

# An Algorithm for Binary Linear Programming

Subhendu Das<sup>1</sup>

## Abstract

A polynomial time algorithm, which is a modification of the simplex algorithm for Linear Programming (LP), is presented for solving Binary Linear Programming (BLP) problems. It is an n-step process where n is the number of binary variables of the problem. Conditions for existence, uniqueness, termination, and optimality are discussed. Three numerical examples are presented to explain the details and features of the algorithm. The algorithm generates two matrices. One of them is based on the minimum ratio concept of the Simplex method. It creates a Ratio Test Matrix from the ratio of corresponding elements of b vector and all column vectors. The other matrix, called Correlation Matrix, is generated from pseudo inverse estimates of the primal and dual variables. These two matrices provide an array of Column Selection Logic to assign binary values for variables corresponding to the matrix columns.

**Mathematics Subject Classification:** 90C10; 90C05; 90C09

**Keywords:** Integer programming; Binary programming; Optimization; Simplex method

---

<sup>1</sup> CCSI, West Hills, California, USA. E-mail: Subhendu11Das@gmail.com

## 1 Introduction

In this paper we present an algorithm for solving Binary Linear Programming (BLP) problems. The algorithm is based on pseudo inverse and concepts derived from the simplex algorithm for standard Linear Programming (LP) problems.

The BLP is defined in the following way.

$$\begin{aligned} \text{BLP: Maximize} \quad & c^T x \\ \text{Subject to} \quad & Ax = b, \quad A \in R^{m \times n}, \quad b \in R^m, \quad m < n, \quad x \in \{0,1\}^n \end{aligned} \quad (1)$$

The above means that  $x$  can take only binary values, 0 or 1, in the solution for the matrix equation  $Ax = b$ . Without loss of any generalities we will assume that the matrix  $A$  has full row rank, all elements of  $A$  are nonnegative (Section 4), and that the vector  $b$  belongs to the binary range space, as defined later, of columns of  $A$ . In a similar way the LPLU is defined as

$$\begin{aligned} \text{LPLU: Maximize} \quad & c^T x \\ \text{Subject to} \quad & Ax = b, \quad A \in R^{m \times n}, \quad b \in R^m, \quad m < n, \quad x \in R^n, \\ & 0 \leq x_i \leq 1, \quad i = 1, \dots, n \end{aligned} \quad (2)$$

The difference is that in LPLU all solutions will have positive real values between 0 and 1, but in BLP only 0 or 1 values are allowed.

Although the BLP algorithm borrows ideas from the Simplex method but it does not carry the pitfalls of the Simplex method with it. However, it should be noted that there are variants of the Simplex method that has solved many such pitfalls [Vanderbei, p. 32]. LP is considered as polynomial time problem [Dasgupta, p. 243], but we are not directly using LP methods, other than verifying our results using LPLU. Although Simplex is considered as a non-polynomial time algorithm [Luenberger, p.114], even then our BLP algorithm belongs to the polynomial time solutions. Once we go through the examples it will be clear that BLP algorithm ends in at most  $n$  iterations, where  $n$  is the number of variables in

(1). Like the Simplex method [Vanderbei, p.32] the BLP algorithm presented here consists of a class of algorithms under a common basic structure. BLP is considered as NP complete problem [Dasgupta, p. 243]. It seems then the BLP algorithm presented here will resolve an important issue.

First we present the algorithm in some details using a pseudo mathematical language. Next, we discuss some theoretical results that justify the validity of the algorithm. Finally we solve three numerical example problems to explain more details of the algorithm, computational techniques, types of tables we may need, and some variants for implementations. We also discuss a numerical method for generating test problems for our algorithms. This test method actually helps to identify how to reverse engineer the solution of BLP. We present one example from such class of problems. The algorithm has been tested extensively for these kinds of test problems for both small and medium size examples.

## **2 The BLP Algorithm**

We can now state the algorithm using some kind of descriptive language and in a hierarchical way using pseudo code. The main program is shown in Code-A0 in Figure-1

As we can see that the main program has three sub-programs. The sub-program 3 has again two more sub-programs, which are marked as 3a and 3b. All these subprograms are repeated n times as indicated in the first line. The Loop exits if any one of the subprograms 3a or 3b becomes valid. We describe the subprograms 1 and 2, using similar pseudo codes, as in Code-A1 and Code-A2 in the same Figure-1.

## 2.1 Process Ratio Test Matrix (RTM)

This part of the algorithm creates a matrix  $\{r_{ij}\}$ . The idea is borrowed from the Simplex method, but we create an entire matrix. Each element of the RTM is obtained by dividing the corresponding elements of b vector by the elements of the a vectors, which are the columns of the A matrix, as shown in the second line in the Code-A1. There are three possible values for  $r_{ij}$  which we analyze next.

If  $r_{ij}$  is less than 1 then the j-th component of x cannot be set to 1, that is,  $x_j$  cannot have a value of 1. This is because the ratio means  $a_{ij}$  is greater than  $b_i$  and  $a_{ij} x_j > b_i$  will violate the constraint i. Thus  $x_j$  must be assigned as 0. This is shown in line four of the Code-A1. It is clear that this step gives an optimal CSL.

Code-A0: Main Program	
Repeat n times	
1	Process Ratio Test Matrix (RTM);
2	Process Reduced Cost Row (rC);
3	Stop If
	3a All variables are processed or
	3b Norm of b is zero

Code-A1: Process Ratio Test Matrix: $\{r_{ij}\}$	
Create RTM	
	$r_{ij} = b_i/a_{ij}, a_{ij} \neq 0$
	If $\exists r_{ij} < 1$
	Set $x_j = 0, c_j = 0, a_j = 0$
	If $\exists r_{ij} = 1$
	Decide if $x_j = 1$ is possible or ignore
	$\forall r_{ij} > 1$
	Do nothing and exit subprogram

Code-A2: Process Reduced Cost Row: rC	
$Ax = b \Rightarrow X_e = A^{-1}b$ ; If $X_e$ is binary Stop.	
$A^T y = c \Rightarrow Y_e = A^{-T}c$	
	Correlation Matrix = $R_{ij} = \text{Abs}[Y_{ei} * X_{ej}]$
	Max row = $r_j = \text{Max}[R_{ij}, \forall i]$ , for $\forall j$
	Reduced cost row = $rC_j = r_j * c_j, c_j \neq 0$
	Max Index = $\{k   rC_k = \text{Max}[rC_j, \forall j]\}$
Setup for next iteration	
$x_k = 1, b_{i+1} = b_i - a_k, c_k = a_k = 0$	

Figure 1: Pseudo code for BLP algorithm

It is optimal because you cannot increase the variable to 1 to raise the cost value.

If the ratio is 1, then  $x_j$  can take both 1 or 0 value. But if we assign 1, then no other variables in the i-th constraint can be 1 because  $a_{ij} x_j = b_i$  will satisfy the constraint with equality. Thus assigning 1 will stop the BLP algorithm and we

will exit. If that is acceptable then the result is optimal and we have found an optimal CSL in this step. If that is not acceptable, then we can set the variable to 0, or ignore the variable for now, and the future steps will take care of it. This is mentioned in line 6 of Code-A1. Examples illustrate these cases in a better way.

For all  $r_{ij} > 1$  there is nothing that can be done, so skip the algorithm described in Code-A1 and begin examining the process described by Code-A2. This is the last line of Code-A1.

Code-A1 of the column selection logic (CSL) may assign values to many variables in a single iteration, as illustrated by examples. As a database management process, we set the cost coefficient corresponding to each assigned  $x$  variable to zero, so that we do not process the same column in any future iteration. Also we remove the corresponding columns from the matrix  $A$ . More details of the process are available from the computer programs and examples.

