An Efficient Method for Pivoting Free Variables in Linear Programming: A Computational Approach

Syed Inayatullah,
Asma Rani,
Tanveer Ahmed Siddiqi,
Hina Zaheer,
Muhammad Imtiaz,
Hafsa Athar Jafree,
Department of Mathematics, University of Karachi, Karachi, Pakistan

Doi: 10.19044/esj.2019.v15n9p1 URL:http://dx.doi.org/10.19044/esj.2019.v15n9p1

Abstract

Commonly used simplex method to solve linear programming problem do not allow variables to be negative during solution process and suggest to break each free variable (variable allowed to be negative) into difference of two non-negative variables. This transformation significantly increases the number of variables as well as after this the problem leaves its original variable space. , thus making the geometry of problem (during solution process) difficult to handle and understand. In this paper, we developed a natural generalization of simplex pivots for free variables. Described approach is capable of handling any general linear programming in its original variable space. In our computational study, the primary results showed that the new method outperforms simplex method on general LPs.

Keywords: Linear programming, unrestricted variables, simplex method, decomposition

Introduction

Since 1947, after World War II, linear programming has gained importance amongst the researchers of different fields (Dantzig, 1963). Today because of its tremendous impact in various disciplines, it has become a core research area of many Mathematicians, Economists and Decision Scientists. Linear programming is the optimization of an outcome based on some set of constraints using a linear mathematical model. It deals with maximizing (minimizing) of a linear function over a convex polyhedron specified by a set of linear constraints. The origin of developing algorithms to solve a given

system of linear inequalities actually goes back to the 19th century, where they were first studied by Fourier (Grattan-Guinness, 1970). Later, several mathematicians such as Dines (1918) and Motzkin (1952) rediscovered these algorithms. Simplex method developed by Dantzig (1963), which is specially designed to solve LPs with non-negative variables and so far, the most preferred method for solving LPs because of its efficiency (Shamir, 1987).

We categorize the variables in an LP in two kinds, first kind is *non-negative variables*, i.e. the variables having explicit non-negativity restrictions and the second kind is *free/unrestricted variables*, i.e. variables having no explicit non-negativity restriction.

Just a few versions of simplex algorithm presented in the literature and textbooks for solving LPs with free variables, which mostly initiated by decomposing the free variables as a difference of two non-negative variables thus converting it into an LP with explicit non-negativity restrictions on all variables. Dantzig stated in (Dantzig, 1963) another decomposition technique, which requires insertion of a single additional variable to the problem, and attributed this decomposition to A. W. Tucker. Later on, Schechter (1991) has presented geometrical interpretation of above technique, but in 1985, Gass (1985) had already proven that defining free variables as difference of two nonnegative variables is computationally inefficient.

For larger LPs, implementation of the simplex method with the decomposition of free variables increases the number of variables and importantly loses the geometry of problem in the original variable space. Furthermore, this makes a technically incorrect impression on the reader that linear programming with free variables is perhaps a special case of linear programming with non-negative variables. Actually linear programming with unrestricted variables is a generalization of linear programming with non-negative variables, so there must be a generalized way of choosing entering and leaving basic variables that can directly deal unrestricted variables and as well as non-negative variables.

Orchard-Hays (1968), Spivey and Thrall (1970), Gass (1985) and Dantzig and Thapa (1997) discussed a way of handling free variables in terms of explicit representation within a simplex tableau format. But that method lacks reliability from the perspective of efficiency on large LPs, because of the randomness involved in the selection of initial explicit representations.

Here, this paper would reveal a similar but systematic and efficient procedure that could directly handle unrestricted variables in solving general LPs.

A Linear Programming Problem

A general LP problem with mix kind of variables could be defined as,

$$\begin{aligned} &\text{Maximize} && z = \mathbf{c}^T\mathbf{x} \\ &\text{subject to} && A\mathbf{x} = \mathbf{b} \\ &\text{where} && \mathbf{x} = \begin{bmatrix} \mathbf{x}_U \\ \mathbf{x}_O \end{bmatrix}, \mathbf{x}_O \geq 0, \mathbf{x} \in \Re^n \end{aligned} \tag{1}$$

where, *U* as index set of variables that have no explicit bound, and *O* as index set of variables that have explicit non-negativity conditions, $A \in \Re^{m \times n}, \mathbf{b} \in \Re^m, \mathbf{c} \in \Re^n$ and $m \leq n$. It is assumed that A is full rank.

