A MODIFICATION OF REVISED SIMPLEX METHOD

Marko D. Petković¹, Predrag S. Stanimirović¹∗ and Nebojša V. Stojković²

¹University of Niš, Department of Mathematics, Faculty of Science, Višegradska 33, 18000 Niš, Yugoslavia.
²University of Niš, Faculty of Economics, Trg Kralja Aleksandra 11, 18000 Niš, Yugoslavia

E-mail: dexter_of_nis@inmail.sk, peckois@ptt.yu, nebojsas@orion.eknfak.ni.ac.yu

Abstract

We introduce a modification of the revised simplex method, which accelerates the process of finding the first basic feasible solution in the two phases revised simplex method. We report computational results on numerical examples from Netlib test set.

AMS Subj. Class.: 90C05
Key words: Linear programming, Revised simplex method, MATHEMATICA.

1 Introduction

Consider the linear program in the general form

\[ \text{Maximize } f(x) = f((x_N)_1, \ldots, (x_N)_{n_1}) = \sum_{i=1}^{n_1} c_i (x_N)_i + d \]

subject to

\[ \begin{align*}
N_1^{(1)} : & \quad \sum_{j=1}^{n_1} a_{ij} (x_N)_j \leq b_i, \quad i = 1, \ldots, r \\
N_1^{(2)} : & \quad \sum_{j=1}^{n_1} a_{ij} (x_N)_j \geq b_i, \quad i = r + 1, \ldots, s \\
J_i : & \quad \sum_{j=1}^{n_1} a_{ij} (x_N)_j = b_i, \quad i = s + 1, \ldots, m \\
(x_N)_j \geq 0, \quad & j = 1, \ldots, n_1.
\end{align*} \]

∗Corresponding author
Every inequality of the form \( N_i^{(1)} \) (LE constraint) we change into the equality by adding a slack variable \( x_{B,i} \):
\[
N_i^{(1)} : \sum_{j=1}^{n_1} a_{ij}(x_N)_j + (x_B)_i = b_i, \quad i = 1, \ldots, r.
\]

Also, every inequality of the form \( N_i^{(2)} \) (GE constraint) we transform into the equality by subtracting a surplus variable \( (x_B)_i \):
\[
N_i^{(2)} : \sum_{j=1}^{n_1} a_{ij}(x_N)_j - (x_B)_i = b_i, \quad i = r + 1, \ldots, s.
\]

After these transformations we get the equivalent linear program into the standard form
\[
\begin{align*}
\text{Maximize} & \quad c_1 x_1 + \cdots + c_n x_n + d = c^T x + d, \quad c_i = 0, i > n_1 \\
\text{subject to} & \quad Ax = b, \\
& \quad b = (b_1, \ldots, b_m), \\
& \quad x = ((x_N)_1, \ldots, (x_N)_n, (x_B)_1, \ldots, (x_B)_s), \\
& \quad (x_N)_j \geq 0, \quad j = 1, \ldots, n, \quad (x_B)_i \geq 0, \quad i = 1, \ldots, s,
\end{align*}
\]

where the real matrix \( A \) is of the order \( m \times n \), for \( n = n_1 + s \).

The paper is organized as follows. In the second section we describe two-phases revised simplex method which is analogous with corresponding two-phases simplex method from [3], [4], [6] and [7]. In the third section is described an algorithm for finding the first basic feasible solution. In the last section we arrange several Netlib test problems.

### 2 The revised simplex method

By \( A_i \) we denote \( i \)th column of \( A \), \( i = 1, \ldots, n \). We suppose that matrix \( A \) is of full rank (\( \text{rank}(A) = m \)), so there exists a base \( \{A_{i_k} \mid k = 1, \ldots, m \} \) in \( \mathbb{R}^m \). The matrix \( B = [A_{i_1}, \ldots, A_{i_m}] \) containing corresponding columns from \( A \) is called basic matrix. Variables \( x_{i_1}, \ldots, x_{i_m} \) are called basic variables, and the vector containing basic variables is denoted by \( x_B \). The \( m \times (n - m) \) matrix containing columns from \( A \) corresponding to nonbasic variables is denoted by \( N \), and the \( (n - m) \) vector containing nonbasic variables is denoted by \( x_N \). Column \( T_i \) of Tucker table \( T \) are representations of vector \( A_i \) in the base \( B \), i.e. \( T = B^{-1}N \).