## 2.2 Process Reduced Cost Row

This is the second most important sub-function of the BLP algorithm. Like in simplex method, we find a reduced cost row and then from this row we select the optimal cost element and the optimal column. This then becomes the optimal column selection logic (CSL) for this step. This step is described in Code-A2.

In the simplest case, the reduced cost row can be just the original cost row vector. That is, the elements  $r_j = 1$  in line 5 of Code-A2. But in a general case we can multiply each cost element by a weight  $r_j$  derived from some meaningful process. As an example we choose a correlation matrix. This correlation matrix estimates how  $x$  and  $y$  vectors are related to matrix  $A$  and how they create the vectors  $b$  and  $c$  respectively. The idea behind this comes from the duality theorem of standard LP program and has been explained in some details in the test problem generation method (Section 4).

In a standard LPLU, the  $b$  vector is created by a linear combination of columns of  $A$  matrix. That is, each element in a row  $A$  is multiplied by the corresponding element of  $x$  to produce a factor  $a_{ij} x_j$  for the corresponding element of  $b$ . Similarly each element of  $A$  is multiplied by a corresponding element of  $y$  to get a factor for the corresponding element of  $c$ . This relationship comes from the law of conservation that relates  $A$ ,  $b$ ,  $c$ ,  $x$ , and  $y$  as pointed out in [Dantzig, p. 33]. Since we are dealing with BLP as opposed to LPLU, we try to estimate these factors from the correlation matrix elements obtained from the corresponding pseudo matrices.

The formulation of a mathematical model is not dependent on the nature of solution. This formulation should be based on the laws of nature, i.e. conservation of mass, energy, forces, voltage, current etc. Any mathematical model that violates the laws of nature is fictitious and hypothetical. Thus correlation concept will always be there, be it LPLU or BLP. However, rather than correlation matrix, depending on the problem, a better or simpler algorithm for the CSL may be feasible, as illustrated by examples.

The first line of Code-A2 gets an estimate for  $x$  using pseudo-inverse. If the estimate of  $x$  is a binary vector then we have found the solution and the BLP algorithm ends. For, this binary vector can be used for the solution of the remaining  $x$  variables. We claim that this binary vector is the optimal vector. We show later that if the number of columns of the matrix is  $k \leq m$ , then this binary solution is unique. This happens because a full rank matrix always has  $k \leq m$  number of independent columns. The rest of the lines in Code-A2 will be skipped. It is clear that this step gives an optimal CSL. If however  $k$  is greater than  $m$ , then we should ignore the binary solution and continue with the rest of the steps in the algorithm.

If there is no binary vector then we estimate  $y$  and create the correlation matrix  $R_{ij}$  as defined in lines 2 and 3 in Code-A2. Then create the reduced cost row as shown in line 5 of the Code-A2. The rest of the rows of the table are self

evident. It should be clear from this algorithm for Code-A2 that this is an optimal procedure for this step, i.e. an optimal CSL, since it selects, like in the Simplex method, the variable that gives the maximal value for this part of the BLP.

### 3 Theoretical Framework

First we define some basic notations and terminologies to refresh the communication channels with all readers. These are well known and simple definitions.

**Definition 3.1** Column Selection Logic (CSL) is a logic or an algorithm that selects a column of the matrix  $A$  of the BLP defined by (1). It is a method of assigning a specific value, 0 or 1, to a variable  $x_j$ .

**Assumption** Let us assume that this CSL is an optimization process at each step.

Then we show that an application of a finite sequence of such CSL finds the optimal solution of the BLP problem (1). If each CSL is a polynomial time algorithm then the entire BLP algorithm is also a polynomial time algorithm.

**Definition 3.2** Assume that the matrix  $A$  of (1) consists of the following  $m$ -dimensional columns of  $A$

$$A = \{a_i \mid a_i \in R^m, i = 1, \dots, n\}.$$

The Binary Range Space  $\mathcal{B}$  of  $A$  is defined as all binary linear combinations of columns of  $A$ , that is,

$$\mathcal{B} = \{w \mid w = \sum_{i=1}^n x_i * a_i, x_i \in \{0,1\}, a_i \in A\}$$

**Definition 3.3** We define the feasible region of (1) without the binary constraints as:

$$F = \{x \mid Ax \leq b, x \geq 0, x \in R^n\}$$

That is, F represents all positive vectors in the n-dimensional real vector space satisfying (1) with less than equal to condition.

**Definition 3.4** Similarly, the n-cube is defined as the n-dimensional hyper cube U, with length of each side being 1 along each axis. Thus U can be written as:

$$U = \{x \mid x \in R^n, \quad 0 \leq x_i \leq 1, \quad \forall i = 1 \dots n\}$$

**Definition 3.5** The corner points of U can be defined as a sequence of n binary digits or bits

$$C = \{x \mid x \in R^n, x_i \in \{0,1\}, \forall i = 1 \dots n\}.$$

Observe that  $x \in \{0,1\}^n$  means that x is a corner point of U.

**Existence** It is clear that if F and C do not intersect then there cannot exist any solutions to the BLP problem (1). Thus an optimal solution to (1) will exist only when F contains some corner points of C. One of these corner points will be the optimal solution. And therefore we do not need to analyze the corner points of the feasible region F. The unboundedness of F does not also affect the solutions of (1), as long as F has a non-empty intersection with C.

**Theorem 3.1** If a solution to (1) exists then it has an optimal solution. The optimal solution is always bounded, as long as the coefficients of the objective hyper plane are bounded.

**Proof** The proof of Theorem 3.1 is very trivial. By definition all solutions of (1) are the corner points of C. If any of the corner points are inside the feasible region F, then the solutions to (1) exists. Since the feasible corner points of U are finite in number, all of them can be enumerated to find the maximal value to the BLP



problem (1). Since all corner points are bounded, and they are finite in number, the objective values for each corner point is finite and bounded, since the coefficients are bounded.  $\square$

The following theorem is very fundamental to the CSL based algorithms for the BLP. Without loss of generality we can state

**Theorem 3.2** Let  $u = \{x_1, x_2, \dots, x_{n-1}, x_n\}$ ,  $x_1 = 1$ ,  $v = \{x_2, \dots, x_{n-1}, x_n\}$ ,  $x_i \in \{0,1\}$ . Then  $u$  is a binary solution of the equation

$$Ax = b, \quad A = \{a_1, a_2, a_3, \dots, a_n\}, \quad \text{Rank}[A] = m, \quad b \in \mathcal{B} \quad (3)$$

If and only if  $v$  is the binary solution of

$$Bx = b - a_1, \quad B = \{a_2, a_3, \dots, a_n\} \quad (4)$$

**Proof** As before the proof is very trivial also. Since  $u$  is a solution of (3) we can write

$$a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n = b$$

As  $x_1 = 1$ , the last expression reduces to

$$a_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n = b \quad \text{or}$$

$$a_2x_2 + a_3x_3 + \dots + a_nx_n = b - a_1$$

This clearly indicates that  $v$  is a solution of (4). Conversely if  $v$  is the solution of (4) then we have

$$a_2x_2 + a_3x_3 + \dots + a_nx_n = b - a_1, \quad \text{choosing } x_1 = 1 \text{ we can write}$$

$$a_2x_2 + a_3x_3 + \dots + a_nx_n = b - a_1x_1 \quad \text{or} \quad a_1x_1 + a_2x_2 + a_3x_3 + \dots + a_nx_n = b$$

That is,  $u$  is a solution of (3).  $\square$

The above theorem is true for any  $x_i$  and not just for  $x_1$ . It is also true for more than one variable fixed at predefined binary values. The theorem holds also for 0 values for the variables. In case of 0 values we just do not have to subtract

the columns from the  $b$  vector. It should be recognized that the above theorem is in general not correct for real number solutions of  $x$ .