Let *B* be the set of indices of the variables in the basis, and *N* be the set of indices of variables in the non-basis, such that A_B is invertible, and non-basis $N := \{1, \cdots, n\} \setminus B$. We may write

$$\begin{aligned} & A_B\mathbf{x}_B + A_N\mathbf{x}_N = \mathbf{b} \\ \Rightarrow\ & \mathbf{x}_B + A_B^{-1}A_N\mathbf{x}_N = A_B^{-1}\mathbf{b} \end{aligned} \tag{2}$$

Now by substituting the value of $\mathbf{x}_B$ from equation (2), the objective of system (1) can be reformulated as

$$\begin{aligned} & z - (\mathbf{c}_B^T\mathbf{x}_B + \mathbf{c}_N^T\mathbf{x}_N) = 0 \\ \Rightarrow\ & z - \mathbf{c}_B^T(A_B^{-1}\mathbf{b} - A_B^{-1}A_N\mathbf{x}_N) - \mathbf{c}_N^T\mathbf{x}_N = 0 \\ \Rightarrow\ & z = (\mathbf{c}_N^T - \mathbf{c}_B^TA_B^{-1}A_N)\mathbf{x}_N + \mathbf{c}_B^TA_B^{-1}\mathbf{b} \end{aligned}$$

The following collection of equations along with non-negativity condition on variables $\mathbf{x}_0$,

$$\frac{\mathbf{x}_B + A_B^{-1}A_N\mathbf{x}_N = A_B^{-1}\mathbf{b}}{Max\ z = (\mathbf{c}_N^T - \mathbf{c}_B^TA_B^{-1}A_N)\mathbf{x}_N + \mathbf{c}_B^TA_B^{-1}\mathbf{b}} \tag{3}$$

are termed as dictionary of the LP system (1) for basis *B*.

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

$$D(B) = \left\{ \begin{array}{l} x_i + \sum_{j \in N} \alpha_{ij}x_j = \beta_i,\ i \in B \\ \hline Maximize\ z = \sum_{j \in N} \gamma_j x_j + \hat{z} \end{array} \right\} \tag{4}$$

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 $i \in O$.

Selection of entering and leaving variables:

The foremost requirement in solving a linear programming problem by simplex method is an initial basic feasible solution. Geometrically, it lies at the origin. The simplex method then iterates along edges to adjacent corner points of the feasible region in search of a better objective value. Algebraically, step of the selection of moving edge is known as the selection of entering basic variable.

Consider the following LP,

$$\begin{aligned}
&\textit{Maximize } Z = 10x_1 - 100x_2 + 9x_3 - 110x_4 \\
&\textit{subject to} \\
&2x_1 + 3x_2 + 3x_3 - 2x_4 + x_5 = 12 \\
&x_1 - x_2 - 5x_3 + x_4 + x_6 = 20 \\
&-3x_1 + 4x_2 + 8x_3 + 4x_4 + x_7 = 24 \\
&2x_1 - 4x_2 - 5x_3 - 6x_4 + x_8 = 60 \\
&3x_1 - 4x_2 - 3x_3 - 2x_4 + x_9 = 12 \\
&x_3, x_4, x_5, x_6, x_7, x_8 \geq 0
\end{aligned}$$

Here x_1, x_2 and x_9 could be treated as unrestricted variables because there are no bounds on their values mentioned explicitly, set $U = \{1,2,9\}$. Whereas x_3, x_4, x_5, x_6, x_7 and x_8 are termed as non-negative variables because of an explicit description of zeros as their lower bounds, set $O = \{3,4,5,6,7,8\}$.

For the initial feasible basis *B*, setting $B = \{5,6,7,8,9\}$, is the easiest choice. Therefore corresponding $N = \{1,2,3,4\}$, the initial solution (0,0,0,0,12,20,24,60,12) with $Z = 0$ is obtainable by plugging in the values of

non-basic variables equal to zero. From the expression of objective function it is clear that any increase in values of the non-basic variables x_1 and x_3 have positive impact and x_2 and x_4 have negative impact on objective value *Z*. In other words, one could say that to increase the value of *Z* the non-basic x_1 and x_3 could be increased while x_2 and x_4 could instead be decreased. But x_4 already on its non-negativity lower bound so could not be decreased any more. In short, x_1, x_2 and x_3 are *candidate entering* variables of which x_1 and x_3 are *candidate increasing* and x_2 is *candidate decreasing (here x_4 is not considered as candidate decreasing variable)*.

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 non-negative variables, some of which are, Dantzig's largest coefficient rule (Dantzig, 1963), steepest edge rule (1977), Devex rule (Harris, 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). Here in this paper we are using a generalization of Dantzig's largest coefficient method. According to this criterion, *most preferred entering variable is the variable along which Z has highest increasing rate*. In contrast to Dantzig's original method here entering variable would not be necessarily increasing and may be decreasing as well. In the example defined above x_2 is a preferred entering variable, because one unit decrease in its value results a 100 units increase in the value of *Z*, which is highest with respect to other candidate variables.

For any increasing (decreasing) variable x_i, a variable x_j is said to be *leaving variable* if x_j provides most stringent bound on the increase (decrease) of variable x_i.

As in the example defined above,

- if x_1 is entering variable then x_5 would be the leaving variable, because as x_1 increased to value 6, x_5 get struck with its zero lower bound firstly among other basic variables.
- If x_2 were selected as entering variable then x_8 would be the leaving variable, because as x_2 decreased to -15, x_8 reached its zero lower bound quicker than other basic variables.

- If x_3 were selected as entering variable then x_7 is the leaving variable because as x_3 increased to value 3, x_7 get struck with its zero lower bound firstly among other basic variables.

Note: Unrestricted variables would never become a leaving variable because they didn't have any upper or lower bound.

