FEASIBILITY ACHIEVEMENT WITHOUT THE HASSLE OF ARTIFICIAL VARIABLES: A COMPUTATIONAL STUDY

Syed Inayatullah[1], Nasir Touheed[2], Muhammad Imtiaz[3] , Tanveer Ahmed Siddiqi[4], Saba Naz[5]

[1,3,4,5] Department of Mathematics, University of Karachi, Karachi, Pakistan. 75270
[2] Department of Mathematical Sciences, Institute of Business Administration, Karachi, Pakistan.75270

Abstract

The purpose of this article is to encourage students and teachers to use a simple technique for finding feasible solution of an LP. This technique is very simple but unfortunately not much participated in the textbook literature yet. This article discusses an overview, advantages, computational experience of the method. This method provides some pronounced benefits over Dantzig's simplex method phase 1. For instance, it does not require any kind of artificial variables or artificial constraints; it could directly start with any infeasible basis of an LP. Throughout the procedure it works in original variables space hence revealing the true underlying geometry of the problem. Last but not the least; it is a handy tool for students to quickly solve a linear programming problem without indulging with artificial variables. It is also beneficial for the teachers who want to teach feasibility achievement as a separate topic before teaching optimality achievement. Our primary result shows that this method is much better than simplex phase 1 for practical Net-lib problems as well as for general random LPs.

Keywords: Linear Programming, Simplex Method, Artificial Variables, Dictionary, Phase 1

1. Introduction

Linear programs frequently show up in various areas of applied sciences today. The prime reason for this is their tractability: linear programs frame problems in optimization as a system of linear inequalities. This template is general enough to express many different problems in engineering, operations research, economics, and even more abstract mathematical areas such as combinatorics.

The linear programming problem is usually solved by incorporating one of the two algorithms: either with simplex algorithm or interior point algorithm. Both of the methods are extensively used these days and continue to contest with each other. They have their own advantages and disadvantages in different contexts. For example interior point method has a big advantage that it has a polynomial time complexity to solve a general LP problem. But the Dantzig's simplex method still seems to be the most efficient algorithm for great majority of practical problems because most of the practical problems are not "very large" when interior point becomes more efficient. Simplex method also exhibits efficiency when a problem needs to be re-solved with certain modifications. The complexity of IPM for a single iteration is $O(n^3)$ while that of simplex method is $O(n^2)$. So when handling large-scale problems even thousands of pivots of simplex method require considerably less effort as compared to a single IPM iteration. Also this factor provides simplex method an edge to perform sensitivity analysis more efficiently. Additionally simplex method so far is superior to IPM for solving (mixed) integer linear optimization problems. One reason contributing to this factor is that generating a cut requires a basic solution.

Considering vast applicability of LPs in various fields, learning LPs has become an important part of undergraduate and graduate courses. Because of this many researchers are now focused in designing algorithms which are more efficient and easily implementable for classroom teaching.

2. Literarture Review:

Essentially the simplex method by Dantzig was developed to solve only the LPs having a known feasible solution, commonly referred as the initial basic feasible solution. For the LPs having no initial basic feasible solution, almost all of the practical variants of simplex method suggest to apply the simplex method in two phases (Dantzig, Orden, & Wolfe, 1955) (Wagner, 1956), called phase 1 and phase 2. In phase 1, a basic feasible solution is created by adding some (non-negative) artificial variables to the problem with an additional objective, equal to minimization of the sum of all the artificial variables, called SP1 objective. The purpose of phase 1 process is to maintain the feasibility and minimize the sum of artificial variables as much as possible. If phase 1 terminates with an objective value equal to zero, it implies that all artificial variables have reached value zero and the current basis has become feasible to the original problem, then return to the original objective and proceed with simplex phase 2. Otherwise, conclude that the problem has no solution.

For a bit larger LPs, generally implementation of two phase simplex method significantly increases the number of variables, number of iterations and thus the complexity as well. From the point of view of class room teaching it often becomes a tedious job too. The above mentioned factors led to the need of developing more general algorithms for solving a given linear program in which one may directly start from an initial infeasible basic solution.

Papparrizos (1990) presented an artificial variable free method but his method uses additional artificial constraint with a big-M number. His method also requires a tedious evaluation of series of additional objective functions besides the original objective function. Moreover at each iteration this algorithm must check both primal and dual feasibility.

Arsham (Arsham H. , 1997) (Arsham H. , 1997) proposed an algorithm for general LP models in which he claimed that his algorithm will either provide a feasible solution or will declare infeasibility after a finite number of iterations. Enge and Huhn (1998) and Inayatullah, Touheed and Imtiaz (2015) presented counter examples in which Arsham's algorithm is declaring a feasible problem inconsistent.

Later on (Arsham, Baloh, Damij, & Grad, 2003) (Arsham, Damij, & Grad, 2003) (Arsham H. , 1989) presented an artificial-free algorithm named push and pull algorithm which initiates with an incomplete basic variable set (BVS). As the algorithm proceeds the variables are brought in to the basis. The Push phase continues until the basic variable set is complete. This phase may terminate yielding an infeasible BVS. The problem then advances by starting the Pull Phase, which pulls the solution back to feasibility by incorporating pivot rules similar to the dual simplex method. Arsham claims that the push and pull algorithm is artificial-free, however, his claim would be correct if we are only concerned with artificial variables, in fact his method requires adding artificial constraints so his method is not truly an artificial free method.

Serang (2012) claimed that simplex method is so far still practically the best known pivot algorithm for solving LPs. But from teaching point of view the hurdle is that simplex method for feasibility (simplex phase 1) cannot be illustrated, prior to simplex method for optimality (simplex phase 2). So, usually students learn phase 2 before phase 1, which of course sounds unpleasant for both teachers and students.

The main method (also discussed in (Pan, 2014)), is an easy to use alternative of simplex phase 1 process which obviates the role of artificial variables by allowing negative variables into the basis. This method is artificial variable/constraint free so consequently avoid stalling and save degenerate pivots in many cases of linear programming problems. In this article we would call this method as *Dynamic Phase 1(DP1)*.