It is clear that the CSL will stop after  $n$  iterations, if after each step the cost coefficient for the selected variable is set to zero. Thus CSL must be based on the non-zero cost elements.

**Theorem 3.3** The CSL algorithm ends in less than equal to  $n$  number of iterations, where  $n$  is the number of variables in the BLP problem (1).

**Proof** The CSL algorithm always selects a column that has nonzero cost value in the reduced cost row. After it selects a column, the process automatically sets the cost to zero, for that column in the cost row, as a data base management step of the software. Setting the cost to zero indicates that the column is to be eliminated from all future steps. Thus within  $n$  steps all elements of the reduced cost row in Table-A2 will become zero and the process will stop.

As the columns of  $A$  matrix are subtracted from the  $b$  vector, as stated in Theorem 3.2, the values of  $b$  vector reduces. This step will eventually make many columns of  $A$  larger than  $b$ . Thus the CSL will set all of their costs to zero in one step, because of the ratio test matrix of Code-A1. This process will reduce the number of steps to less than  $n$ .

If the CSL uses pseudo inverse then at some point in the chain of iterations, the CSL may encounter binary values from the pseudo inverse. This process then also will set many variables to 0 and 1 in one single step. Thus the CSL algorithm ends in at most  $n$  number of steps. □

**Theorem 3.4** The CSL algorithm finds the optimal solution for the BLP.

**Proof** Because there are only two choices, at each iteration the CSL algorithm assigns a 0 or an 1 value to one of the variables  $x_i$ . In this process the algorithm moves, at each step, from higher dimensional hyper space to the next lower dimensional hyper space. It is also clear that in this process the CSL algorithm is

climbing up along the corner points of the n-cube, because the cost functional is increasing and is going away from origin. Since the optimal solution is always a corner point of the n-cube, at every step the cost functional increases its value or remains same, it never goes down. Thus when the process ends the algorithm finds the optimal solution.

Let us assume that  $u$  and  $v$  are two bit patterns or corner points of  $C$  claiming to be the optimal solutions. Without loss of generality we can assume that cost coefficients are arranged in a decreasing order. Let  $k$  be the first bit position where  $u$  and  $v$  differ, that is,  $u$  has a 0 and at that same location  $v$  has an 1. This means  $v$  gives a higher value for the objective function. Now this cannot happen, because it is assumed that the CSL gives higher value at each step. Therefore  $u$  must be same as  $v$ .  $\square$

**Theorem 3.5** The CSL is a polynomial time algorithm.

**Proof** The computationally involved most complicated step in the CSL logic is the evaluation of pseudo inverse of  $A$  matrices at each step. In the simplest case of pseudo inverse, the process will evaluate the matrix directly from the relation  $A^- = A'(AA')^{-1}$ . This step takes two matrix multiplications and one matrix inversion. It is well known [Trahan] that they are polynomial time algorithms. Thus an iteration of the CSL logic involves only polynomial time computations. Therefore the entire BLP algorithm presented is a polynomial time process.  $\square$

The foundation of the CSL algorithm is based on an assumption which is established by the following matrix rank theorem.

**Theorem 3.6** Let the matrix  $A$  be as in (1) with full row rank  $m$ . Then every set of  $k \leq m$  column vectors of  $A$  is independent.

**Proof** Since the matrix  $A$  has full row rank  $m$ , we can then write

$$\sum_{i=1}^m c_i A_i = 0 \text{ or } \sum_{i=1}^m c_i \{a_{i1}, a_{i2}, \dots, a_{in}\} = \{0, 0, \dots, 0\}, \text{ } n\text{-tuple} \quad (5)$$

Where  $c_i = 0$ , are linear combination coefficients of the rows  $A_i$  of the  $m \times n$  matrix  $A$ . Let us choose any set of  $k \leq m$  columns of the matrix, for these columns we can derive from (5)

$$\sum_{i=1}^m c_i \{a_{ij_1}, a_{ij_2}, \dots, a_{ij_k}\} = \{0, 0, \dots, 0\}, \text{ } k\text{-tuple}$$

The above then gives  $k$  simultaneous equations, one for each one of the  $k$  columns.

$$\begin{aligned} c_1 a_{1j_1} + c_2 a_{2j_1} + \dots + c_m a_{mj_1} &= 0 \\ c_1 a_{1j_2} + c_2 a_{2j_2} + \dots + c_m a_{mj_2} &= 0 \\ &\dots \\ c_1 a_{1j_k} + c_2 a_{2j_k} + \dots + c_m a_{mj_k} &= 0 \end{aligned} \quad (6)$$

Multiplying each equation by an arbitrary constant  $\{u_1, u_2, \dots, u_k\}$  and summing them together we can write as

$$u_1 \sum_{i=1}^m c_i a_{ij_1} + u_2 \sum_{i=1}^m c_i a_{ij_2} + \dots + u_k \sum_{i=1}^m c_i a_{ij_k} = 0 \quad (7)$$

Observe that (6) and (7) are equivalent, since  $u$  variables are arbitrary. To prove that the  $k$  columns are independent we must show that

$$\begin{aligned} b_1 a_{1j_1} + b_2 a_{1j_2} + \dots + b_k a_{1j_k} &= 0 \\ b_1 a_{2j_1} + b_2 a_{2j_2} + \dots + b_k a_{2j_k} &= 0 \\ &\dots \\ b_1 a_{mj_1} + b_2 a_{mj_2} + \dots + b_k a_{mj_k} &= 0 \end{aligned}$$

Multiplying each equation by an arbitrary constant  $\{v_1, v_2, \dots, v_m\}$  and summing them together we can write as

$$b_1 \sum_{i=1}^m v_i a_{ij_1} + b_2 \sum_{i=1}^m v_i a_{ij_2} + \dots + b_k \sum_{i=1}^m v_i a_{ij_k} = 0 \quad (8)$$

Comparing the coefficients of each matrix element, in (7) and (8), we find, for example, that

$$a_{1j_1}: u_1c_1 = b_1v_1 \text{ or } 0 = b_1v_1 \text{ or } b_1 = 0$$

Similarly we can show that all b values will be zeros. This shows that the k columns are independent.  $\square$

## 4 Test Problems

The algorithm has been tested on various problems. Many problems were taken from standard textbooks and published literature. However, majority of test problems were constructed using a specific method described here. The Mathematica software tools have been used for many of our problems. One of the important requirements of the test problems is that it must have a proven known optimal solution; so that the results of the algorithm can be compared. And the other important factor is that the matrix A must be of full row rank, i.e.,  $\text{rank}(A) = m$ , where m is the number of rows.

**Definition 4.1** The set of functions  $\{g_1(t), g_2(t), \dots, g_n(t)\}$  is linearly dependent on an interval I if there are scalars  $c_1, c_2, \dots, c_n$  not all zero such that

$$c_1g_1(t) + c_2g_2(t) + \dots + c_ng_n(t) = 0$$

for all t in I. Otherwise the set is linearly independent [Farlow, p.179].

$g_1(t) = 1 + \text{Sin}(2 \pi f_0 t)$	$g_2(t) = 1 + \text{Cos}(2 \pi f_0 t)$
$g_3(t) = 1 - e^{-3000t}$	$g_4(t) = 1 + \text{Cos}(4 \pi f_0 t)$
$g_5(t) = 1 + \text{Sin}(4 \pi f_0 t)$	$g_6(t) = 1 + \text{Sin}(\pi f_0 t)$
$g_7(t) = 1 + \text{Cos}(\pi f_0 t)$	$g_8(t) = 1 + \text{Cos}(6 \pi f_0 t)$
$g_9(t) = 1 + \text{Sin}(6 \pi f_0 t)$	$g_{10}(t) = e^{-3000t}$

Figure 2: A set of Independent Functions

To create such an A matrix of any given size with m independent rows, we selected m independent functions, and then sampled them n times to produce the elements of the m x n matrices [Das]. All these functions are defined over a fixed time interval. Thus, as n increases the sample interval decreases. To show that the algorithm can handle a problem of reasonable size we have considered one problem with m = 10 rows, and so we have 10 functions. These functions are listed in Figure-2. It is clear that there are many choices of independent functions. Observe that all functions produced only nonnegative numbers.

