Research Article

On the Simplex Algorithm Initializing

Nebojša V. Stojković, Predrag S. Stanimirović, Marko D. Petković, and Danka S. Milojković

1 Faculty of Economics, University of Niš, Trg Kralja Aleksandra 11, 18000 Niš, Serbia
2 Department of Computer Science, Faculty of Sciences, University of Niš, Višegradska 33, 18000 Niš, Serbia
3 LEDIB Programme Component Coordinator, Nisava District, 18000 Niš, Serbia

Correspondence should be addressed to Nebojša V. Stojković, nebojsas@eknfak.ni.ac.rs

Received 15 June 2012; Revised 19 July 2012; Accepted 27 July 2012

© 2012 Nebojša V. Stojković et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper discusses the importance of starting point in the simplex algorithm. Three different methods for finding a basic feasible solution are compared throughout performed numerical test examples. We show that our two methods on the Netlib test problems have better performances than the classical algorithm for finding initial solution. The comparison of the introduced optimization softwares is based on the number of iterative steps and on the required CPU time. It is pointed out that on average it takes more iterations to determine the starting point than the number of iterations required by the simplex algorithm to find the optimal solution.

1. Introduction

There are two main methods for solving linear programming problem: the Simplex method and the interior point method. In both of these two methods it is necessary to determine the initial point. It is known that the application of the simplex algorithm requires at least one basic feasible solution. For the interior point method, in most cases it is enough that the point belongs to the so-called central path. Note also that in most algorithms based on interior point methods, the starting point need not to be feasible. However, in practical implementation it is shown that the numerical stability of the algorithm still depends on the choice of the starting point. These issues are investigated in papers [1, 2].

Two classical algorithms for generating the initial point for the Simplex algorithm are the two-phase Simplex algorithm and the Big-M method. The main drawback of these algorithms is requiring the introduction of artificial variables, increasing the dimension of the problem.
From that reason, algorithms which do not require the introduction of artificial variables are developed. A heuristic for finding the starting point of a linear programming problem, which is based on the method of minimal angles, is proposed in [3]. Unfortunately, these heuristics can be used only for certain classes of problems. Similar heuristics are investigated in the papers [4–6]. Different versions of the algorithm which does not introduce artificial variables are given in the papers [7–12]. One variant of these algorithms is introduced and discussed in the paper [11], which was the inspiration for our paper. We compare three different methods for finding an initial basic feasible solution (the starting point for the Simplex algorithm) throughout performed numerical test examples. Two methods, called M1 and M2, are introduced in [13], while the third method, called NFB, is established in [11].

2. The Simplex Method and Modifications

Consider the linear programming (LP) problem in the standard matrix form:

\[
\text{Maximize } c^T x - d, \\
\text{subject to } A x = b, \\
x \geq 0,
\]

where \( A \in \mathbb{R}^{m \times (m+n)} \) is the full row rank matrix (\( \text{rank}(A) = m \)), \( c \in \mathbb{R}^{n+m} \), and the system \( A x = b \) is defined by \( x \in \mathbb{R}^{m+n} \), \( b \in \mathbb{R}^m \). It is assumed that \((i, j)\)-th entry in \( A \) is denoted by \( a_{ij} \), \( b = (b_1, \ldots, b_m)^T \), \( d \in \mathbb{R} \), \( x^T = (x_1, \ldots, x_{n+m}) \), and \( c^T = (c_1, \ldots, c_{n+m}) \).

Then we choose \( m \) basic variables \( x_{B,1}, \ldots, x_{B,m} \), express them as a linear combination of nonbasic variables \( x_{N,1}, \ldots, x_{N,n} \), and obtain the canonical form of the problem (2.1). We write this canonical form in Table 1.

Coefficients of the transformed matrix \( A \) and the transformed vector \( c \) are again denoted by \( a_{ij} \) and \( c_j \), respectively, without loss of generality.

For the sake of completeness we restate one version of the two-phase maximization simplex algorithm from [12, 14] for the problem (2.1), represented in Table 1.