Here the target function is represented as equality \( c_1 x_1 + \cdots + c_n x_n + d = -x_c \) where \( x_c \) is a basic variable representing a value of goal function. After corresponding modifications of the corresponding algorithm from [3], [4], [6], [7], we state the following two phases revised simplex algorithm. In this algorithm the notation \( A^i \) means \( i \)th row of the matrix \( A \).

**Algorithm 1.** (Revised simplex method for basic feasible solution.)
A modification of revised simplex method

Step S1A. Reconstruct the vector $b_T = B^{-1}b$, Tucker table $T = B^{-1}N$ and target function $c_T = T^{n+1}$.

Step S1B. If $(c_T)_1, \ldots, (c_T)_n \leq 0$, then the basic solution is an optimal solution.

Step S1C. Choose an arbitrary $(c_T)_j > 0$.

Step S1D. If $t_{1j}, \ldots, t_{mj} \leq 0$, stop the algorithm. Maximum is $+\infty$.

Otherwise, go to the next step.

Step S1E. Compute

$$\min_{1 \leq i \leq m} \left\{ \frac{(b_T)_i}{t_{ij}} \mid a_{ij} > 0 \right\} = \frac{(b_T)_p}{t_{pj}},$$

replace nonbasic and basic variables $(x_N)_j$ and $(x_B)_p$, respectively, and go to Step S1B.

If the condition $(b_T)_1, \ldots, (b_T)_m \geq 0$ is not satisfied, we use the following algorithm to search for the first basic feasible solution. Algorithm in this version is not found in the literature.

Algorithm 2. (Find the first basic feasible solution).

Step S1. Reconstruct the vector $b_T = B^{-1}b$, Tucker table $T = B^{-1}N$.

Step S2. Select the last $(b_T)_i < 0$.

Step S3. If $t_{i1}, \ldots, t_{in} \geq 0$ then STOP. Linear program can not be solved.

Step S4. Otherwise, find $t_{ij} < 0$, compute

$$\min_{k > i} \left\{ \frac{(b_T)_i}{t_{ij}} \cup \frac{(b_T)_k}{t_{kj}} \mid t_{kj} > 0 \right\} = \frac{(b_T)_p}{t_{pj}},$$

replace nonbasic and basic variables $(x_N)_j$ and $(x_B)_p$, respectively and goto Step S1. We use the last $t_{ij} < 0$.

3 Modification of algorithm for finding first basic solution

The problem of the replacement of a basic and a nonbasic variable in the general revised simplex method is contained in Step S4. We observed two drawbacks of Step S4.

1. If $p = i$ and if there exists index $y < i = p$ such that

$$\frac{(b_T)_y}{t_{yj}} < \frac{(b_T)_p}{t_{pj}}, \quad (b_T)_y > 0, \quad t_{yj} > 0$$

in the next iteration $(x_B)_y$ becomes negative:

$$(x_B)_y^1 = (b_T)_y^1 = (b_T)_y - \frac{(b_T)_p}{t_{pj}} t_{yj} < (b_T)_y - \frac{(b_T)_y}{t_{yj}} t_{yj} = 0.$$
2. If $p > i$, in the next iteration $(b_T)_i$ is negative:

$$
\frac{(b_T)_p}{t_{pj}} < \frac{(b_T)_i}{t_{ij}} \Rightarrow (b_T)^1_i = (b_T)_i - \frac{(b_T)_p}{t_{pj}} t_{ij} < 0.
$$

We can find solution of this problem in two ways.