We define the parameters of the above functions as

$$T = 0.001, \quad f_0 = \frac{1}{T}, \quad 0 \leq t \leq T$$

The equal sample interval is  $dT = T/n$ . The first sample starts at  $t = dT$ . The characteristics of the problem sets are quite dependent on the types of functions and the way they are sampled.

We have used random number generator for creating arbitrary binary solution vector x. To create a n-bit binary sequence, we used an integer of size  $2^n$ . Then we generated a random integer number within that size, starting with a given seed value for the random generator. Finally we converted the random integer to a binary number with n-digits. We multiply A by this binary number x to get the b vector. So our BLP algorithm takes these A and b matrices and solves for the x. The solution will be correct only when it matches the known random x vector.

To ensure optimality, we used another random sequence of m real numbers for the dual variable y for the standard linear programming problem. Then we use

$A^T y = c$  to generate the cost vector  $c$ . This will automatically ensure that the  $x$  vector is the optimal solution for the LPLU, because the variables so selected will satisfy the optimality condition:

$$y^T b = c^T x = y^T Ax = z$$

Such a LPLU problem will always have binary vector  $x$  as the optimal solution. Therefore that same binary vector  $x$  will also be the optimal solution of the corresponding BLP problem. It is clear that we are using a sufficient condition and not as a necessary condition.

In addition to satisfying the above condition, we will also use LP software with  $0 \leq x \leq 1$  bounded variables condition to produce and therefore verify the same binary solution. In some examples we will see that LPLU will not give the binary solutions, but will give the same optimal value  $z$ . This will happen because condition for LPLU is not binary, it only satisfies bounds. Under this situation we will show that the BLP algorithm can still give the correct binary solution. For some of the problems, we verified the results using direct enumeration of all binary vectors.

It should be realized that this method of problem generation follows an important law of nature, as pointed out by [Dantzig, p. 33]. That is the law of conservation. This law is same as: material balance or law of conservation of mass, energy, force etc. It is also same as Kirchoff's current and voltage laws of electric circuit theory. Thus a BLP problem example may be hypothetical or unrealistic or impractical if it violates the law of conservation. This law is also reflected in the relationship between the primal and the dual variables. Thus the coefficients in  $A$ ,  $b$ , and  $c$  cannot be arbitrarily changed; they are related by this law.

We have chosen non-negativity of elements of  $A$  to keep the computer program simpler. Our objective was not to venture for creating a commercially robust program. The focus is to provide a high level concept of the algorithm by eliminating complexities arising from negativities of  $A$ ,  $b$ , and  $c$  parameters. The

algorithm, once understood, can be easily customized for all such special cases. As we have seen that the solution of BLP does not depend of the nature of the feasible region and its corner points, which are defined by the  $A$ ,  $b$ , and  $c$  matrices. The solution depends on the existence of corner points of the unit cube inside the feasible region.

It is always possible to add an equation with all positive coefficients for all variables to all the equations to make all  $A$  matrix elements nonnegative. However this may produce right hand side  $b$  vector with negative elements. This kind of  $b$  vector can always be replaced by a linear combination of vectors from the first quadrant. We can then solve all equations separately and then combine them linearly to find the original solution. Thus matrix equations with positive coefficients are really not a constraint on the principle behind the BLP algorithm. We should realize that in nature there are no negative numbers. Negative numbers are products of human imagination, and therefore abstract and artificial.

## 5 Numerical Examples

In the following subsections we provide three numerical examples to illustrate the algorithm and the nature of tables it may require. The problems are:

- (a) An example generated using the method described in test problems ( $m < n$ )
- (b) An OR Library example taken from a website ( $m > n$ )
- (c) A 0-1 Knapsack problem taken from a standard text book ( $m = 1$ )

This exercise is not designed to prove the algorithm. They are provided to illustrate the details of the algorithm. Note that the algorithm is a complicated computer program. It has many variants and adjustments depending on the nature of the problem. All computer programs require customization to make things work for individual problems. These examples are presented to explain such details.



The algorithm is documented at high level or at conceptual level. Even then it may appear cryptic. This is true for all computer programs. More details we add more cryptic the algorithm will look. We have taken a hierarchical and modular approach in describing the algorithm; which is a well known concept in software engineering. This method allows every lower level to be smaller and simpler.

### 5.1 A 3R5C Example

First we illustrate our algorithm, using an example generated by the method described in the previous section on Test Problems. The first three functions were used with five samples to produce the following A matrix. The matrix A has 3 rows and 5 columns, and hence the name 3R5C.

A: 3R5C	1.000000000	0.8138079236	0.2112777074	0.02508563094	0.5125428155
	0.6545084972	0.09549150281	0.09549150281	0.6545084972	1.000000000
	0.4748286925	0.7354202041	0.8784358579	0.9569245129	1.000000000

Initial values for the vectors b and c are given below and are labeled as B0 and C0.

Example 5.1					
B0	2.351436370	2.404508497	3.167173409		
C0	1.034693911	0.8100068609	0.8731983764	1.358575743	1.717000249
X0	{1,1,0,1,1}	z0=		4.920276764	
Y0	0.104971	0.779101	0.884097		

Randomly generated binary solution vector X0, and the associated optimal cost z0 are reproduced in the table. For the sake of completeness, the dual random variable Y0 is also mentioned.

This problem can be solved by the standard LPLU algorithm of the Mathematica Software tool. It gives the same binary result that our problem formulation method selected. Thus the result of the BLP algorithm has been validated, for this example, by the Simplex method also. Our algorithm is best illustrated using tabular forms like in simplex method for LP. The first one, Table-1A, shows the setup of the initial data.

The Table-1A shows the given matrix data placed in a specific form. The problem variable names X1-X5 are displayed in row-3. The last column represents

Table-1A: Setup of given data $x_0 = \{1,1,0,1,1\}$ $z_0 = 4.920$								
0	0	0	0	0	1	1	1	
X1	X2	X3	X4	X5	S1	S2	S3	B0
1.00	0.81	0.21	0.02	0.51	2.35			2.35
0.65	0.09	0.09	0.65	1.00		2.40		2.40
0.47	0.73	0.87	0.95	1.00			3.16	3.16
1.03	0.81	0.87	1.35	1.71	0	0	0	C0

Example 5.1					
Table-1B: Ratio Test Matrix $\{r_{ij}\}$					
X1	X2	X3	X4	X5	B0
2.35	2.88	11.1	93.7	4.58	2.35
3.67	25.1	25.1	3.67	2.40	2.40
6.67	4.30	3.60	3.30	3.16	3.16

the given B0 vector. The elements of the B0 vector are also placed in a diagonal matrix, as slack variable coefficients, labeled S1-S3. The last row represents the cost vector C0. The given optimal solution is recorded in row-1, for the purpose of monitoring the status of iterations of our algorithm. Initially all problem variables are set to 0 and all slack variables are set to 1. These are shown in row-2. At this time, in Table-1A, all equations are satisfied; and therefore we have a feasible solution.