Most recently Inayatullah, Touheed and Imtiaz (2015) constructed another artificial variable free version (may be considered as *clone*) of simplex phase 1, called *Art-free Simplex Method* (ASM). That method is also artificial variable free and as well as artificial constraint free version of simplex phase 1. The authors showed that this method is computationally more efficient than simplex phase 1 because it saves unnecessary computations. The key difference between ASM and DP1 is that ASM follows the same sequences of pivots as simplex phase 1 does, even in the case of highly degenerate LPs whereas DP1 is not.

Computationally DP1 is similar to ASM in the sense that it also saves unnecessary computations, while iterations-wise this DP1 may produce different sequence of pivots from the pivoting sequence of simplex method. The difference would arise only in the problems where degeneracy due to artificial variables occurs. Simplex method (hence also its artificial variable free clone ASM) has a technical flaw that it takes into account degenerate artificial variables in making the phase 1 objective, which is undoubtedly peripheral , because degenerate artificial variables do not actually reflect the infeasibility status of the problem. DP1 overcomes the flaw using the refresh computation of the phase 1 objective in each iteration. DP1, in contrasting phase I simplex method, does not require any abrupt changes in the LP structure to start with. Indeed it could start to avail feasibility at any time without making any adjustments in the simplex table. Unlike (Arsham, Baloh, Damij, & Grad, 2003) (Arsham, Damij, & Grad, 2003) (Arsham H. ,

1989) and (Papparrizos, 1990) , it neither requires any artificial variable nor any artificial constraint.

In this article we have denoted *simplex phase 1* and *simplex phase 2* by SP1 and SP2 respectively, and *dynamic phase 1* by DP1.

3. Advantages of DP1

Several features of DP1 are alike ASM. For instance, it could start with any feasible or infeasible basis of an LP. In fact it could also be very useful for solving integer programming problems. This method solves general LP problem without any need of artificial variables which also makes it space efficient. Usually the learning sequence for simplex method is first SP2 and then SP1, because SP1 requires a prerequisite knowledge of SP2. DP1 would be a fruitful tool for the teachers who want to teach feasibility achievement as a separate topic before teaching optimality achievement, because working rule of this method can easily be demonstrated to the students having either a little or even no prior knowledge of simplex method for optimality (SP2). So after the development of DP1, the learning sequence would be to firstly learn DP1 and then SP2 i.e. it eradicates the need to illustrate SP2 before SP1.

Distinguishing features of DP1 over ASM are, during ASM one may have to store the feasibility status of each variable by '+' and '–' flags. So, ASM is vulnerable to face infeasibility flagged degenerate variables. This issue effects ASM in the same way as degeneracy effects simplex method. Despite DP1 is immune to such kind of pseudo infeasibilities, because it does not need any flags to show the feasibility status of a variable.

3.1 Description of the procedure DP1 through an example:

Consider the following linear system,

$$\begin{aligned}
-11x_1 - 3x_2 - 3x_3 + 2x_4 + x_5 &= -44 \\
-x_1 + x_2 + 5x_3 - x_4 + x_6 &= -20 \\
3x_1 - 4x_2 - 8x_3 - 4x_4 + x_7 &= -24 \\
2x_1 - 4x_2 - 5x_3 - 6x_4 + x_8 &= 60 \\
3x_1 - 4x_2 - 3x_3 - 2x_4 + x_9 &= 12 \\
x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9 &\geq 0
\end{aligned}$$

For the initial basis B, setting B = {5,6,7,8,9}, is the easiest choice. So corresponding N = {1,2,3,4}. Here it could be observed that negative right-hand side values could be used as a tentative infeasibility measure of each constraint for current basis. For example 1st, 2nd and 3rd constraints are infeasible by 44, 20 and 24 units of slack values. In all, current basis is infeasible by 88 units. Let us construct an objective function of minimizing overall infeasibility (negative sum of basic infeasible variables) of the problem. That is *Minimize cis(B)* $= -_5 - x_6 - x_7$. Where *cis*(B) stands for *cumulative infeasibility status* and its value could be obtained for a basis B, by setting all non-basic variables x_N equal to zero. In general $cis(B) = -\sum_{x_i<0, i\in B} x_i$.

On replacing the basic variables in terms of non-basic variables, *cis*(B) becomes

$$Minimize\ \ cis(B) = -9x_1 - 7x_2 - 6x_3 + 88 .$$

The *cis*(B) function is an alternative to traditional phase 1 objective function of minimizing artificial variables. In this function the constant value 88 is unimportant, so to compute the coefficients only, one would have to just vertically sum-up the coefficients of non-basic variables in the constraints with negative right hand side. The coefficient vector of *cis* function is denoted by $\mathbf{w}$, in this case $\mathbf{w} = [-9,-7,-6]$.

1. *Selection of entering variable:*

Like SP1, this method (DP1) also follows a gradient ascent approach that iteratively decreases the value of *cis*, while maintaining feasibility of basic variables. From the expression of objective function it is clear that any increase in values of the non-basic variables x_1,x_2 and x_3 would decrease value of *cis*. So, x_1,x_2 and x_3 are all *candidate entering* variables.

The rules of selecting an entering basic variable among all candidate entering variables are usually known as *Pricing Rules*. So far many pricing rules have been developed for entering variables, some of which are, Dantzig's most negative coefficient rule (Dantzig,1963) steepest edge rule (1977), Devex rule (1973), Minimum angle method (Inayatullah, Khan, Imtiaz, & Khan, 2010), Largest-distance rule (Pan, 2008), Nested Pricing rule (Pan, 2008) Nested largest-distance rule (Pan, 2010).

In this paper we would use Dantzig's most negative coefficient criteria for selection of entering variable. According to this criteria, *preferred entering variable is the variable along which cis* *has highest decreasing rate*. So in the above example, x_1 would be our preferred entering basic variable.

2. *Selection of leaving variable:*

Dantzig's ratio test (DRT) suggests to select the basic variable as leaving that imposes the most stringent upper bound on the increase of the entering variable.