General rule:

For Dictionary 0, let R_1 be index sets of increasing variables in $\mathbf{x}_N$, Clearly, $R_1 := \{j : \gamma_j > 0, j \in N\}$. Let R_2 is the index set of decreasing variables in $\mathbf{x}_{N \cap U}$. Clearly $R_2 := \{j : \gamma_j < 0, j \in N \cap U\}$. So, index set of preferred entering variables among all the non-basic variables would be defined by, $R = \{j : j = \arg\max\{\gamma_{R_1}, -\gamma_{R_2}\}\}$. If *R* gets only a single element, say *k*, then x_k would be preferred entering variable and if *R* gets multiple elements then choice could be arbitrary.

For leaving variable, since the variables in $\mathbf{x}_{B \cap U}$ has no upper or lower bound, they have no reason to leave the basis. So the leaving variable is chosen from $\mathbf{x}_{B \cap O}$ only, by performing the following ratio test: "*If* $k \in R_1$ *then index of the leaving variable is obtained by* $r = \arg\min\{\beta_i / \alpha_{ik} : \alpha_{ik} > 0, i \in B \cap O\}$, *while if* $k \in R_2$ *then* $r = \arg\min\{-\beta_i / \alpha_{ik} : \alpha_{ik} < 0, i \in B \cap O\}$".

Theorem 1: Optimality condition

A feasible basis *B* is said to be optimal if in associated dictionary $\gamma_j \leq 0$ $\forall j \in O$ and $\gamma_j = 0$ $\forall j \in U$.

Proof:

$\gamma_j \leq 0$ for all $j \in O$ implies that there is no non-negative non-basic variable available that could be used to increase the value of *Z* without violating its zero lower bound, and $\gamma_j = 0$ for all $j \in U$ implies that there is no free non-basic variable available. Hence it shows optimality of current basis.

Theorem 2: Unboundedness condition

The linear program (1) is unbounded if it has a feasible basis *B* and in the associated dictionary there exists either $\gamma_j > 0$, $j \in N$ such that $\alpha_{Bj} \leq 0$ or $\gamma_j < 0$, $j \in N \bigcap U$ such that $\alpha_{Bj} \geq 0$.

Proof:

Consider the case of $\gamma_j > 0$, $j \in N$ and $\alpha_{Bj} \leq 0$ which implies that from current (feasible) basis one can increase the value of x_j indefinitely and the objective value will increase in direct proportion to x_j.

Now consider the case of $\gamma_j < 0$, $j \in N \bigcap U$ and $\alpha_{Bj} \geq 0$ which implies that from current (feasible) basis one can decrease the value of x_j indefinitely and the objective value will increase in direct proportion to decrease in x_j.

Description of the procedure:

Problem

Given a dictionary *D*(*B*), with index set *U* of free variables and the index set *O* of nonnegative variables. Obtain an optimal basis.

Algorithm

Step 1: Let $R_1 \subseteq N$ such that $R_1 := \{j : \gamma_j > 0, j \in N\}$,

$R_2 \subseteq N \bigcap U$ such that $R_2 := \{j : \gamma_j < 0, j \in N \bigcap U\}$

If $R_1 \bigcup R_2 = \phi$ then *D(B)* is optimal. **Exit**.

Step 2: Set $R := \{j : j = \arg\max\{\gamma_{R_1}, -\gamma_{R_2}\}\}$. If *R* gets only a single element, say *k*, then x_k would be entering basic variable and if *R* gets multiple elements then choice could be made on maximum of these.

Step 3: if $k \in R_1$,

$$r = \arg\min\{\beta_i / \alpha_{ik} : \alpha_{ik} > 0, i \in B \bigcap O\}$$

Otherwise,

$$r = \arg\min\{-\beta_i / \alpha_{ik} : \alpha_{ik} < 0, i \in B \bigcap O\}$$

Step 4: If *r* does not exist, then the given problem is unbounded. Exit. Otherwise make a pivot on (r,k).

Set $B=(B\cup\{k\})\setminus\{r\}$, $N=(N\cup\{r\})\setminus\{k\}$ and update *D(B).*Go to step 1

Example 1:

Consider the following LP,

$$\begin{aligned} &\textit{Maximize} \quad z=-82x_1-87x_2-9x_3 \\ &\textit{subject to} \\ &19x_1-27x_2+4x_3\le 11 \\ &25x_1+42x_2+50x_3\le 97 \\ &-4x_1-34x_2-42x_3\le 1 \\ &-41x_1+33x_2-5x_3\le 78 \end{aligned}$$

On insertion of non-negative slack variables x_4, x_5, x_6 and x_7 problem becomes,

$$\begin{aligned} &\textit{Maximize} \quad z=-82x_1-87x_2-9x_3 \\ &\textit{subject to} \\ &19x_1-27x_2+4x_3+x_4=11 \\ &25x_1+42x_2+50x_3+x_5=97 \\ &-4x_1-34x_2-42x_3+x_6=1 \\ &-41x_1+33x_2-5x_3+x_7=78 \\ &x_4\ge 0, x_5\ge 0, x_6\ge 0, x_7\ge 0 \end{aligned}$$

Initial dictionary for basic variables x_4, x_5, x_6 and x_7 will be,

$$\begin{aligned} x_4 &= 11-19x_1+27x_2-4x_3 \\ x_5 &= 97-25x_1-42x_2-50x_3 \\ x_6 &= 1+4x_1+34x_2+42x_3 \\ x_7 &= 78+41x_1-33x_2+5x_3 \\ \hline \textit{Max}\,Z &= 0-82x_1-87x_2-9x_3 \end{aligned}$$

Here all the non-basic variables are decreasing variables, according to criteria defined above in section 4, x_2 is most preferred choice to enter in to the basis and then x_6 would leave the basis.