Within each iteration of the simplex method, exactly one variable goes from nonbasic to basic and exactly one variable goes from basic to nonbasic. Usually there is more than one choice for the entering and the leaving variables. The next algorithm describes the move from the current base to the new base when the leaving-basic and entering-nonbasic variables have been selected.
Algorithm 2.1 (consider interchange a basic variable $x_{B,p}$ and nonbasic variable $x_{N,j}$).

$$a_{ql} = \begin{cases} \frac{1}{a_{pj}}, & q = l, \ l = j \\ \frac{a_{pl}}{a_{pj}}, & q = p, \ l \neq j \\ \frac{a_{ql}}{a_{pj}}, & q \neq p, \ l = j \\ \frac{a_{ql} - a_{pl}a_{qj}}{a_{pj}}, & q \neq p, \ l \neq j \end{cases}, \quad b_{l} = \begin{cases} \frac{b_{p}}{a_{pj}}, & l = p \\ \frac{b_{l} - \frac{b_{p}}{a_{pj}}a_{ij}}{a_{pj}}, & l \neq p \end{cases},$$

$$c_{l} = \begin{cases} c_{l} - \frac{c_{j}a_{pl}}{a_{pj}}, & l \neq j, \\ \frac{c_{j}}{a_{pj}}, & l = j \end{cases}, \quad d_{l} = d - \frac{b_{p}c_{j}}{a_{pj}}. \quad (2.2)$$

The next algorithm finds an optimal solution of the LP problem when the condition $b_{1}, \ldots, b_{m} \geq 0$ is satisfied.

**Algorithm 2.2.**

*Step S1A.* If $c_{1}, \ldots, c_{n} \leq 0$, then the basic solution is an optimal solution.

*Step S1B.* Choose $c_{j} > 0$ according to the Bland’s rule [15].

*Step S1C.* If $a_{1j}, \ldots, a_{mj} \leq 0$, stop the algorithm. Maximum is $+\infty$.

Otherwise, go to the next step.

*Step S1D.* Compute

$$\min_{1 \leq i \leq m} \left\{ \frac{b_{i}}{a_{ij}} \mid a_{ij} > 0 \right\} = \frac{b_{p}}{a_{pj}}. \quad (2.3)$$

If the condition $b_{1}, \ldots, b_{m} \geq 0$ is not satisfied, we use the algorithm from [12, 14] to search for the initial basic feasible solution. In contrast to approach used in [16], it does not use artificial variables and therefore does not increase the size of the problem. It is restated here as the following Algorithm 2.3.

**Algorithm 2.3.**

*Step S2.* Select the last $b_{l} < 0$.

*Step S3.* If $a_{1i}, \ldots, a_{mi} \geq 0$, then STOP. LP problem is infeasible.

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

$$\min_{k > l} \left( \left\{ \frac{b_{k}}{a_{ki}} \right\} \cup \left\{ \frac{b_{k}}{a_{kj}} \mid a_{kj} > 0 \right\} \right) = \frac{b_{p}}{a_{pj}}. \quad (2.4)$$

choose $x_{N,j}$ as the entering-nonbasic variable, $x_{B,p}$ as the leaving-basic variable, apply Algorithm 2.1, and go to Step S2.
Note that Algorithm 2.3 and the NFB algorithm from [11] are equivalent. As shown in the paper [13], the lack of the algorithm from [12] is that when it is applied to transform the selected coordinate from the basic solution into a positive, it can increase the number of negative coordinates.

The problem of selecting a leaving-basic variable and corresponding entering-non-basic variable in the two-phase simplex method is contained in Step S1D of Algorithm 2.2 and Step S4 of Algorithm 2.3. We observed two drawbacks of Step S4. By \( i \) we denote the index of the last negative \( b_i \).

(1) If \( p = i \) for each index \( t < i = p \) such that

\[
\frac{b_t}{a_{ij}} < \frac{b_p}{a_{pj}}, \quad b_t > 0, \quad a_{ij} > 0
\]

in the next iteration \( x_{B,t} \) becomes negative:

\[
x_{B,t}^1 = b_{t}^1 = b_t - \frac{b_p}{a_{pj}} a_{ij} < b_t - \frac{b_i}{a_{ij}} a_{ij} = 0.
\]

(2) If \( p > i \), in the next iteration \( b_i^1 \) is negative:

\[
\frac{b_p}{a_{pj}} < \frac{b_i}{a_{ij}} \Rightarrow b_i^1 = b_i - \frac{b_p}{a_{pj}} a_{ij} < 0.
\]

Although there may exist \( b_t < 0, \ t < i \) such that

\[
\min_{k>t} \left( \left\{ \frac{b_t}{a_{ij}}, \ a_{ij} < 0 \right\} \cup \left\{ \frac{b_k}{a_{kj}}, \ a_{kj} > 0, \ b_k > 0 \right\} \right) = \frac{b_t}{a_{ij}}.
\]

In such case, it is possible to choose \( a_{ij} \) for the pivot element and obtain

\[
x_{B,t} = b_t^1 = \frac{b_t}{a_{ij}} \geq 0.
\]