An easier way to identify leaving variable is to examine the ratios of right hand side of the constraints to the corresponding coefficients of the entering variable, x_1, as shown in the following table,

Basic	Constraint coefficients of entering variable x_1	Right hand side value	Non-negative Ratio (or intercept)	Selected ratio	Preferred Leaving variable
x_5	-11	-44	$x_1 = \frac{44}{11} = 4$	4 (min)	x_5
x_6	-1	-20	$x_1 = \frac{20}{1} = 20$		
x_7	+3	-24	(ignored)		
x_8	2	60	$x_1 = \frac{60}{2} = 30$		
x_9	3	66	$x_1 = \frac{66}{3} = 22$		
Note: In the column of ratios we only consider variable with non-negative ratios because leaving variable only with non-negative ratios, restricts the entering variable. Preferred leaving variable is the variable corresponding to minimum of these ratios.					

3. *Pivot operation:*

   In this step we perform elementary row operations to obtain a new equivalent LPP with new basis.

4. *Refreshing the objective function:*

   After each change of basis (pivot operation), if infeasibility is not completely removed, re-compute the $cis(B)$ function.

4. The Dynamic Phase 1 Method: (Pan, 2014)

To formally develop the algorithm we'll use the following dictionary notation originally introduced by (Chvatal, 1983) but used in a slightly different form by (Khan, Inayatullah, Imtiaz, & Khan, 2009).

The dictionary of any LP for a basis *B*, may be element-wise represented in the following collection of equations, denoted by *D*(*B*), which is slightly modified form of (Chvatal, 1983) (Kaluzny, 2001).

(1)
$$D(B) = \left\{ \begin{array}{c} x_i + \sum_{j \in N} \alpha_{ij} x_j = \beta_i,\ i \in B \\ \hdashline \mathit{Maximize}\ z = \sum_{j \in N} \gamma_j x_j + \hat{z} \end{array} \right\}$$

Where β_i is the component of vector $A_B^{-1}\mathbf{b} \in \Re^B$ representing value of the basic variable x_i, α_{ij} is the element of $A_B^{-1}A_N \in \Re^{B \times N}$ denoting the coefficient of the non-basic variable x_j in the equation containing basic variable x_i, γ_j is the component of $(\mathbf{c}_N^T - \mathbf{c}_B^T A_B^{-1} A_N)^T \in \Re^N$ representing the coefficient of non-basic variable x_j in the objective function of the current dictionary, and $\hat{z} = \mathbf{c}_B^T A_B^{-1}\mathbf{b} \in \Re$ is the objective scalar value associated with current basis *B*. A basis *B*(or a dictionary *D*(*B*)) is said to be *feasible* if $\beta_i \geq 0$ for all $i \in B$.

As discussed in the last section, We can formally describe $cis(B)$ as follows,

(2)
$$cis(B) = -\sum_{i:\beta_i < 0} x_i$$

Or equivalently in the form of non-basic variables,

(3)
$$cis(B) = -\sum_{i:\beta_i < 0} \left(\beta_i - \sum_{j \in N} \alpha_{ij} x_j \right)$$

On simplification we get,

(4)
$$cis(B) = \sum_{j \in N} \left(\sum_{i:\beta_i < 0} \alpha_{ij} x_j \right) - \sum_{i:\beta_i < 0} \beta_i = \sum_{j \in N} \left(w_j x_j \right) - \sum_{i:\beta_i < 0} \beta_i$$

where $w_j = \sum_{i:\beta_i < 0} \alpha_{ij}$, $j \in N$. We would call w_j's as components of *cumulative infeasibility vector* (*civ*) **w***(B)*.

Lemma 4.1:

For a basis *B*, if all $w_j(B) \geq 0$ then the associated problem is primal inconsistent.

Proof:

Associated *cis* function for would be,

$$cis(B) = -\sum_{i:\beta_i<0} x_i = \sum_{j\in N}\left(w_j x_j\right) - \sum_{i:\beta_i<0}\beta_i$$

Implies that
$$\sum_{j\in N}\left(w_j x_j\right) + \sum_{i:\beta_i<0} x_i = \sum_{i:\beta_i<0}\beta_i$$

Which is a clearly impossible for any feasible (non-negative) solution x because

$$\sum_{j\in N}\left(w_j x_j\right) + \sum_{i:\beta_i<0} x_i \geq 0 \text{ and } \sum_{i:\beta_i<0}\beta_i < 0. \qquad \blacksquare$$

Problem 1

Given a dictionary $D(B)$, obtain a primal feasible basis if exists.

Algorithm: Dynamic Phase 1 (DP1)

Step 1: Let S be a maximal subset of B such that $S = \{s \mid \beta_s < 0, s \in B\}$. If $S = \varphi$ then basis B is primal feasible. **Exit.**

Step 2: Construct a row-vector $\mathbf{w} \in \Re^N$ such that $w_j = \sum_s \left(\alpha_{sj}\right)$.

Step 3: Let $K \subseteq N$ such that $K = \{j : w_j > 0, j \in N\}$. If $K = \varphi$, according to Lemma 3.1 basis B is primal inconsistent. **Exit.**

Step 4: Choose $k \in K$ such that $w_k \geq w_h \ \forall h \in K$

(Ties could be broken on minimum index)

Step 5: Choose $r \in B$ such that

$$r = \arg\min\left\{\left\{\frac{\beta_i}{\alpha_{ik}} : \beta_i < 0, \alpha_{ik} < 0, i \in B\right\} \cup \left\{\frac{\beta_i}{\alpha_{ik}} : \beta_i \geq 0, \alpha_{ik} > 0, i \in B\right\}\right\}$$

(Ties could be broken on minimum index)

Step 6: Make a pivot on (r,k) ($\Rightarrow$ Set $B := (B \cup \{k\}) \setminus \{r\}$, $N := (N \cup \{r\}) \setminus \{k\}$ and update $D(B)$).

Step 7: Go to Step 1.

Theorem 4.2: The dynamic phase 1 is guaranteed to stop in a finite number of iterations if there is no degeneracy.