After performing the change of basis operations, we would get following system,

$$x_4 + \frac{377}{17}x_1 - \frac{27}{34}x_6 + \frac{635}{17}x_3 = \frac{347}{34}$$
$$x_5 + \frac{341}{17}x_1 + \frac{21}{17}x_6 - \frac{32}{17}x_3 = \frac{1670}{17}$$
$$x_2 + \frac{2}{17}x_1 - \frac{1}{34}x_6 + \frac{21}{17}x_3 = -\frac{1}{34}$$
$$x_7 - \frac{763}{17}x_1 + \frac{33}{34}x_6 - \frac{778}{17}x_3 = \frac{2685}{34}$$
$$MaxZ = -\frac{1220}{17}x_1 - \frac{87}{34}x_6 + \frac{1674}{17}x_3 + \frac{87}{34}$$

Here x_1 is decreasing and x_3 is increasing variable, in which x_3 is our preferred choice (see section 4). Then leaving variable would be x_4.

$$x_3 + \frac{377}{635}x_1 - \frac{27}{1270}x_6 + \frac{17}{635}x_4 = \frac{347}{1270}$$
$$x_5 + \frac{13447}{635}x_1 + \frac{759}{635}x_6 + \frac{32}{635}x_4 = \frac{62706}{635}$$
$$x_2 - \frac{391}{635}x_1 - \frac{2}{635}x_6 - \frac{21}{635}x_4 = -\frac{233}{635}$$
$$x_7 - \frac{6146}{347} * x_1 - \frac{3}{1270}x_6 + \frac{778}{635}x_4 = \frac{12715}{139}$$
$$MaxZ = -\frac{12632}{97}x_1 - \frac{591}{1270}x_6 - \frac{1674}{635}x_4 + \frac{2033}{69}$$

Here x_1 is only choice for entering variable. Leaving variable is x_7.

$$x_3 + \frac{377}{11247}x_7 - \frac{80}{3749}x_6 + \frac{158}{2329}x_4 = \frac{1633}{489}$$
$$x_5 + \frac{599}{501}x_7 + \frac{2875}{2411}x_6 + \frac{944}{623}x_4 = \frac{3538}{17}$$
$$x_2 - \frac{17}{489}x_7 - \frac{1}{326}x_6 - \frac{37}{489}x_4 = -\frac{3469}{978}$$
$$x_1 - \frac{347}{6146}x_7 + \frac{1}{7498}x_6 - \frac{103}{1489}x_4 = -\frac{2102}{407}$$
$$MaxZ = -\frac{2169}{295}x_7 - \frac{267}{596}x_6 - \frac{2981}{256}x_4 + \frac{19657}{28}$$

Since all the variables are in their allowable range (implies feasibility) and there is no one which could be used to increase Z without violating any constraint. Therefore optimality achieved and the optimal solution is $(x_1, x_2, x_3) = \left(\frac{-2102}{407}, \frac{-3469}{978}, \frac{1633}{489}\right)$ with $z = \frac{19657}{28}$.

Example 2:

Now consider another example,

$$\begin{aligned} &\textit{Maximize} && z = 15x_1 + 35x_2 + 9x_3 \\ &\textit{subject to} \\ & && -30x_1 + 3x_2 - 10x_3 \le 26 \\ & && -36x_1 + 4x_2 + 18x_3 \le 4 \\ & && 20x_1 + 37x_2 + 25x_3 \le 75 \\ & && -40x_1 - x_2 + 3x_3 \le 24 \end{aligned}$$

Initial Dictionary:

$$\begin{aligned} x_4 &= 26 - 30x_1 + 3x_2 - 10x_3 \\ x_5 &= 4 - 36x_1 + 4^* x_2 + 18x_3 \\ x_6 &= 75 + 20x_1 + 37x_2 + 25x_3 \\ x_7 &= 24 + 40x_1 - 1x_2 + 3x_3 \\ \hline MaxZ &= 0 + 15x_1 + 35x_2 + 9x_3 \end{aligned}$$

Iteration 1:

$$\begin{aligned} x_4 &= 23 - 3x_1 - \tfrac{3}{4}x_5 - \tfrac{47}{2}x_3 \\ x_2 &= 1 - 9x_1 + \tfrac{1}{4}x_5 + \tfrac{9}{2}x_3 \\ x_6 &= 38 + 353^* x_1 - \tfrac{37}{4}x_5 - \tfrac{283}{2}x_3 \\ x_7 &= 25 - 49x_1 + \tfrac{1}{4}x_5 + \tfrac{15}{2}x_3 \\ \hline MaxZ &= 35 + 330x_1 - \tfrac{35}{4}x_5 - \tfrac{332}{2}x_3 \end{aligned}$$