For fixed $(b_T)_i$ try to find $t_{ij} < 0$ such that

$$
\min_k \left\{ \frac{(b_T)_k}{t_{kj}} \mid t_{kj} < 0 \right\} \cup \left\{ \frac{(b_T)_k}{t_{kj}} \mid t_{kj} > 0, (b_T)_k > 0 \right\} = \frac{(b_T)_i}{t_{ij}}.
$$

In this case, it is possible to choose $t_{ij}$ for the pivot element and obtain a new solution with less number of negative $(b_T)_i$.

This way is good for the basic simplex but not for the revised simplex because we need to reconstruct columns corresponding to all $t_{ij} < 0$.

In the second one, for fixed $t_{ij} < 0$ where $(b_T)_i < 0$ the main idea is to find a minimum

$$
\frac{(b_T)_p}{t_{pj}} < \min \left\{ \frac{(b_T)_k}{t_{kj}} \mid (b_T)_k < 0, t_{kj} < 0, k = 1, \ldots, m \right\}
$$

and if relation (3.1) is valid then choosing $(b_T)_p > 0$ as pivot element we obtain a new solution with less number of negative $(b_T)_i$.

For this purpose, we propose a modification of Step S4. This modification follows from the following lemma.

**Lemma 3.1** Let the problem (1.2) be feasible and let $x$ be the basic infeasible solution with $q$ negative coordinates. There exists $t_{ij} < 0$, and in the following two cases:

a) $q = m$, and

b) $q < m$ and

$$
Neg = \left\{ \frac{(b_T)_k}{t_{kj}} \mid (b_T)_k < 0, t_{kj} < 0, k = 1, \ldots, m \right\},
$$

$$
Pos = \left\{ \frac{(b_T)_k}{t_{kj}} \mid (b_T)_k > 0, t_{kj} > 0, k = 1, \ldots, m \right\}
$$

(3.2) \( \min (Neg \cup Pos) = \frac{b_T,r}{t_{rj}} \in Neg \)

it is possible to produce the new basic solution with at most $q - 1$ negative coordinates in only one iterative step of the revised simplex method, if we choose $t_{rj}$ for the pivot element, i.e. replace nonbasic variable $(x_N)_j$ with the basic variable $(x_B)_r$.

**Proof.** For every $(b_T)_i$, if all $t_{ij} > 0$ then $-(x_B)_i = t_{i1}(x_N)_1 + \cdots + t_{im}(x_N)_m - (b_T)_i > 0$. That is contradiction because for all $(x_N)_k > 0$ we have $(x_B)_i < 0$ and problem (1.2) is feasible.
a) If \( q = m \), for pivot element \( t_{ij} < 0 \) we get a new solution with at least one coordinate positive:

\[
(x_B)^1_i = (b_T)^1_i = \frac{(b_T)_i}{t_{ij}} > 0.
\]

b) Assume now that the conditions \( q < m \) and (3.1) are satisfied. Choose \( t_{rj} \) for the pivot element.

For \((b_T)_k > 0\) and \( t_{kj} < 0 \) it is obvious that

\[
(b_T)^1_k = (b_T)_k - \frac{(b_T)_r}{t_{rj}} t_{kj} > (b_T)_k \geq 0.
\]

For \((b_T)_k > 0\) and \( t_{kj} > 0 \), using \( \frac{(b_T)_k}{t_{kj}} \geq \frac{(b_T)_r}{t_{rj}} \), we conclude immediately

\[
(b_T)^1_k = b_T,k - \frac{(b_T)_r}{t_{rj}} t_{kj} \geq 0.
\]

Hence, all positive \((b_T)_k\), it remains positive. Moreover, for \((b_T)_r < 0\) we get

\[
(b_T)^1_r = \frac{(b_T)_r}{t_{rj}} \geq 0,
\]

which completes the proof.

\[\square\]

**Remark 3.1** Let the condition (3.2) is not satisfied, \( \min (\text{Neg} \cup \text{Pos}) = \frac{(b_T)_r}{t_{rj}} \in \text{Neg} \). If we choose \( t_{rj} \) for pivot element we will also have

\[
(b_T)^1_k = (b_T)_k - \frac{(b_T)_r}{t_{rj}} t_{kj} \geq 0.
\]

for \((b_T)_k > 0\) and \( t_{rj} \) both positive or negative. But for negative \((b_T)_k > 0\) we can prove similarly that

\[
(b_T)^1_k = (b_T)_k - \frac{(b_T)_r}{t_{rj}} t_{kj} < 0.
\]

for both \( t_{rj} \) positive or negative. Finally new solution has the same number of negative coordinates \( q \).

