, India, Dec 26-29, 2000, 88-96.
2. N. Karmarkar, A new polynomial-time algorithm in linear programming, *Combinatorica*, **4**, 1984, 373-395.
3. E.R. Barnes, A variation of Karmarkar's algorithm for solving linear programming problems, *Math. Program.*, 1986, 174-182.
4. V. Ch. Venkaiah, Variation of Karmarkar's algorithm for linear programming: On detection of basic variables, Ph. D. Thesis, Department of Applied Mathematics, Indian Institute of Science, Bangalore, October, 1987.
5. H.A. Taha, Operations Research: An Introduction, Macmillan, New York, 1989.
6. V. Lakshmikantham and S.K. Sen, Computational Error and Complexity in Science and Engineering, Elsevier, Amsterdam, 2005.
7. E.V. Krishnamurthy and S.K. Sen, Numerical Algorithms: Computations in Science and Engineering, Affiliated East-West Press, New Delhi, 2007.
8. V. Lakshmikantham, S.K. Sen, and A. Mohanty, Error in error-free computation for linear system, *Neural, Parallel & Scientific Computations*, **12**, 2004, 113-122.
9. V. Lakshmikantham, S.K. Sen, A.K. Maulloo, and S. Sivasundaram, Solving linear programming problems exactly, *Applied Mathematics and Computation* (Elsevier Science Pub. Co., New York)**, 81,** 1997, 69-87.
10. R.T. Gregory and E.V. Krishnamurthy, Methods and Applications of Error-free Computation, Springer-Verlag, New York, 1984.