At each iteration we create a matrix  $\{r_{ij}\}$ , called Ratio Test Matrix (RTM), an idea borrowed from the Simplex method, to determine the feasibility of all possible solutions. This matrix is shown in Table-1B. The column-X1 of Table-1B is obtained by dividing corresponding elements of B0 by the first column of the A

Table-1C: Correlation Matrix $\{R_{ij}\}$							
		X1	X2	X3	X4	X5	B0
1	$X_e/Y_e$	1.08	0.70	0.39	0.65	1.15	
2	0.10	0.11	.073	.041	.069	0.12	2.35
3	0.77	0.84	0.54	0.30	0.51	0.90	2.40
4	0.88	0.96	0.61	0.35	0.58	1.02	3.16
5	C0	1.03	0.81	0.87	1.35	1.71	
	a	b	c	d	e	f	g

Example 5.1						
Table-1D: Reduced Cost Row $\{rC_j\}$						
		X1	X2	X3	X4	X5
1	Col-Max	0.96	0.61	0.35	0.58	1.02
2	C0	1.03	0.81	0.87	1.35	1.71
3	Product	0.99	0.50	0.30	0.79	1.75
4	Max-loc					x
	a	b	c	d	e	f

matrix. Thus the element in the first location of column-X1 is obtained from  $2.3514/1.000$ . As long as all elements of the RTM are greater than 1, the problem is considered as feasible. If any entry becomes less than 1, then that variable cannot be set to 1, and therefore its value must be 0. If a value becomes exactly equal to 1, then more advanced logic can be used to decide the value of the corresponding column variable. In this paper we ignore that step. As we can see, from Table-1B, that the problem is feasible at this iteration.

Next we compute the pseudo inverse of the A matrix to estimate the primal variable X and the dual variable Y. We get them from the following relations:

$$X_e = A^-B_0 \quad \text{and} \quad Y_e = A^{-T}C_0 \quad (9)$$

In (9)  $X_e$  and  $Y_e$  are the estimates for X and Y respectively;  $A^-$  and  $A^{-T}$  are the pseudo inverse and its transpose for the matrix A. As we show below, in each iteration A matrix will change in size, but not in values, we have to estimate X and Y in all iterations. Therefore we have to compute  $A^-$  also, for each iteration. In each iteration we create a correlation matrix, called  $\{R_{ij}\}$ , by using the following relations:

$$R_{ij} = Abs[X_{ej} * Y_{ei}] \quad i = 1, \dots, m, \quad j = 1, \dots, n \quad (10)$$

The following is a little description of the Table-1C. We hope, to an attentive reader it will be self evident. Row-1 represents the estimate of X vector; and Column-a for the estimate of the Y vector. The shaded matrix is the corresponding correlations. Row-5 is the original cost vector C0 and Column-g is the original RHS vector B0. From Table-1C we create our next Table-1D, called the reduced cost row, rC.

The elements in Row-1 of Table-1D are the maximum of corresponding columns of the Correlation Matrix R given in the previous Table-1C. Each row element is multiplied by the corresponding cost vector elements of C0. The C0 is repeated in Row-2 and the product is given in Row-3. The maximum of Row-3 is marked by x in Row-4. Thus X5 will be assigned as 1 for its value. All other variables still remain 0. Observe that, for the initial step, the maximum of C0 and Row-3 are at same location in Column-f. However, this will not be so in all iterations. This ends the first iteration. In each iteration we decide a value for at least one variable. Thus the algorithm will end at most after n iterations. Note that in this step this is an optimal CSL, because this gives the highest value for the reduced cost, just like in the standard Simplex method.

The next iteration is initialized in the following way and is shown in Table-2A. Since we have selected X5 as 1, we subtract elements of this column from the corresponding diagonal elements of the artificial variables. The values for the artificial variables always remain as 1. The column for X5 is removed from the Table-2A; the cost element for X5 is set to 0, for the data base management information. We repeat the C0 row and call it Ci for the iteration. Table-2A now takes the new form as shown.

Our next step will be to create the Ratio Test Matrix and present it as Table-2B. This time however, the ratios will be created using the modified diagonal

Table-2A: Iteration 2, Initial setup. $X_0 = \{1,1,0,1,1\}$ $Z_0 = 4.92027$								
0	0	0	0	1	1	1	1	
X1	X2	X3	X4	X5	S1	S2	S3	B
1.00	0.81	0.21	0.02		1.83			2.35
0.65	0.09	0.09	0.65			1.40		2.40
0.47	0.73	0.87	0.95				2.16	3.16
1.03	0.81	0.87	1.35	0				Ci
1.03	0.81	0.87	1.35	1.71	0	0	0	C0

Example 5.1						
Table-2B: Ratio Test Matrix $\{r_{ij}\}$						
X1	X2	X3	X4	X5	Bi	
1.8	2.2	8.7	73.		1.83	
3	5	0	3			
2.1	14.	14.	2.1		1.40	
4	7	7	4			
4.5	2.9	2.4	2.2		2.16	
6	4	6	6			

elements of the artificial variables. We have represented the diagonal elements of the artificial variable coefficients as column in Bi. In Table-2B there are no entries with values less than 1. So we ignore this step. Next we find the pseudo inverse of the matrix in Table-2A and the corresponding estimates of primal and dual variables using Bi and Ci values shown in Table-2C. Then we compute the correlation matrix R as shown in Table-2C.

Once we have the correlation matrix, we can then create the reduced cost

Table-2C: Correlation Matrix $\{R_{ij}\}$							
	X1	X2	X3	X4	X5	Bi	
$X_e/Y_e$	1.19	0.66	0.39	0.79			
0.104	0.12	0.07	0.04	0.08		1.83	
0.779	0.92	0.52	0.30	0.62		1.40	
0.884	1.05	0.59	0.34	0.70		2.16	
Ci	1.03	0.81	0.87	1.35	0		
C0	1.03	0.81	0.87	1.35	1.71		

Example 5.1					
Table-2D: Reduced Cost Row					
		X1	X2	X3	X4
1	Col-Max	1.05	0.59	0.34	0.70
2	Ci	1.03	0.81	0.87	1.35
3	Product	1.08	0.47	0.30	0.96
4	Max-loc	x			
	a	b	c	d	e

Example 5.1: Table-3A: Iteration 3, $X_0 = \{1,1,0,1,1\}$ , $Z_0 = 4.920$						
1	0	0	0	1	1	
X1	X2	X3	X4	X5	Bi	B0
	0.81380	0.21127	0.02508		0.83889	2.35143
	0.09549	0.09549	0.65450		0.75000	2.40450
	0.73542	0.87843	0.95692		1.6923	3.16717
0	0.81000	0.87319	1.35857	0		Ci
1.03469	0.81000	0.87319	1.35857	1.71700	0	C0

row. This is shown in Table-2D. Row-3 is the product of corresponding elements of Row-1 and Row-2 of Table-2D. Note that this time  $C_i$  max, which is in column-e, has not been selected by the BLP algorithm. The  $X_1$  variable will now have value 1 and will be eliminated from the process in the next iteration.

Since we are not using the artificial variables as diagonal elements, we can represent them as a column vector and call it  $B_i$ , the iterations of  $B_0$ . This will help to reduce the size of the Table-3A as shown.

The next step is to create the pseudo inverse of the matrix in Table-3A. This  $A$  is a  $3 \times 3$  matrix. We also compute estimates of the primal and the dual variables using  $B_i$  and  $C_i$  values shown in Table-3A. However now we get an interesting result. The estimate for the primal variables becomes all binary, and is given below:

$$X_e = \{1.0000, 0, 1.0000\}$$

Which gives  $X_2=1$ ,  $X_3=0$ , and  $X_4=1$ . Thus our process ends here and our results match with the original values we selected during the problem design phase. The final  $Z_0$  value can be computed using the cost values. In many other problems, the process steps can be reduced by the ratio test matrix also. We will present an example later for this situation.

Summarizing, we can see that the algorithm contains the following steps for each iteration. More details will be given later, after we illustrate with more examples.