Iteration 2:

$$\begin{aligned} x_4 &= \tfrac{8233}{353} + \tfrac{3}{353}x_6 - \tfrac{585}{706}x_5 - \tfrac{4817}{195}^* x_3 \\ x_2 &= \tfrac{695}{353} + \tfrac{9}{353}x_6 + \tfrac{5}{353}x_5 + \tfrac{315}{353}x_3 \\ x_1 &= \tfrac{38}{353} + \tfrac{1}{353}x_6 - \tfrac{37}{1412}x_5 - \tfrac{283}{706}x_3 \\ x_7 &= \tfrac{3966}{131} + \tfrac{49}{353}x_6 - \tfrac{365}{353}x_5 - \tfrac{4286}{353}x_3 \\ \hline MaxZ &= \tfrac{11707}{166} - \tfrac{330}{353}x_6 - \tfrac{145}{1412}x_5 - \tfrac{24159}{706}x_3 \end{aligned}$$

Iteration 3:

$$
\begin{aligned}
x_3 - \tfrac{3}{8720}x_6 + \tfrac{117}{3488}x_5 - \tfrac{195}{4817}x_4 &= -\tfrac{896}{949} \\
x_2 + \tfrac{45}{1744}x_6 - \tfrac{55}{3488}x_5 + \tfrac{63}{1744}x_4 &= \tfrac{1684}{599} \\
x_1 + \tfrac{47}{17440}x_6 - \tfrac{89}{6976}x_5 - \tfrac{283}{17440}x_4 &= -\tfrac{296}{1093} \\
x_7 + \tfrac{587}{4360}x_6 - \tfrac{1093}{1744}x_5 - \tfrac{724}{1473}x_4 &= \tfrac{4684}{249} \\
\hline
MaxZ = \; -\tfrac{1525}{1611}x_6 + \tfrac{1111}{10632}x_5 - \tfrac{658}{475}x_4 &+ \tfrac{20875}{203}
\end{aligned}
$$

Here x_5 is entering variable but there is no leaving variable, which is indication of unbounded optimal solution.

Computational Results:

Following table presents a comparison of average number of iterations of our algorithm (USM) with Danzig's simplex method (SM) (Dantzig, 1963). Using random models suggested by Kaluzny (2001),

$$\text{Maximize} \qquad \sum_{j=1}^{n} c_j \mathbf{x}_j$$

subject to

$$\sum_{j=1}^{n} a_{ij}\mathbf{x}_j \le \mathbf{b}_i, \quad i = 1,2,...,m$$

$$\mathbf{x}_j \ge 0, \quad j = 1,2,...,n$$

We generated 250 linear programs with the coefficients c_j, b_i and a_{ij} chosen randomly from the integer interval $[-50,50]$, and used MATLAB to generate the following results. The results depict that on average USM take much lesser number of iterations than SM, especially on higher order problems.

Table 1: Average number of iterations on random LPs.

Order	USM	SM	Average Number of Iterations to be saved in USM (in %)
3 x 3	1.58566	1.79283	11.55547
3 x 5	1.64542	1.73705	5.275035
3 x 7	1.5259	1.58566	3.768778
5 x 3	2.53386	2.80876	9.787237
5 x 5	2.7012	3.17928	15.03737
5 x 7	2.60159	2.79681	6.980095
7 x 5	3.98008	4.75697	16.33161
7 x 7	3.83665	4.36255	12.05488
7 x 10	3.67331	3.98805	7.892078
10 x 10	5.77689	6.77689	14.75603
10 x 15	5.41434	6.0757	10.88533
15 x 10	9.0757	11.988	24.29346
15 x 15	8.54183	10.2032	16.28283
20 x 20	12.0518	14.9841	19.56941
20 x 30	11.3586	13.2789	14.46129
30 x 20	19.9562	29.3426	31.98899
30 x 30	18.0598	22.7888	20.75142
30 x 40	17.5777	21.9681	19.98534
40 x 40	24.3984	33.0279	26.12791
40 x 50	23.9801	31.4143	23.66502
50 x 50	30.8446	42.3347	27.14109
50 x 70	30.3825	40.9482	25.8026
50 x 100	29.6175	39.4701	24.96219
70 x 50	47.0199	76.9044	38.85929
70 x 70	43.6614	64.3984	32.20111
70 x 100	43.6175	60.8008	28.26163
100 x 70	69.749	125.833	44.57018
100 x 100	64.5498	100.251	35.61181
100 x 200	62.4064	91.4223	31.73832
200 x 100	205.163	403.558	49.16146
200 x 200	133.769	250.06	46.50524
200 x 300	130.661	231.869	43.64878
300 x 200	223.02	533.534	58.19948
300 x 300	204.243	418.183	51.15942
300 x 400	202.279	399.147	49.32218
400 x 300	289.347	699.709	58.64752
400 x 400	275.598	611.841	54.95594
400 x 500	273.084	589.139	53.64693
500 x 400	355.96	881.558	59.62149
500 x 500	348.578	823.964	57.695

Furthermore, the comparison between USM and SM illustrates by the graphs between “number of elements in coefficient matrix” versus “average number of iterations” plotted below.