Proof: There are only a finite number bases, clearly not more than $\binom{n}{m}$, and every non-degenerate pivot performed according to step 4 and step 5 strictly increase the value of the *cis*(*B*) function. This implies that a basis cannot be encountered twice. ■

Example 1

Obtain a feasible basis of the following system of equations using DP1.

$$\begin{aligned}
20x_1 - x_2 + x_3 + x_4 &= 10 \\
-100x_1 + x_2 - 5x_3 + x_5 &= -50 \\
-x_1 + x_2 - 2x_3 + x_6 &= 2 \\
-60x_1 + 4x_2 - 3x_3 + x_7 &= -30 \\
7x_1 - 79x_2 - 10x_3 + x_8 &= -5
\end{aligned}$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0, x_7 \geq 0, x_8 \geq 0$$

By taking initially $B = \{4,5,6,7,8\}, N = \{1,2,3\}$, we can construct the associated dictionary *D*(*B*) (Khan, Inayatullah, Imtiaz, & Khan, 2009) (because objective function *z* is not given, we omitted *z*-row here) of the above problem as

$$\begin{array}{c|c|ccc}
 & & 1 & 2 & 3 \\
4 & 10 & 20 & -1 & 1 \\
5 & -50 & -100 & 1 & -5 \\
6 & 2 & -1 & 1 & -2 \\
7 & -30 & -60 & 4 & -3 \\
8 & -5 & 7 & -79 & -10
\end{array}$$

Here $\beta_5 < 0, \beta_7 < 0, \beta_8 < 0 \Rightarrow S = \{5,7,8\}$. Clearly, current basis is infeasible.

So, row vector $\mathbf{w}(B) \in \Re^N$ would be

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
\mathbf{w} = & [-153 & -74 & -18]
\end{array}$$

The small number written above each component of **w**-vector is its index number. According to Dantzig's pricing rule described in step 4, we get k = 1 so entering basic variable is x_1 and according to the ratio test rule as described in step 5, we get r = 4 so the leaving basic variable is 'x_4'. Perform the pivot operation on $(4,1)$

$$\begin{array}{c|c|ccc} & \mathbf{b}' & 4 & 2 & 3 \\ \hline 1 & 1/2 & 1/20 & -1/20 & 1/20 \\ 5 & 0 & 5 & -4 & 0 \\ 6 & 5/2 & 1/20 & 19/20 & -39/20 \\ 7 & 0 & 3 & 1 & 0 \\ 8 & -17/2 & -7/20 & -1573/20 & -207/20 \end{array}$$

Iteration 2: Now, $B = \{1,5,6,7,8\}, N = \{4,2,3\}$. As $\beta_8 < 0 \Rightarrow S = \{8\}$. Re-compute $\mathbf{w} \in \Re^N$ for new dictionary.

$$\mathbf{w} = \begin{array}{c} \begin{array}{ccc} 4 & 2 & 3 \end{array} \\ \left[\begin{array}{ccc} -7/20 & -1573/20 & -207/20 \end{array}\right] \end{array}$$

Here $k = 2$ and $r = 7$, perform pivot on $(7, 2)$,

$$\begin{array}{c|c|ccc} & \mathbf{b}' & 4 & 7 & 3 \\ \hline 1 & 1/2 & 1/5 & 1/20 & 1/20 \\ 5 & 0 & 17 & 4 & 0 \\ 6 & 5/2 & -14/5 & -19/20 & -39/20 \\ 2 & 0 & 3 & 1 & 0 \\ 8 & -17/2 & 1178/5 & 1573/20 & -207/20 \end{array}$$

Iteration 3:

Here $\beta_8 < 0 \Rightarrow S = \{8\}$. Re-compute $\mathbf{w}$-vector for this new dictionary.

$$\mathbf{w} = \begin{array}{c} \begin{array}{ccc} 4 & 7 & 3 \end{array} \\ \left[\begin{array}{ccc} 1178/5 & 1573/20 & -207/20 \end{array}\right] \end{array}$$

$k = 3$ and $r = 8$, perform pivot on $(8, 3)$

$$\begin{array}{c|c|ccc} & \mathbf{b}' & 4 & 7 & 8 \\ 1 & 95/207 & 277/207 & 89/207 & 1/207 \\ 5 & 0 & 17 & 4 & 0 \\ 6 & 283/69 & -3256/69 & -1088/69 & -13/69 \\ 2 & 0 & 3 & 1 & 0 \\ 3 & 170/207 & -4712/207 & -1573/207 & -20/207 \end{array}$$

Since $S = \varphi \Rightarrow$ The current basis $B = \{1,5,6,2,3\}$ is primal feasible.

Remark 4.3:

If the above problem would have solved by SP1 or ASM it requires 6 iterations and as shown above if it were solved by DP1 it just needs 3 iterations.

Remark 4.4:

There are some similarities and dissimilarities between SP1 and DP1.

Dissimilarities: SP1 starts with a pseudo (artificial) primal feasible basis and DP1 could start directly from any primal infeasible basis. In the start of every iteration, DP1 freshly computes its objective function *cis* for current basis but on the other hand SP1 keeps tracking its original objective function (SP1 objective function).

Similarities: Both the methods analogously preserve the existing feasibility of basic variables and initiate with an equivalent objective of minimizing cumulative infeasibilities in the basic variables.

Remark 4.5:

DP1 has some advantages over SP1.

- In SP1 underlying geometry of original problem is hidden under superimposed framework of artificial variables but in DP1 it is not.
- SP1 may face a situation in which feasibility is achieved but the method didn't show it up, because of degenerate artificial variables. Despite DP1 doesn't need any artificial variables, so it is invulnerable of such circumstances.
- DP1 gives full freedom to user in the construction of initial basis, for example in an equality constraint, a variable of any sign (positive or negative) having coefficient 1 and zero in other constraints could be taken as basic variable. Unlike SP1 which obliges to

take artificial variables into the initial basis. It means that DP1 gives opportunity to put any original variable from the constraint into the basis without needing to introduce any extra variables (i.e. artificial variables).