- (a) Create the ratio test matrix to find if any element is less than 1. This will help to set those column variables to 0.
- (b) Create pseudo inverse of the  $A$  matrix, estimate the primal and dual variables, and create a correlation matrix using the estimates. This step may end the process, if it gives binary values for  $X_e$ .
- (c) Create a max row by selecting maximal elements of each column from correlation matrix. Multiply this row by the corresponding cost values from  $C_0$ . We call it the reduce cost row.

OR-Lib Prob. Matrix A0					
8	12	13	64	22	41
8	12	13	75	22	41
3	6	4	18	6	4
5	10	8	32	6	12
5	13	8	42	6	20
5	13	8	48	6	20
0	0	0	0	8	0
3	0	4	0	8	0
3	2	4	0	8	4
3	2	4	8	8	4

- (d) Select the max element from this reduced cost row. This selected column variable will have a value of 1.
- (e) Update the tables for next iteration, by subtracting the selected column of the A matrix from B0 vector. Set the cost element to 0 corresponding to the selected variables. Remove the column from the A matrix.
- (f) Continue as above from step (a) until we process all n variables.

## 5.2 An OR Library Problem

The following problem is taken from the operations research (OR) database library [Beasley]. In the problem the matrix A has more number of rows than columns. The matrices for the problem are given by the following tables. Notice that it is a less than equal to problem, as oppose to R3C5 problem, which had equality only. The optimal solution vector  $x_0$  and the optimal cost  $z_0$  are also given. The problem is stated as

Example 5.2: OR-Lib Problem, Initial given data										
B0	80	96	20	36	44	48	10	18	22	24
C0	100	600	1200	2400	500	2000				
$x_0 = \{0, 1, 1, 0, 0, 1\}$ , $z_0 = 3800$										

$$\text{Max } z = c^T x, \quad Ax \leq b, \quad x \in \{0,1\}^n, \quad n = 6$$

The LPLU approach of Mathematica fails to solve this problem. But the BLP algorithm presented gives the correct solution. The solution has been verified using direct enumeration as well.

We present the initial data in the following way in Table-1A. Like in the case of R3C5 problem, we collapse the slack variables as a column and call it Bi vector. We also retain the original B0 at each step, to track the status. First we create a ratio test matrix RTM and present it in Table-1B. Each entry in this table is a ratio of the following type, using a simplified notation, hopefully will not be confusing to the reader.

$$r_{ij} = \frac{b_i}{a_{ij}}, \quad a_{ij} \neq 0 \quad (11)$$

If any element in the RTM is less than 1 then that column variable must be

Table-1A: Given database				X0=011001, Z0=3800			
0	0	0	0	0	0	1	
X1	X2	X3	X4	X5	X6	Bi	B0
8	12	13	64	22	41	80	80
8	12	13	75	22	41	96	96
3	6	4	18	6	4	20	20
5	10	8	32	6	12	36	36
5	13	8	42	6	20	44	44
5	13	8	48	6	20	48	48
0	0	0	0	8	0	10	10
3	0	4	0	8	0	18	18
3	2	4	0	8	4	22	22
3	2	4	8	8	4	24	24
100	600	1200	2400	500	2000		

Example 5.2						
Table-1B: Ratio Test Matrix RTM						
	X1	X2	X3	X4	X5	X6
1	10	6.67	6.15	1.25	3.63	1.95
2	12	8	7.38	1.28	4.36	2.34
3	6.67	3.33	5	1.11	3.33	5
4	7.2	3.6	4.5	1.12	6	3
5	8.8	3.84	5.5	1.05	7.33	2.2
6	9.6	3.69	6	1	8	2.4
7	X	X	X	X	1.25	X
8	6	X	4.5	X	2.25	X
9	7.33	11	5.5	X	2.75	5.5
10	8	12	6	3	3	6

assigned a 0 value. However, if an element is 1 then a complex logic will have to be used.

We can see that the RTM matrix in Table-1B has an entry of 1 in column X4 and Row-6. Since all numbers in Row-6 are nonzero, therefore X4 cannot be assigned a value of 1, because then no other variables can be assigned 1, and the algorithm will terminate. So at this step we decide to assign a 0 value to X4. This assignment does not affect Bi vector. The iteration step ends here, because we have selected one variable.



Example 5.2: Table-2A: Iteration 2, Initial matrix setup, X0=011001, Z0=3800																
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1		
X1	X2	X3	X4	X5	X6	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	B0
8	12	13		22	41	80										80
8	12	13		22	41		96									96
3	6	4		6	4			20								20
5	10	8		6	12				36							36
5	13	8		6	20					44						44
5	13	8		6	20						48					48
0	0	0		8	0							10				10
3	0	4		8	0								18			18
3	2	4		8	4									22		22
3	2	4		8	4										24	24
100	600	1200	0	500	2000	0	0	0	0	0	0	0	0	0	0	Ci
100	600	1200	2400	500	2000	0	0	0	0	0	0	0	0	0	0	C0

Next we create the Table-2A as shown. For this specific problem, we use two different matrices for pseudo inverses. To explicitly show them we present the diagonal form again. To take the pseudo inverse and estimate the primal variable  $X_e$  we consider the entire matrix including the diagonal elements, but eliminate column X4. This will produce an estimate with 15 elements in the vector. We then chop the last 10 elements, and consider the remaining 5 elements to create the

Example 5.2							
Table-2B: Correlation Matrix (R), X0=011001, Z0=3800							
	X1	X2	X3	X4	X5	X6	Bi
$X_e/Y_e$	0.35	0.45	0.50		0.75	0.16	
37.864	13.2	17.3	19.25		28.4	6.33	80
37.864	13.2	17.3	19.25		28.4	6.33	96
-701.87	246.3	320.8	356.8		527.5	117.4	20
1048.7	368.1	479.4	533.2		788.2	175.4	36
-190.63	66.91	87.15	96.92		143.2	31.88	44
-190.63	66.91	87.15	96.92		143.2	31.88	48
459.24	161.2	209.9	233.5		345.1	76.82	10
235.05	82.51	107.4	119.5		176.6	39.31	18
-407.12	142.9	186.1	207.0		306.0	68.10	22
-407.12	142.9	186.1	207.0		306.0	68.10	24
Ci	100	600	1200	0	500	2000	
C0	100	600	1200	2400	500	2000	

correlation matrix R. However, to estimate the dual variables, we take only the 5 columns of the A matrix and the 5 elements of  $C_i$  cost vector. Observe that, as shown in the table,  $C_i$  has one item as 0, corresponding to the X4 column.

Extended  $X_e =$

{0.35104,0.45719,0.50846,0.94987,0.75161,0.16728,0.30606,0.27321,0.30241,0.26141,0.32296,0.24011,0.40634,0.33606,0.39139}

Reduced  $X_e = \{0.35104, 0.45719, 0.50846, 0.75161, 0.16728\}$

$Y_e = \{37.864, 37.864, -701.87, 1048.7, -190.63, -190.63, 459.24, 235.05, -407.12, -407.12\}$

In our computer program we carefully keep track of deleted variables, matrix columns, and cost rows. We maintain one tag for all original variables, and

Example 5.2						
Table-2C: Reduced Cost Row, $X_0=011001$ , $Z_0=3800$						
	X1	X2	X3	X4	X5	X6
Col-Max	368.15	479.47	533.24		788.25	175.43
C0	100.00	600.00	1200.0	2400	500.00	2000.0
Product	36815.	$2.87 \times 10^5$	$6.39 \times 10^5$		$3.94 \times 10^5$	$3.50 \times 10^5$
Max-loc			x			

another one for variables remaining to be processed. The correlation matrix for this iteration has the data shown in Table-2B; we take the absolute value of each product. From this correlation matrix we can then create the reduced cost row, by selecting the maximum element from each column and multiplying them by the corresponding cost values. This is shown in the Table-2C. The Table-2C shows that max cost is at column X3. Therefore we assign 1 to variable X3. This will end this step and we enter the next iteration beginning with Table-3A.