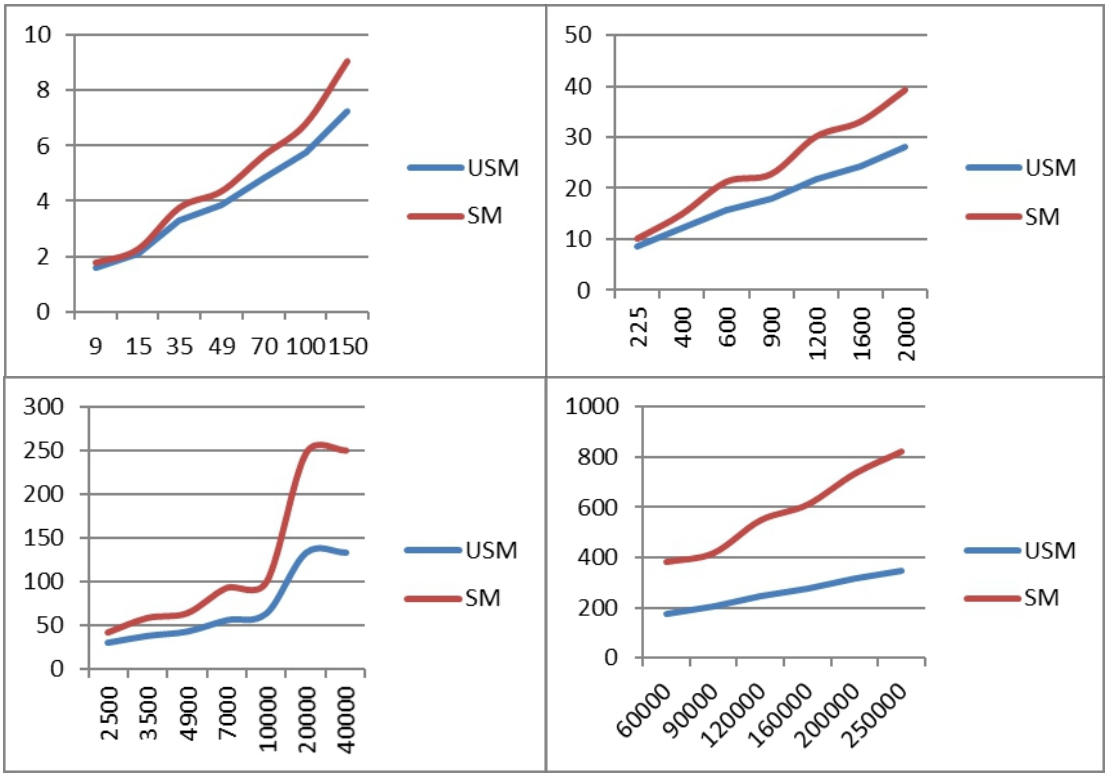

Figure (1): Graphs showing the comparison of average iterations between USM and SM with respect to number of elements in the coefficient matrix. Here numbers of elements in coefficient matrix mentioned on horizontal axis and average number of iterations on vertical axis.

From figure (1), it is clearly observable that USM has greater efficiency for large coefficient matrices. To get a visualization of this increasing trend of efficiency, we also plotted the following graph between relative efficiency of USM and the number of elements in coefficient matrices.

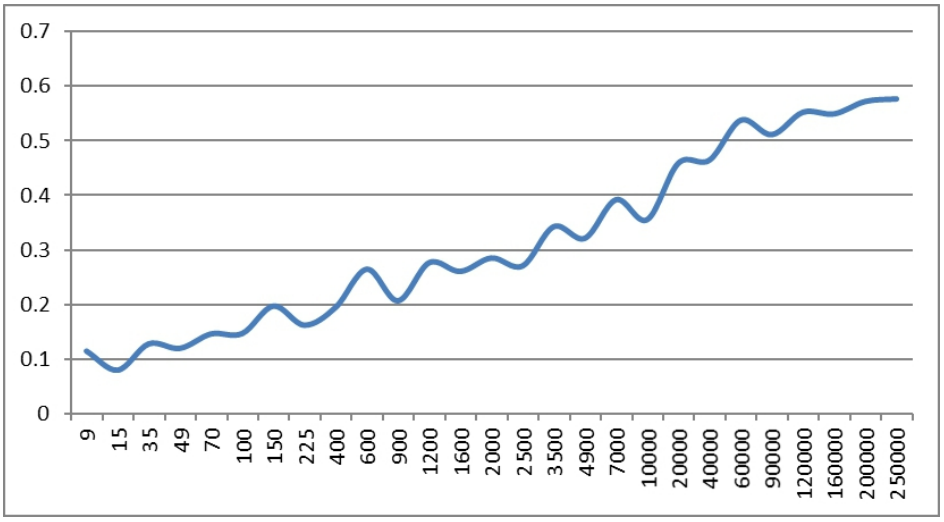

Figure (2): Trend of relative efficiency of USM with respect to number of elements in coefficient matrix. Total number of elements in coefficient matrix mentioned on horizontal axis and fraction of number of average saved iterations mentioned on vertical axis.