**Algorithm 3.** (Modification of Algorithm 2).

**Step SM1.** Let \( B \) is basic and \( N \) non-basic matrix. Reconstruct vector \( b_T \).

**Step SM2.** Construct the set

\[
B = \{(b_1)_{1}, \ldots, (b_q)_n\} = \{(b_i)_k \mid (b_i)_k < 0, \ k = 1, \ldots, q\}.
\]

**Step SM3.** Select \( (b_i)_s < 0 \).
Step SM4. Reconstruct \( r_i \)th row. If \( t_{i,1}, \ldots, t_{i,n} \geq 0 \) then STOP. Linear program is not solvable.

Otherwise, choose first \( t_{i,j} \leq 0 \).

Step SM5. Reconstruct \( j \)th column \( T_j \). Compute

\[
\min_{1 \leq i \leq n} \left\{ \frac{(b^T)_i}{t_{ij}} \mid (b^T)_i t_{ij} > 0, i = 1, \ldots, m \right\} = \frac{(b^T)_p}{t_{pj}},
\]

replace nonbasic and basic variables \((x_N)_j\) and \((x_B)_p\) and go to Step SM2.

4 Implementation and numerical experience

In previous algorithms the reconstruction of the Tucker table is based on the matrix inversion. We also require only few rows and columns of Tucker table. Next we will show how to reconstruct a simple row or column of Tucker table using linear equation solver. It is well known that it is much more efficient to solve a linear system than inverting a matrix [2]. Reconstruction of \( i \)th column \( T_i \) is simply: \( BT_i = A_i \). To find a row \( R^i \) we need to know \((B^{-1})^i\), so:

\[
T_i = (B^{-1})^i N
\]

Vector \((B^{-1})^i\) can be found from \( B^{-1} B = I \), which implies \((B^{-1})^i B = I^i\). If we apply transpose on left and right side we get: \( B^T ((B^{-1})^i)^T = (I^i)^T \). This method is implemented in following algorithm :

\[\text{Algorithm 4. Reconstruction of row of Tucker table } T^i.\]

\begin{itemize}
  \item \textbf{Step 1.} Let \( B \) be basic, \( N \) non-basic, and \( I \) identity matrix. Solve a system \( B^T((B^{-1})^i)^T = (I^i)^T \)
  \item \textbf{Step 2.} Compute and return \( T^i = (B^{-1})^i N. \)
\end{itemize}

Revised simplex and modification are implemented in program RevMarPlex in program language MATHEMATICA[8] and tested on some Netlib test problems. Results are compared with robust LP solver PCx.

\[ \text{From the columns two and three in Table 1 we can see that our results are in accordance with the results obtained by PCx. In the columns four and five of Table 1 we give the number of iterations needed for the construction of the first basis feasible solution, using Algorithm 3 and Algorithm 2, respectively.} \]
A modification of revised simplex method

Table 1.