Since the variable X3 was 1, the X3 column must be subtracted from the diagonal elements corresponding to the slack variables. We do not show the table in the diagonal form to reduce paper space; but we process in diagonal form for the determination of the estimates of primal and the dual variables and for the generation of the reduced cost row. Table-3A shows the initial setup for this iteration.

Next we show the Ratio Test Matrix, RTM. However this matrix does not contribute anything new this time, because none of the elements is less than or equal to 1. The RTM matrix is shown in Table-3B.

Table-3A: Iteration 3, X0=011001, Z0=3800							
0	0	1	0	0	0	1	
X1	X2	X3	X4	X5	X6	Bi	B0
8	12			22	41	67	80
8	12			22	41	83	96
3	6			6	4	16	20
5	10			6	12	28	36
5	13			6	20	36	44
5	13			6	20	40	48
0	0			8	0	10	10
3	0			8	0	14	18
3	2			8	4	18	22
3	2			8	4	20	24
100	600	0	0	500	2000		Ci
100	600	1200	2400	500	2000		C0

Example 5.2						
Table-3B: Ratio Test Matrix (RTM)						
	X1	X2	X3	X4	X5	X6
1	8.37	5.58			3.04	1.63
2	10.3	6.91			3.77	2.02
3	5.33	2.66			2.66	4.00
4	5.60	2.80			4.66	2.33
5	7.20	2.76			6.00	1.80
6	8.00	3.07			6.66	2.00
7	0	0			1.25	0
8	4.66	0			1.75	0
9	6	9			2.25	4.5
10	6.67	10			2.5	5.0

Table-3C: Correlation Matrix (R), X0=011001, Z0=3800

	X1	X2	X3	X4	X5	X6	Bi
$X_e/Y_e$	0.36	0.45			0.09	0.27	
22.92	8.38	10.5			2.25	6.24	67
22.92	8.38	10.5			2.25	6.24	83
-5.67	2.07	2.59			0.55	1.54	16
-18.7	6.84	8.57			1.84	5.09	28
14.53	5.32	6.66			1.43	3.95	36
14.53	5.32	6.66			1.43	3.95	40
33.42	12.2	15.3			3.29	9.10	10
-47.0	17.2	21.5			4.63	12.8	14
-26.7	9.78	12.2			2.63	7.27	18
-26.7	9.78	12.2			2.63	7.27	20
Ci	100	600	0	0	500	2000	
C0	100	600	1200	2400	500	2000	

Example 5.2: Table-3D

Reduced Cost Row, X0=011001, Z0=3800

	X1	X2	X5	X6
Col-Max	17.21	21.5	4.63	12.8
C0	100.0	600.	500.	2000.
Product	1721.	12932.	2315.	25613.
Max-loc				x

Next we generate the correlation matrix. Here again, as mentioned before, we use both extended matrix and short matrix to estimate  $X_e$  and  $Y_e$ . We do not

Table-4A: Iteration 4, X0=011001, Z0=3800

0	0	1	0	0	1	1	
X1	X2	X3	X4	X5	X6	Bi	B0
8	12			22		26	80
8	12			22		42	96
3	6			6		12	20
5	10			6		16	36
5	13			6		16	44
5	13			6		20	48
0	0			8		10	10
3	0			8		14	18
3	2			8		14	22
3	2			8		16	24
100	600	0	0	500	0		Ci
100	600	1200	2400	500	2000		C0

Example 5.2

Table-4B: Ratio Test Matrix RTM

X1	X2	X3	X4	X5	X6
3.25	2.16			1.18	
5.25	3.5			1.90	
4.00	2.00			2.00	
3.20	1.60			2.67	
3.20	1.23			2.67	
4.00	1.53			3.33	
0	0			1.25	
4.67	0			1.75	
4.67	7.00			1.75	
5.33	8.00			2.00	

write the details, as they have been explained in the previous iteration, we just present the contents in Table-3C.

Extended  $X_e =$

{0.36598,0.45819,0.97238,0.74599,0.098442,0.27224,0.20841,0.24293,0.20721,0.28649,0.22209,0.36593,0.29015,0.36113}

Once the matrix R is known we can determine the reduced cost matrix, which is shown in Table-3D. The Table-3D selects the variable X6, and we assign a value of 1 to it. Thus in the next iteration column-6 gets eliminated. For the next

Table-5A: Iteration 5				X0=011001, Z0=3800			
	0	0	1	0	0	1	1
	X1	X2	X3	X4	X5	X6	Bi
	8				22		14
	8				22		30
	3				6		6
	5				6		6
	5				6		3
	5				6		7
	0				8		10
	3				8		14
	3				8		12
	3				8		14
Ci	100	0	0	0	500	0	
C0	100	600	1200	2400	500	2000	

Example 5.2						
Table-5B: Ratio Test Matrix RTM						
	X1	X2	X3	X4	X5	X6
1	1.75				0.63	
2	3.75				1.36	
3	2.0				1.0	
4	1.2				1.0	
5	0.60				0.50	
6	1.40				1.16	
7	0				1.25	
8	4.67				1.75	
9	4.0				1.50	
10	4.67				1.75	

iterations we present all the tables sequentially with no or minimal explanations.

Extended  $X_e = \{0.39387,0.51991,0.97459,-$

0.18580,0.26593,0.15428,0.18650,0.089018,0.27121,0.22033,0.35869,0.28442,0.37387}

Iteration 4, assigns 1 to the variable X2, as shown in Table-4D. Iteration 5 is

Table-4C: Correlation Matrix RXY							
	X1	X2	X3	X4	X5	X6	Bi
$X_e/Y_e$	0.39	0.51			0.26		
13.94	5.49	7.24			3.70		26
13.94	5.49	7.24			3.70		42
6.622	2.60	3.44			1.76		12
-12.7	5.02	6.63			3.39		16
16.80	6.61	8.73			4.46		16
16.80	6.61	8.73			4.46		20
47.60	18.7	24.7			12.6		10
-40.6	15.9	21.1			10.7		14
-20.8	8.22	10.8			5.55		14
-20.8	8.22	10.8			5.55		16
Ci	100	600	0	0	500	0	
C0	100	600	1200	2400	500	2000	

Example 5.2			
Table-4D: Reduced Cost Row			
	X1	X2	X5
Col-Max	18.748	24.748	12.658
C0	100.00	600.00	500.00
Product	1874.8	14849.	6329.1
Max-loc		x	

somewhat interesting; it begins with X1 and X5 only, as shown in Table-5A. We can see from RTM that both columns of Table-5B have elements which are less than 1. Therefore both variables for those columns must be assigned 0 values. This terminates the process and we have found the matching optimal solution.

### 5.3 A 0-1 Knapsack Problem

The following problem is taken from the book [Martello and Toth, p. 21]. It has 8 variables, one linear inequality constraint, and one linear objective function. Martello-Toth defines, using their notations, the problem in the following way:

$$\begin{aligned} \text{Maximize} \quad & z = \sum_{j=1}^n p_j x_j \\ \text{Subject to} \quad & \sum_{j=1}^n w_j x_j \leq c, \quad x_j = 0 \text{ or } 1, \quad j \in N = \{1, \dots, n\} \end{aligned}$$

The data for the problem is given by:

$$(p_j) = (15, 100, 90, 60, 40, 15, 10, 1), \quad (w_j) = (2, 20, 20, 30, 40, 30, 60, 10), \quad c = 102$$

The optimal solution, of value 280, is  $(x_j) = (1, 1, 1, 1, 0, 1, 0, 0)$

The purpose of this example is to illustrate how the algorithm works for single row cases. The objective is not to solve large problems or problems with complex characteristics. Computer programs can always be modified to solve any problem with special cases as long as the theory is consistent. In this problem we do not use the pseudo inverse method, although it can be used, and we have used it for this problem successfully. Instead we use the ratio test matrix and derive our reduced cost from this data. In iteration one, we begin with Table-1.