Now, to further analyze the trend behavior with respect to order of the coefficient matrices, we observed relationship between % relative efficiency and the row-column ratio $\left(\frac{m}{n}\right)$ of coefficient matrices, for *m*= 40,60,80, and 100.

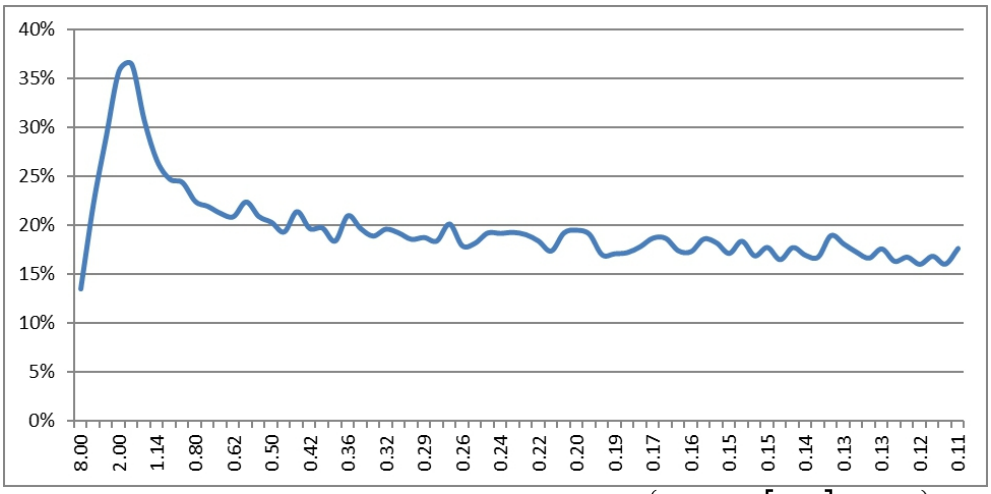

Figure (3): The data obtained by taking *m=40* and $n = \{5p \mid p \in [1,70], p \in Z\}$.

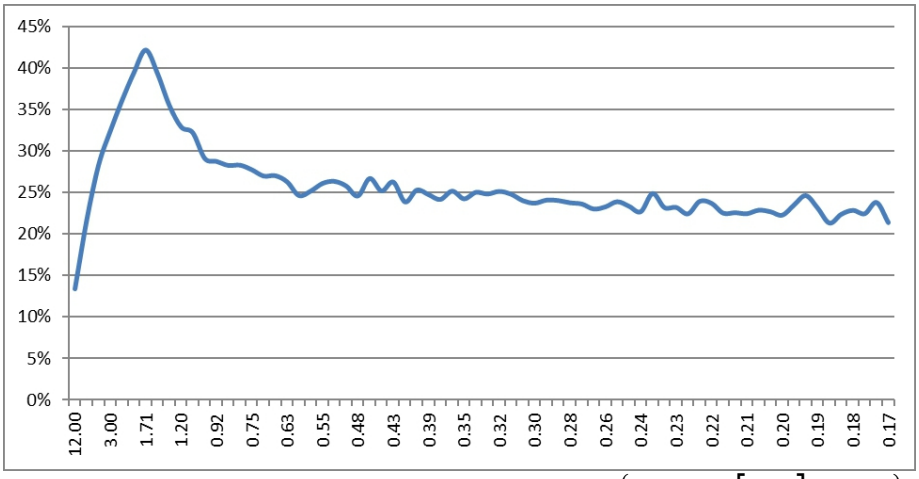

Figure (4): The data obtained by taking *m=60* and $n = \{5p \mid p \in [1,70], p \in Z\}$.

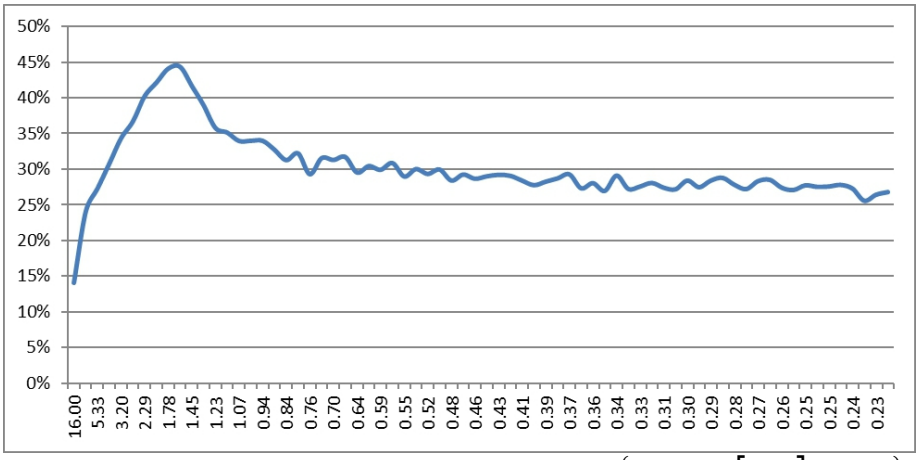

Figure (5): The data obtained by taking *m=80* and $n = \{5p \mid p \in [1,70], p \in Z\}$.