Example 2

Consider the following system of inequalities,

$$\begin{aligned} 20x_1 - x_2 + x_3 &\le 10 \\ 50x_1 - 7x_2 + x_3 &\ge 25 \\ -7x_1 - 2x_2 + 10x_3 &\ge 5 \\ x_1 \ge 0,\ x_2 \ge 0, x_3 &\ge 0 \end{aligned}$$

By adding non-negative slack variables x_4, x_5 and x_6 and taking $B = \{4,5,6\}, N = \{1,2,3\}$, we can construct the associated dictionary $D(B)$ (here because objective function z is not given, we omitted z-row) of the above problem as

Initial dictionary:

$$\begin{array}{c} \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c|ccc} \mathbf{b}' & 1 & 2 & 3 \\ \hline 10 & 20 & -1 & 1 \\ -25 & -50 & 7 & -1 \\ -5 & 7 & 2 & -10 \end{array}$$

$$\text{Here } \mathbf{w} = \begin{array}{ccc} 1 & 2 & 3 \\ -43 & 9 & -11 \end{array}$$

Pivot on (4,1)

$$\begin{array}{c} \\ 1 \\ 5 \\ 6 \end{array} \begin{array}{c|ccc} \mathbf{b}' & 4 & 2 & 3 \\ \hline 1/2 & 1/20 & -1/20 & 1/20 \\ 0 & 5/2 & 9/2 & 3/2 \\ -17/2 & -7/20 & 47/20 & -207/20 \end{array}$$

$$\text{Here } \mathbf{w} = \begin{array}{ccc} 4 & 2 & 3 \\ -7/20 & 47/20 & -207/20 \end{array}$$

Pivot on (5,3)

$$\begin{array}{c} \\ 1 \\ 3 \\ 6 \end{array} \begin{array}{c|ccc} \mathbf{b}' & 4 & 2 & 5 \\ \hline 1/2 & -1/30 & -1/5 & -1/30 \\ 0 & 5/3 & 3 & 2/3 \\ -17/2 & 169/10 & 167/5 & 69/10 \end{array}$$

Here $\mathbf{w} = \begin{array}{ccc} 4 & 2 & 5 \\ \left[169/10\right. & 167/5 & \left.69/10\right] \end{array}$

According to step 3, the last **w**-vector shows that the problem is primal inconsistent.

5. Applications

Since DP1 is an efficient alternative to the SP1 as well as to ASM. So it can effectively be incorporated in the solution process of linear programming problems, integer programming problems, sensitivity analysis, parametric programming etc. It can become an essential tool which can directly be employed by researchers in solving various problems of diverse fields like biological sciences and engineering for example: biological sciences (Bruijn, Ketelaars, Gelsema, & Sorber, 1991) (Haefner, 2005), medical sciences (Darvas, 1974) (Pulgarín, Molina, & Alañón, 2002), biochemical sciences (Sun, Fleming, Thiele, & Saunders, 2013)**,** mechanical engineering (Jatau, Datau, & Agyo, 1999) etc.

6. Computational Results

In this section Table 1 and Table 2 present a comparison of the computational results of DP1 algorithm with the SP1. Using random models suggested by Kaluzny (2001) we generated 250 consistent linear programs and 250 inconsistent linear programs with the associated dictionary coefficients γ_j, β_i and α_{ij} chosen randomly from the integer interval $[-50,50]$. The results of both consistent and inconsistent problems exhibit that on average DP1 is better than SP1. For example to determine consistency of LPs of orders 50×50, on average DP1 saved 3.16% and to determine inconsistency of LPs of orders 50×70, on average DP1 saved 5.43% iterations.

For further testing on practical problems, we executed both the algorithms on Net-Lib test problems (The Net-Lib Repository for Linear Problems) which reveal that on most Net-lib problems (e.g. SCTAP2, SCSD8, SCSD6, SCAGR25, SCTAP3 etc.) DP1 is more efficient because number of iterations taken by DP1 much lesser than SP1. The results have been summarized in Table 3.

Table 1: Comparison in terms of average iterations on random consistent LPs

Order	SP1	DP1	% of saved iterations
3x3	1.432	1.432	0.00%
3x5	1.76	1.76	0.00%
3x7	1.796	1.796	0.00%
5x5	2.913	2.821248	3.14%
5x10	3.191	3.1120472	2.46%
7x5	4.177	4.1109268	1.59%
7x10	5.082	5.0063368	1.50%
10x5	6.117	6.052	1.06%
10x10	8.128	8.0029756	1.54%
10x20	7.623	7.44496	2.33%
10x25	7.628	7.4749696	2.00%
15x15	14.407	14.10663	2.08%
15x20	14.359	14.070997	2.00%
20x20	22.167	21.720171	2.02%
20x30	20.164	20.004	0.79%
30x30	41.361	40.325434	2.50%
40x40	66.660	65.727074	1.40%
50x50	102.079	98.849477	3.16%
50x70	87.842	85.170342	3.04%
50x100	60.9154	60.492	0.70%
70x50	172.832	170.84423	1.15%
70x70	187.799	185.66	1.14%
70x100	152.531	147.90194	3.03%
80x80	231.531	227.60356	1.70%
90x90	287.652	286.29242	0.47%
100x100	347.511	341.27314	1.79%

Table 2: Comparison in terms of average iterations on random inconsistent LPs

Order	SP1	DP1	% of saved iterations
3x3	0.96	0.96	0.00%
3x5	1.332	1.332	0.00%
3x7	1.381	1.38	0.00%
5x5	2.533	2.3588768	6.88%
5x10	3.571	3.4343948	3.84%
7x5	3.293	3.044394	7.54%
7x10	5.001	4.8670508	2.67%
10x5	4.236	3.94	6.99%
10x10	7.328	7.0200144	4.21%
10x20	10.076	8.88987	11.77%
10x25	11.465	10.943229	4.55%
15x15	12.884	12.012213	6.76%
15x20	15.643	15.282512	2.30%
20x20	20.385	20.057307	1.61%
20x30	27.136	26.868	0.99%
30x30	41.398	40.376639	2.47%
40x40	66.911	65.922074	1.48%
50x50	94.007	91.020514	3.18%
50x70	125.203	118.40621	5.43%
50x100	136.697	131.508	3.80%
70x50	108.923	107.294	1.50%
70x70	174.128	172.136	1.14%
70x100	230.393	220.19861	4.42%
80x80	219.281	214.98955	1.96%
90x90	279.052	276.24393	1.01%
100x100	317.426	310.88034	2.06%