Example 5.3											
Table-1: Given database & Ratio Test Matrix					X0=11110100, Z0=280						
1		0	0	0	0	0	0	0	0	1	
2		X1	X2	X3	X4	X5	X6	X7	X8	Bi	B0
3	A	2	20	20	30	40	30	60	10	102	102
4	Ratio Test	51	5.1	5.1	3.4	2.55	3.4	1.7	10.2		
5	Reduced Cost	765	510	459	204	102	51	17	10.2		
6	Max	x									
7	Ci	15	100	90	60	40	15	10	1		
8	C0	15	100	90	60	40	15	10	1		

In Table-1 row 3 contains the A matrix as defined in BLP; row 8 is the cost vector C; column B0 is the b vector; X1..X8 are the problem variables. In the first initial table we have repeated the cost Ci as C0. In the same way we have repeated b vector as B0 and Bi. Row 4 shows the ratio test matrix (RTM), in this case it is a row only. It is created using the relation  $B_i/A_j$ . Thus element at row-4 and col-X1 is 51 and is obtained from  $102/2$ . We then multiply ratio test row by the corresponding cost values on row-7, which is a repeat of row-8. This generates the reduced cost shown in row-5. We then identify the maximum value location, marked as x, at column X1. Thus we assign 1 to the variable X1. This ends the first iteration. Next, update the database by copying the Table-1 as Table-2 as shown.

Example 5.3											
Table-2: Given database & Ratio Test Matrix					X0=11110100, Z0=280						
1		1	0	0	0	0	0	0	0	1	
2		X1	X2	X3	X4	X5	X6	X7	X8	Bi	B0
3	A		20	20	30	40	30	60	10	100	102
4	Ratio Test		5	5	3.33	2.5	3.33	1.67	10		
5	Reduced Cost		500	450	200	100	50	16.6	10		
6	Max		x								
7	Ci	0	100	90	60	40	15	10	1		
8	C0	15	100	90	60	40	15	10	1		

In Table-2, we indicate 1 at row-1 and column-X1; set 0 for  $C_i$  at row-7 column-X1; and clear remaining parts of column-X1. Also we subtract A matrix coefficients from  $B_i$ , which now becomes 100. Then we fill the ratio test row by dividing 100 by the corresponding row-3 values for A matrix. To derive reduced cost we multiply ratio test row by the corresponding  $C_i$  elements. The maximum of the reduced cost is now located at X2 as shown by the mark x. Thus X2 becomes 1 now. In the next iteration we repeat the process as shown below.

Example 5.3											
Table-3: Given database & Ratio Test Matrix					X0=11110100, Z0=280						
1		1	1	0	0	0	0	0	0	1	
2		X1	X2	X3	X4	X5	X6	X7	X8	$B_i$	$B_0$
3	A			20	30	40	30	60	10	80	102
4	Ratio Test			4	2.67	2	2.67	1.33	8		
5	Reduced Cost			360	160	80	40	13.3	8		
6	Max			x							
7	$C_i$	0	0	90	60	40	15	10	1		
8	$C_0$	15	100	90	60	40	15	10	1		

Example 5.3											
Table-4: Given database & Ratio Test Matrix					X0=11110100, Z0=280						
1		1	1	1	1	0	0	0	0	1	
2		X1	X2	X3	X4	X5	X6	X7	X8	$B_i$	$B_0$
3	A					40	30	60	10	30	102
4	Ratio Test					0.75	1.0	0.50	3.0		
5	Reduced Cost										
6	Max										
7	$C_i$	0	0	0	0	40	15	10	1		
8	$C_0$	15	100	90	60	40	15	10	1		



From Table-3 we see that the variable  $X_3$  becomes 1. The next we reproduce Table-4. Table-4 represents the fourth iteration. We compute ratio test and observe that in columns  $X_5$  and  $X_7$  the values are less than 1. Thus these two variables must be set to 0. We do not need to compute the reduced cost for this fourth iteration in Table-4. The next step is to eliminate  $X_5$  and  $X_7$  columns. The result is shown in Table-5.

Example 5.3											
Table-5: Given database & Ratio Test Matrix					$X_0=11110100, \quad Z_0=280$						
1		1	1	1	1	0	1	0	0	1	
2		$X_1$	$X_2$	$X_3$	$X_4$	$X_5$	$X_6$	$X_7$	$X_8$	$B_i$	$B_0$
3	A						30		10	0	102
4	Ratio Test						1.0		3.0		
5	Reduced Cost										
6	Max										
7	$C_i$	0	0	0	0	0	15	0	1		
8	$C_0$	15	100	90	60	40	15	10	1		

The ratio test now shows that  $X_6$  has a value of 1.0. If we set this variable as 1 then the process will stop because no other variables can be 1 anymore. Comparing the cost in row-7 we choose 1 for  $X_6$  and 0 for  $X_8$ . We have found the optimal solution. Notice that in this case  $B_i$  has 0 value. In many problems the norm of  $B_i$  can be used to terminate the algorithm. In equality problems, with  $B_0$  in binary range space  $\mathcal{B}$ , final value of  $B_i$  must be 0, as in this problem.

## 6 Conclusion

A polynomial time algorithm for binary linear programs is presented. The algorithm and its theoretical foundations are clearly described. Different types of numerical examples are solved to explain the details of the theory and its features, variants, and capabilities.

**ACKNOWLEDGEMENTS.** Thanks to Professor Sukumar Sikdar for pointing out this problem to the author as a research problem.

## References

- [1] R J Vanderbei, *Linear Programming: Foundations and Extensions*, Second edition, Department of operations research and financial engineering, Princeton University, Princeton, NJ 08544, 2001, available free from: [http://support.dce.felk.cvut.cz/pub/hanzalek/\\_private/ref/Vanderbei\\_Linear\\_Programming.pdf](http://support.dce.felk.cvut.cz/pub/hanzalek/_private/ref/Vanderbei_Linear_Programming.pdf)
- [2] S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani, *Algorithms*, UC Berkeley, California, 2006, 318 pages, available free from: <http://www.cs.berkeley.edu/~vazirani/algorithms/>
- [3] D.G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, Third Edition, Stanford University, 551 Pages, available free from: <http://grapr.files.wordpress.com/2011/09/luenberger-linear-and-nonlinear-programming-3e-springer-2008.pdf>
- [4] G.B. Dantzig, *Linear programming and extensions, R-366-PR, Part 1*, Rand Corporation, California, 1963, 219 Pages. Available free from: <http://www.rand.org/pubs/reports/R366.html>
- [5] S. Martello and P. Toth, *Knapsack Problems, algorithms and computer implementations*, DEIS, University of Bologna, John Wiley, NY, 1990, 306 Pages, Available free from: <http://www.or.deis.unibo.it/knapsack.html>
- [6] S. Das, Binary solutions for overdetermined systems of linear equations, *AMO*, v14, n1, 2012.
- [7] J.E. Beasley, OR library, Imperial college, UK, can be accessed from: <http://people.brunel.ac.uk/~mastjjb/>

- [8] J. Trahan, A. Kaw, and K. Martin, Computational Time for Finding the Inverse of a Matrix: LU Decomposition vs. Naive Gaussian Elimination, University of South Florida, United States of America, Access it from: [http://mathforcollege.com/nm/simulations/nbm/04sle/nbm\\_sle\\_sim\\_inversecomptime.pdf](http://mathforcollege.com/nm/simulations/nbm/04sle/nbm_sle_sim_inversecomptime.pdf)
- [9] J. Farlow, et al., *Differential equations & linear algebra*, Prentice hall, New Jersey, US, 2002.