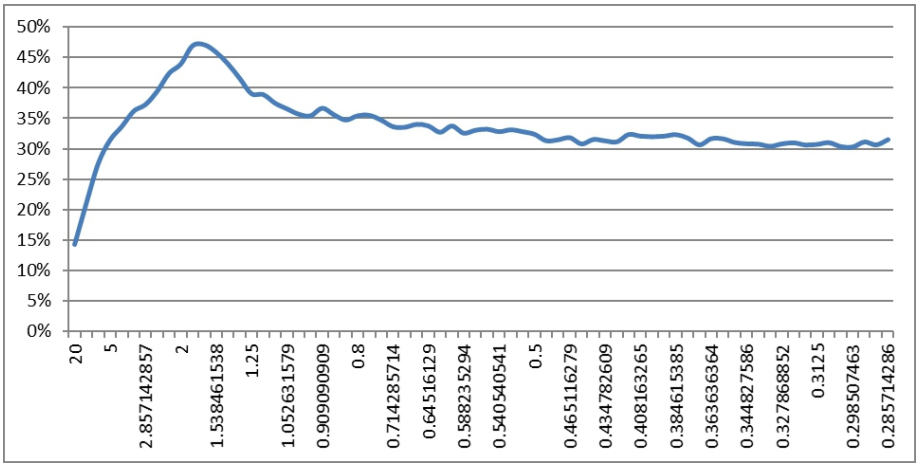

Figure (6): The data obtained by taking *m=100* with $n = \{5p \mid p \in [1,70], p \in Z\}$.

One can notice in figures (3) to (6), as the value of *m* increases, the relative efficiency of USM also increases, and gradually attain a maximum value when number of constraints are nearly double the number of variables ($m \approx 2n$), afterwards the trend is approaching to a limiting value for a long run.

Applications

Although every LP with explicit non-negativity conditions on variables could also be considered as a special case of LP with free variables, besides free variables exclusively arise in a wide number of practical situations too, e.g. production smoothing applications in which decision variables are defined to include periodical differences in production levels which could be positive or negative (Gass, 1985). A linear-programming formulations of zero-sum two-person games that define the unrestricted value of the game as a variable (Gass, 1985) and numerical and statistical problems that utilize linear-programming methods for their solution (Rabinovitz, 1968) etc.

Conclusion

In this paper we have developed an approach that could be applied to generalized LPs having either free or non-negative variables. By introducing the new rules for entering and leaving variables, the presented approach obviates the need of transforming a given LP involving unrestricted variables into an LP with non-negativity restrictions. Consequently this algorithm saves a lot of computational efforts for larger problems. Computational results, discussed in the end, showed that USM is generally more efficient than SM and works exceptionally well when ratio of number of constraints with number of variables lies near 2 i.e, $m/n \approx 2$.

References:

1. Chvatal, V. (1983). *Linear Programming.* United States of America: W.H. Freeman and Company.
2. Dantzig, G. (1963). *Linear Programming and Extensions.* Princeton University Press.
3. Dantzig, G., & Thapa, M. (1997). *Linear Programming, 1: Introduction.* (P. Glynn, Ed.) Springer Verlag.
4. Dines, L. (1918). Systems of inequalities. *Annal of Mathematics, 20 (2),* 191-198.
5. Gass, S. (1985). *Linear Programming Methods and Application.* New York: McGraw-Hill.
6. Gass, S. (1985). On the solution of Linear-programming Problems with free variables. *Comput. & Ops. Res., 12, No.3,* 265-271.

7. Goldfarb, D., & Reid, J. (1977). A practicable steepest edge simplex algorithm. *Mathematical Programming*, 361-371.
8. Grattan-Guinness, I. (1970). Joseph Fourier's Anticipation of Linear Programming. *Operational Research Quarterly, 21(3)* , 361-364.
9. Harris, P. (1973). Pivot Selection methods of the Devex LP code. *Mathematical Programming*, 1-28.
10. 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.
11. 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* .
12. Motzkin, T. (1952). Contributions to the theory of linear inequalities. *RAND Corporation Translation 22*.
13. Orchard-Hays, W. (1968). *Advanced Linear-Programming Computing Techniques.* New york: McGraw-Hill.
14. Pan, P.-Q. (2008). A largest-distance rule for the simplex algorithm. *European Journal of Operational Research, 187, No. 2*, 393-402.
15. Pan, P.-Q. (2008). Efficient nested pricing in the simplex algorithm. *Operations Research Letters, 36, No. 3*, 309-313.
16. Pan, P.-Q. (2010). A Fast Simplex Algorithm for Linear Programming. *Journal of Computational Mathematics, 28, No.6*, 837-847.
17. Rabinovitz, P. (1968, April). Application of linear programming to numerical analysis. *SIAM Review, 10*.
18. Schechter, M. (1991, June). Unrestricted variables in Linear Programming. *Journal of Optimization Theory and Applications, 69*.
19. Shamir, R. (1987). The efficiency of simplex method:A survey. *Management Science*, 301-334.
20. Spivey, W. A., & Thrall, R. M. (1970). *Linear Optimization.* New York: Holt, Rinehart & Winston.