Table 3: Comparison in terms of number of iterations taken on Net-Lib test problems

S. No.	Problem Title	Size	SP1	DP1	% of saved iterations
1	ADLITTLE	56x97	22	23	−4.55%
2	AFIRO	27x32	6	6	0.00%
3	AGG2	516x302	47	47	0.00%
4	AGG3	516x302	37	37	0.00%
5	BANDM	305 x 472	9277	9911	−6.83%
6	BEACONFD	173x262	89	89	0.00%

7	BNL1	643x1175	8486	6914	18.52%
8	E226	223x282	128	133	−3.91%
9	FFFFF800	524x854	2001	1303	34.88%
10	ISRAEL	174x142	8	8	0.00%
11	SCAGR25	471x500	2051	1264	38.37%
12	SCAGR7	129x140	249	260	−4.42%
13	SCFXM2	660x914	1378	1265	8.20%
14	SCFXM3	990x1371	2408	1851	23.13%
15	SCORPION	388x358	398	369	7.29%
16	SCSD1	77x760	121	121	0.00%
17	SCSD6	147x1350	297	173	41.75%
18	SCSD8	397x2750	2784	1628	41.52%
19	SCTAP2	1090x1880	1910	929	51.36%
20	SCTAP3	1480x2480	1923	1234	35.83%
21	SHARE2B	96x79	120	119	0.83%
22	STOCFOR1	117x111	56	56	0.00%
					12.82%

The above comparisons in between DP1 and SP1 strengthen that DP1 is more practically efficient then SP1 in terms of number of iterations.

Now we would turn our focus on comparison of DP1 and SP1 in terms of basic arithmetic operations (e.g. additions and multiplications). Figure 1 and Figure 2 depict the comparison of DP1 and SP1 based on the average number of multiplication and addition operations needed to solve LP problems of different sizes which have been sorted according to increasing sizes of the coefficient matrix.

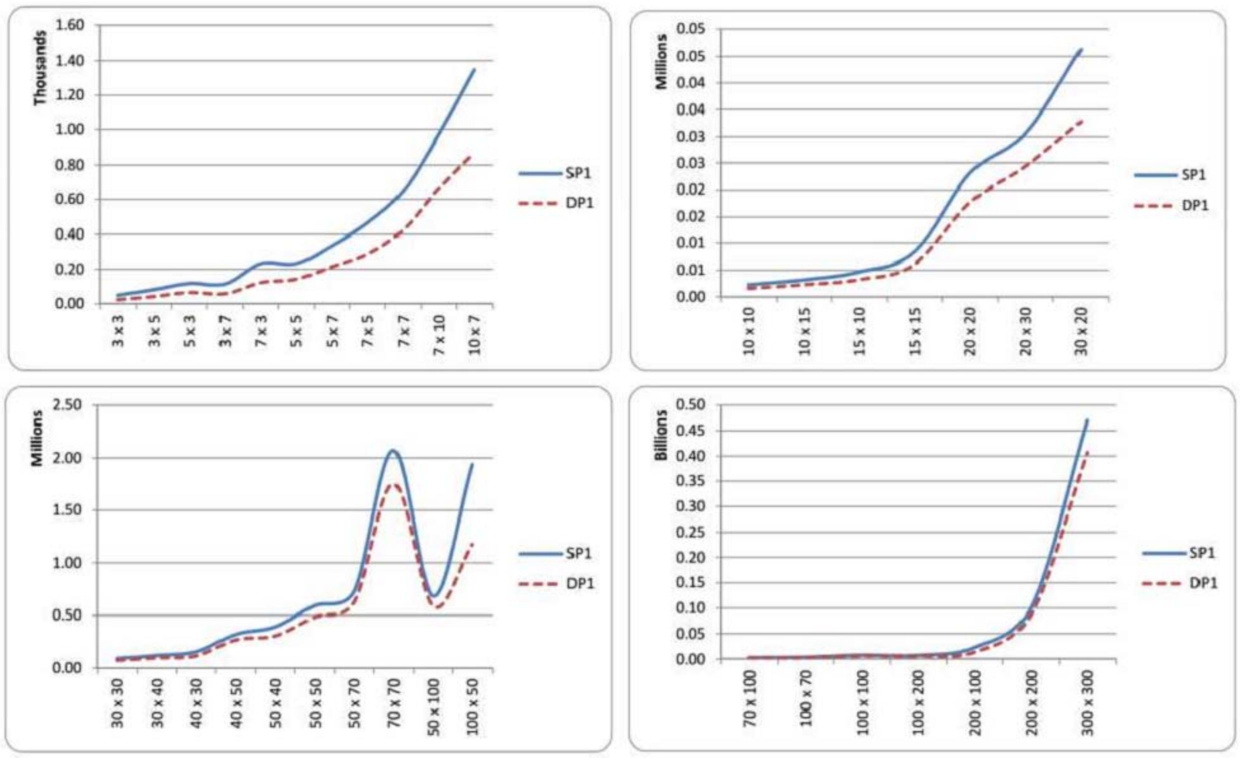

Figure 1: Comparison of DP1 and SP1 in terms of Multiplication Operations.