Also, since \( b_t/a_{ij} \leq b_k/a_{kj} \), each \( b_k > 0 \) remains convenient for the next basic feasible solution:

\[
x_{B,k} = b_k^1 = b_k - \frac{b_t}{a_{ij}} a_{kj} \geq 0.
\]

Therefore, it is possible that the choice of entering and leaving variable defined by Step S4 reduces the number of positive \( b \)'s after the application of Algorithm 2.1. Our goal is to obviate the observed disadvantages in Step S4. For this purpose, we propose a modification of Step S4, which gives a better heuristic for the choice of basic and nonbasic variables.
Proposition 2.4 (see [13]). Let the problem (2.1) be feasible and let $x$ be the basic infeasible solution with $b_{i_1}, \ldots, b_{i_q} < 0$, $q \leq m$. Consider the index set $I = \{i_1, \ldots, i_q\}$.

The following statements are valid.

1. It is possible to produce a new basic solution $x^1 = \{x^1_B, \ldots, x^1_{B,m}\}$ with at most $q - 1$ negative coordinates in only one step of the simplex method in the following two cases:
   (a) $q = m$,
   (b) $q < m$ and there exist $r \in I$ and $s \in \{1, \ldots, n\}$ such that
   \[
   \min_{h \in I} \left\{ \frac{b_h}{a_{hs}} \mid a_{hs} > 0 \right\} \geq \frac{b_r}{a_{rs}}, \quad a_{rs} < 0.
   \] (2.11)

2. It is possible to produce a new basic solution $x^1 = \{x^1_B, \ldots, x^1_{B,m}\}$ with exactly $q$ negative coordinates in one step of the simplex method if neither condition (a) nor condition (b) is valid.

Proof. We restate the proof from [13] for the sake of completeness.

1. (a) If $q = m$, for an arbitrary pivot element $a_{js} < 0$, we get a new basic solution with at least one positive coordinate:
   \[
   x^1_{B,j} = b^1_j = \frac{b_j}{a_{js}} > 0.
   \] (2.12)

The existence of negative $a_{js}$ is ensured by the assumption that the problem (2.1) is feasible.

(b) Now assume that the conditions $q < m$, $r \in I$, and (2.11) are satisfied. Choose $a_{rs}$ for the pivot element and apply Algorithm 2.1. Choose arbitrary $b_k \geq 0$, $k \neq r$.

In the case $a_{ks} < 0$ it is obvious that
\[
 x^1_{B,k} = b^1_k = \frac{b_k}{a_{ks}} a_{ks} \geq b_k \geq 0.
 \] (2.13)

In the case $a_{ks} > 0$, using $b_k / a_{ks} \geq b_r / a_{rs}$, we conclude immediately
\[
 x^1_{B,k} = b^1_k = b_k - \frac{b_r}{a_{rs}} a_{ks} \geq b_k - \frac{b_k}{a_{ks}} a_{ks} = 0.
 \] (2.14)

On the other hand, for $b_r < 0$ we obtain from Algorithm 2.1
\[
 b^1_r = \frac{b_r}{a_{rs}} \geq 0.
 \] (2.15)

Therefore, all nonnegative $b_k$ remain nonnegative and $b_r < 0$ becomes nonnegative.

(2) If neither condition (a) nor condition (b) is valid, let $r \notin I$ and $s \in \{1, \ldots, n\}$ be such that
\[
 \min_{h \notin I} \left\{ \frac{b_h}{a_{hs}} \mid a_{hs} > 0 \right\} = \frac{b_r}{a_{rs}}.
 \] (2.16)
By choosing $a_{rs}$ as the pivot element and by applying the transformations defined in Algorithm 2.1 we obtain the same number of negative elements in the vector $b$. This fact can be proved similarly as the part 1 (b).

Remark 2.5. From Proposition 2.4 we get three proper selections of the pivot element in Step S4:

(i) arbitrary $a_{js} < 0$ in the case $q = m$;
(ii) arbitrary $a_{rs} < 0$ satisfying (2.11) when the conditions $0 < q < m$, $r \in I$ are satisfied;
(iii) arbitrary $a_{rs} > 0$ satisfying (2.16) when $0 < q < m$, and there is no $a_{rs} < 0$ satisfying conditions in the previous case.

In accordance with Proposition 2.4 and considerations in Remark 2.5, we propose the following improvement of Algorithm 2.3.