<table>
<thead>
<tr>
<th>Problem</th>
<th>PCx</th>
<th>RevMarPlex</th>
<th>Alg3</th>
<th>Alg2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adlittle</td>
<td>$2.25494963 \times 10^7$</td>
<td>225494.963162</td>
<td>32</td>
<td>39</td>
</tr>
<tr>
<td>Ajro</td>
<td>$-4.64753143 \times 10^4$</td>
<td>$-464.753142$</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>Agg</td>
<td>$3.59917673 \times 10^7$</td>
<td>$-35991767.286576$</td>
<td>151</td>
<td>151</td>
</tr>
<tr>
<td>Agg2</td>
<td>$-2.0239251 \times 10^7$</td>
<td>$-20239252.355976$</td>
<td>75</td>
<td>129</td>
</tr>
<tr>
<td>Blend</td>
<td>$-3.08121498 \times 10^4$</td>
<td>$-30.812150$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sc105</td>
<td>$-5.2202061212 \times 10^7$</td>
<td>$-52.202061$</td>
<td>552</td>
<td>&gt; 1500</td>
</tr>
<tr>
<td>Sc205</td>
<td>$-5.2202061212 \times 10^7$</td>
<td>$-52.202061$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sc50a</td>
<td>$-6.4575077059 \times 10^7$</td>
<td>$-64.575077$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Sc50b</td>
<td>$-7.000000000 \times 10^4$</td>
<td>-70</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Scagr25</td>
<td>$-1.47534331 \times 10^7$</td>
<td>$-14753433.060769$</td>
<td>520</td>
<td>&gt; 1500</td>
</tr>
<tr>
<td>Scagr7</td>
<td>$-2.33138982 \times 10^6$</td>
<td>$-2331389.823330$</td>
<td>555</td>
<td>74</td>
</tr>
<tr>
<td>Stocforl</td>
<td>$-4.1131976219 \times 10^4$</td>
<td>$-41131.976219$</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>LitVera</td>
<td>$1.999992 \times 10^{-2}$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Kb2</td>
<td>$-1.7499 \times 10^7$</td>
<td>$-1749.9001299062$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Recipe</td>
<td>$-2.6661610^{2}$</td>
<td>$-266.6160$</td>
<td>13</td>
<td>15</td>
</tr>
<tr>
<td>Share1B</td>
<td>$-765893186 \times 10^{2}$</td>
<td>$76589.3185791859$</td>
<td>498</td>
<td>&gt; 1500</td>
</tr>
</tbody>
</table>

After the observation of these columns, it is easy to see that Algorithm 3 is faster with respect to Algorithm 2 in all cases. A streak in the corresponding position means that there are no iterative steps before the first basic feasible solution.

Also, RevMarPlex solved examples taken from [1], [5] which PCx is unable to solve (it ends with UNKNOWN status). Results are presented in Table 2. In all cases numbers of iterations of Algorithm 2 and Algorithm 3 were the same.

Table 2.

<table>
<thead>
<tr>
<th>Problem</th>
<th>RevMarPlex</th>
<th>Alg3 - Alg2</th>
</tr>
</thead>
<tbody>
<tr>
<td>07-20-02</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>15-30-03</td>
<td>$2.16968 \times 10^{-5}$</td>
<td>2</td>
</tr>
<tr>
<td>15-30-04</td>
<td>$2.16968 \times 10^{-5}$</td>
<td>2</td>
</tr>
<tr>
<td>15-30-06</td>
<td>$4.90359 \times 10^{-9}$</td>
<td>2</td>
</tr>
<tr>
<td>15-30-07</td>
<td>$3.2807 \times 10^{-4}$</td>
<td>3</td>
</tr>
<tr>
<td>15-60-07</td>
<td>$-6.29305 \times 10^{-7}$</td>
<td>3</td>
</tr>
<tr>
<td>15-60-09</td>
<td>$-6.29305 \times 10^{-7}$</td>
<td>3</td>
</tr>
<tr>
<td>20-40-05</td>
<td>$1.6394 \times 10^{-6}$</td>
<td>3</td>
</tr>
<tr>
<td>30-60-05</td>
<td>$1.6456 \times 10^{-7}$</td>
<td>-</td>
</tr>
<tr>
<td>30-60-08</td>
<td>$5.22527 \times 10^{-9}$</td>
<td>3</td>
</tr>
<tr>
<td>LitVera</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

RevMarPlex (LinearSolve function) indicates that matrices of these examples are badly conditioned (with the condition number between $10^{15}$ and $10^{20}$). Nevertheless RevMarPlex solved these examples giving solutions close to optimal.
References