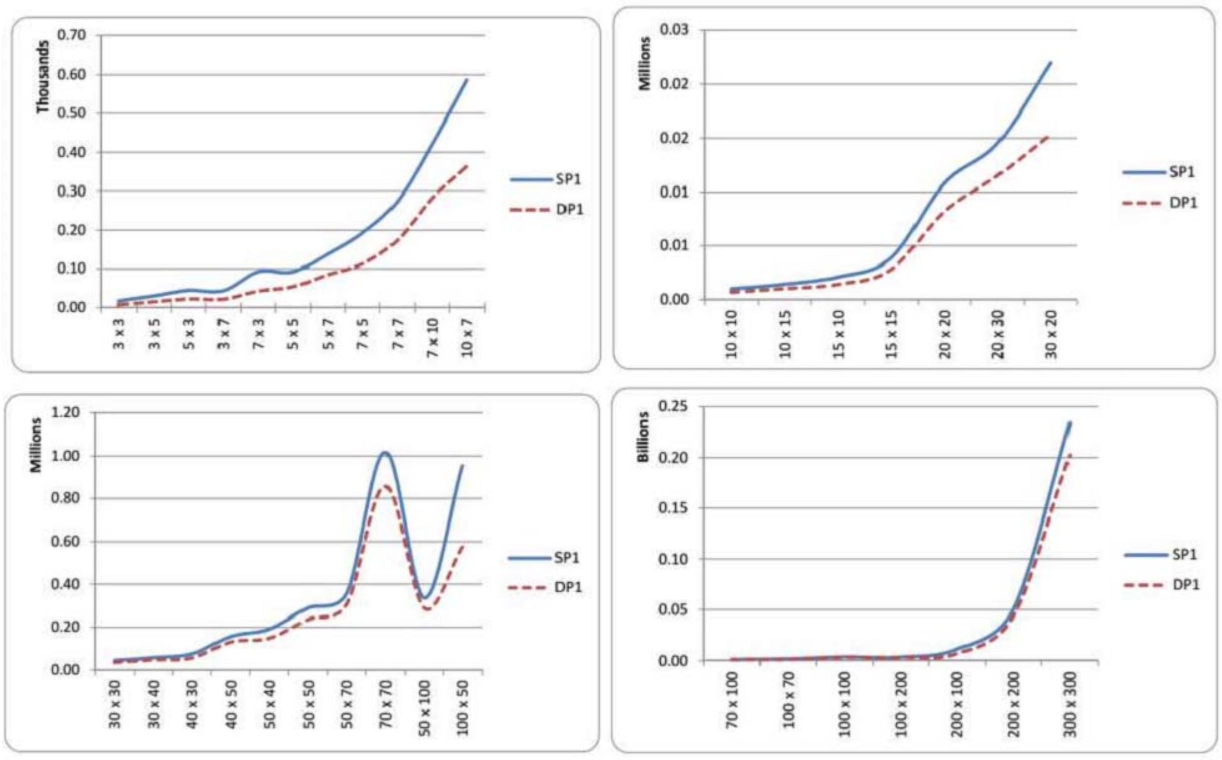

Figure 2: Comparison of DP1 and SP1 in terms of Addition Operations

The tables illustrate that (either we consider multiplications or additions) DP1 always need less number of operation computations as compared to SP1. This difference becomes quite remarkable especially for the problems that have greater number of constraints as compared to the number of variables. So, it is empirically observed that like its recent predecessor ASM (Inayatullah, Touheed, & Imtiaz, 2015) DP1 is much advantageous when "number of constraints minus number of variables" is a large number. This observation could be verified by the graph as depicted in Figure , as the value of $m-n$ increases the percentage of saved computations in DP1 also increases. For example for a 90×135 order problem the average saving in computations is just about 10% but in contrast to a 90×15 order problem it reaches a significant level of 80%. This fact can also be seen in the problems of order 30×90, 90×30, 50×90 and 90×50. For the problems having $m \approx n$, the savings in number of computations is not much high. Basic theory of duality asserts that in contrast to DP1, the dual version of DP1 would be more efficient computationally when $n-m$ is large.

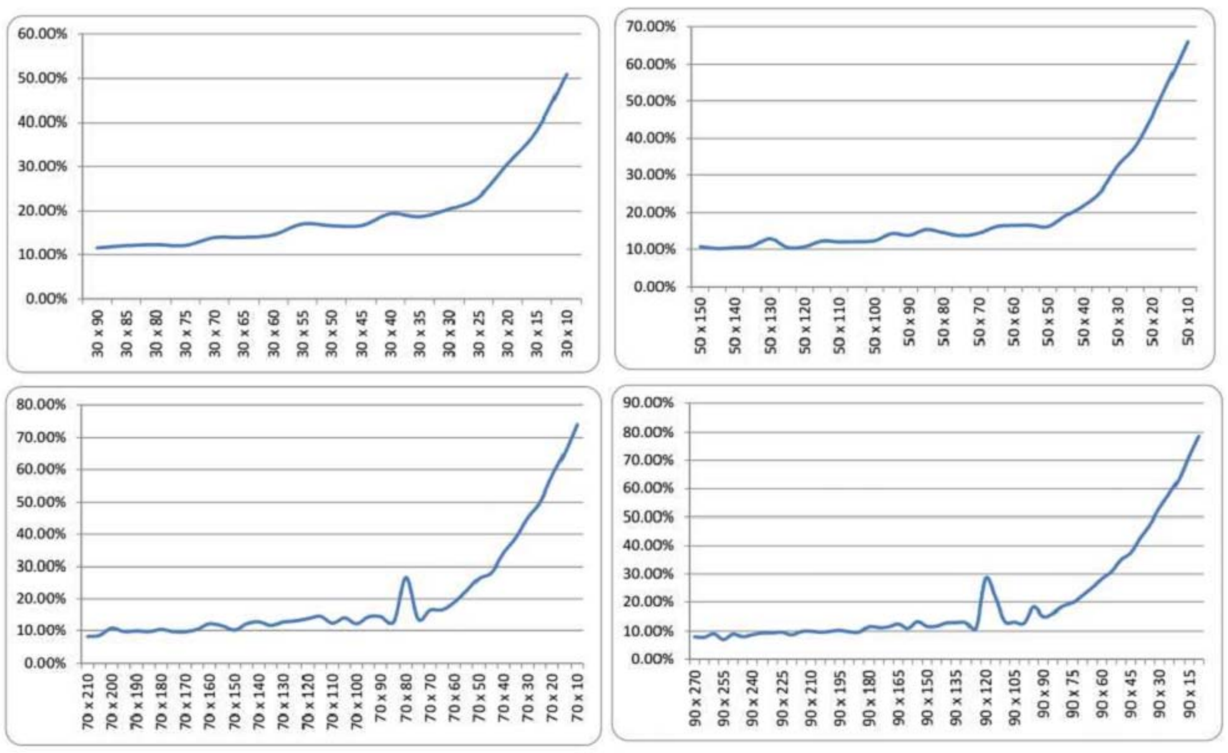

Figure 3 Graphs showing percentage of computations saved by DP1 over SP1.

7. Conclusion

In this article efficient variants of simplex method for feasibility (named DP1) have been discussed. This method does not need any kind of artificial variables or artificial constraints; it could directly start with any infeasible basis of an LP, providing full freedom to the user that whether to start with primal DP1 or dual DP1 without making any abrupt changes to the LP structure. Primary computational results showed that the method requires much lesser number of iterations as of SP1 on Netlib test problems. Computational results for basic arithmetic operations showed that DP1 is more efficient when $m-n$ is large. Hence the method also provides great benefits in class room teaching by eliminating the relatively difficult and tedious calculations of artificial variables and constraints. It is also a teaching aid for the teachers who want to teach feasibility achievement as a separate topic before teaching optimality achievement. It is very helpful tool in integer programming and sensitivity analysis, because it provides an option to avoid dual simplex method.

BIBLIOGRAPHY

Arsham, H. (1989). A tabular, simplex type algorithm as a teaching aid for general LP models. *Mathematical and Computer Modelling, 12,* 1051-1056.

Arsham, H. (1997). Initialization of the simplex algorithm : An artificial-free approach. *SIAM Rev.*, 736-744.

Arsham, H. (1997). An artificial-free simplex algorithm for general LP models. *Mathematical and Computer Modelling, 25*(1), 107-123.

Arsham, H., Baloh, P., Damij, T., & Grad, J. (2003). An Algorithm for Simplex Tableau Reduction with Numerical Comparison. *International Journal of Pure and Applied Mathematics, 4*, 53-80.

Arsham, H., Damij, T., & Grad, J. (2003). An algorithm for simplex tableau reduction: The push-to-pull solution strategy. *Applid Mathematics and computation*, 525-547.

Bruijn, W. D., Ketelaars, D., Gelsema, E., & Sorber, L. (1991). Comparison of the simplex method with several other methods for background-fitting for electron energy-loss spectral quantification of biological materials. *Microsc. Microanal. Microstruct., 2*, 281-291.

Chvatal, V. (1983). *Linear Programming.* United States of America: W.H. Freeman and Company.

Chvatal, V. (1983). *Linear Programming.* United States of America: W.H. Freeman and Company.

Dantzig, G., Orden, A., & Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific J. Math., 5 (2)*, 183-195.

Darvas, F. (1974). Application of the sequential simplex method in designing drug analogs. *Journal of Medicinal Chemistry, 17(8)*, 799-804.

Enge, A., & Huhn, P. (1998). A Counterexample to H.Arsham's "Initialization of the Simplex Algorithm : An Artificial-Free Approach". *Society for Industrial and Applied Mathematics*.

Goldfarb, D., & Reid, J. (1977). A practicable steepest edge simplex algorithm. *Mathematical Programming*, 361-371.

Haefner, J. W. (2005). *Biological Systems Principles and Applications.* United States of America: Springer.

Harris, P. (1973). Pivot Selection methods of the Devex LP code. *Mathematical Programming*, 1-28.

Huangfu, Q. (2013). High performance simplex solver. *Ph.D. dissertation, University of Edinburgh*.

Illes, T., & Terlaky, T. (2002). Pivot versus interior point methods: Pros and cons. *European Journal of Operational Research, 140(2)*, 170-190.

Inayatullah, S., Khan, N., Imtiaz, M., & Khan, F. H. (2010). New Minimum Angle Algorithms for Feasibility and Optimality. *Canadian Journal on Computing in Mathematics, Natural Sciences, Engineering & Medicines.*, 22-36.

Inayatullah, S., Touheed, N., & Imtiaz, M. (2015). A Streamlined Artificial Variable Free Version of Simplex Method. *PLoS ONE, 10(3)*.

Jatau, J. S., Datau, S. G., & Agyo, D. (1999). Application of Simplex Method to Determine the Minimum Deviation in Parameters Affecting Dimensional Accuracy During Rolling From their Set Values. *Bulletin of Science Association of Nigerian, 22*, 113-120.

Kaluzny, B. (2001). Finite Pivot algoirthms and Feasibility. *MS thesis, Faculty of Graduate Studies and Reseach, School of Computer Science, McGill University, Montreal, Quebec, Canada* .

Khan, N., Inayatullah, S., Imtiaz, M., & Khan, F. H. (2009). New artificial-free phase 1 simplex method. *International Journal of Basic and Applied Sciences, 9*, 97-114.

Koberstein, A. (2005). The Dual simplex Method, Techniques for a fast and stable implementation. *Ph. D. Dissertation*.

Pan, P.-Q. (2008). A largest-distance rule for the simplex algorithm. *European Journal of Operational Research, 187, No. 2*, 393-402.

Pan, P.-Q. (2008). Efficient nested pricing in the simplex algorithm. *Operations Research Letters, 36, No. 3*, 309-313.

Pan, P.-Q. (2010). A Fast Simplex Algorithm for Linear Programming. *Journal of Computational Mathematics, 28, No.6*, 837-847.

Pan, P.-Q. (2014). *Linear Programming Computation.* Springer Heidelberg New York Dordrecht London.

Papparrizos, K. (1990). The two-phase simplex without artificial variables. *Methods of Operations Research*, 73-83.

Pulgarín, J. A., Molina, A. A., & Alañón, M. T. (2002). The use of modified simplex method to optimize the room temperature phosphorescence variables in the determination of an antihypertensive drug. *Talanta, 57*, 795-805.

Serang, O. (2012). Conic Sampling: An Efficient Method for Solving Linear and Quadratic Programming by Randomly Linking Constraints within the Interior. *PLoS ONE*.

Sun, Y., Fleming, R. M., Thiele, I., & Saunders, M. A. (2013). Robust flux balance analysis of multiscale biochemical reaction networks. *BMC Bioinformatics*.

The Net-Lib Repository for Linear Problems. (n.d.). Retrieved from http://www.netlib.org/lp/data

Wagner, H. M. (1956, August). A Two-Phase Method for the Simplex Tableau. *Operation Research, Vol. 4, No. 4,*, 443-447.