Algorithm M1 (modification of Algorithm 2.3)

Step 1. If $b_1, \ldots, b_m \geq 0$, perform Algorithm 2.2. Otherwise continue.
Step 2. Select the first $b_i < 0$.
Step 3. If $a_{i,1}, \ldots, a_{i,n} \geq 0$, then STOP. LP problem is infeasible. Otherwise, construct the set

\[ Q = \{ a_{i,p} < 0, \; p = 1, \ldots, t \}, \tag{2.17} \]

initialize variable $p$ by $p = 1$ and continue.
Step 4. Compute

\[
\min_{1 \leq h \leq m} \left\{ \frac{b_h}{a_{h,j_p}} \mid a_{h,j_p} > 0, b_h > 0 \right\} = \min_{h \notin I} \left\{ \frac{b_h}{a_{h,j_p}} \mid a_{h,j_p} > 0 \right\} = \frac{b_r}{a_{r,j_p}}, \tag{2.18}
\]

Step 5. If $b_i / a_{i,j_p} \leq b_h / a_{h,j_p}$, then interchange entering-nonbasic variable $x_{N,j_p}$ and leaving-

basic variable $x_{B,j_p}$ (apply Algorithm 2.1) and go to Step 1. Otherwise go to Step 6.
Step 6. If $p > t$, interchange $x_{N,j_p}$ and $x_{B,r}$ (apply Algorithm 2.1) and go to Step 1. Otherwise, put $p = p + 1$ and go to Step 3.

Algorithm M1 chooses one fixed (the first) value $b_i < 0$ satisfying conditions of Proposition 2.4. But there may exist some other $b_i < 0$ such that conditions of Proposition 2.4 are satisfied, and in the next iteration we can obtain a basic solution with smaller number of negative $b$'s. From that reason, in [13] we gave improved version of Algorithm M1, named M2.

Algorithm M2

Step 1. If $b_1, \ldots, b_m \geq 0$, perform Algorithm 2.2. Otherwise, construct the set

\[ B = \{ i_k \mid b_{i_k} < 0, \; k = 1, \ldots, q \}. \tag{2.19} \]
Step 2. Set \( s = 1 \) and perform the following.

Step 2.1. If \( a_{i,1}, \ldots, a_{i,n} \geq 0 \) then STOP. LP problem is infeasible.
Otherwise, construct the set

\[
Q = \left\{ a_{i,p} < 0, \ p = 1, \ldots, t \right\},
\]  
(2.20)

put \( p = 1 \), and continue.

Step 2.2. Find the minima:

\[
p' = \arg\min \left\{ \frac{b_k}{a_{k,p}} \mid b_k < 0, \ a_{k,j_p} < 0 \right\},
\]

\[
M(j) = \min \left\{ \frac{b_k}{a_{k,j_p}} \mid b_k > 0, \ a_{k,j_p} > 0 \right\}.
\]
(2.21)

If \( b_k / a_{k,j_p} \leq M(j_p) \), then choose \( a_{p',j_p} \) for the pivot element, apply Algorithm 2.1, and go to Step 1.
(In the next iteration \( b_k \) becomes positive.)

Step 2.3. If \( p < t \), then put \( p = p + 1 \) and go to Step 2.2, otherwise continue.

Step 2.4. If \( s < q \), then put \( s = s + 1 \) and go to Step 2.1, otherwise continue.

Step 3. If \( q < m \) and there do not exist \( r \in I \) and \( s \in \{1, \ldots, n\} \) such that

\[
\min_{h \in I} \left\{ \frac{b_h}{a_{hs}} \mid a_{hs} > 0 \right\} \geq \frac{b_r}{a_{rs}}, \ a_{rs} < 0,
\]
then move to the following.

\[
\min_{h \in I} \left\{ \frac{b_h}{a_{hs}} \mid a_{hs} > 0 \right\} \geq \frac{b_r}{a_{rs}}, \ a_{rs} < 0,
\]
(2.22)

Step 3.1. Select \( j_0 = \arg\min \{ x_{N,j} \mid a_{q,j} < 0 \} \).

Step 3.2. Compute

\[
p'' = \arg\min \left\{ x_{B,p} \mid \frac{b_p}{a_{p,j_0}} = M(j_0) \right\}.
\]
(2.23)

Step 3.3. Choose \( a_{p'',j_0} \) for pivot element, apply Algorithm 2.1, and go to Step 1.
Algorithm M1 and Algorithm M2 are well defined. If there is no \( b_i < 0 \) such that the condition \((2.11)\) is valid, we choose pivot element according to Remark 2.5 to obtain a solution with the same number of negative \( b \)'s. To avoid the cycling in this case, we will present an anticycling rule which is based on the following result from [13].

**Proposition 2.6** (see [13]). Assume that there is no \( b_i < 0 \) such that the conditions \((2.11)\) of Proposition 2.4 are satisfied. After choosing the pivot element according to \((2.16)\), we obtain a new base where \( 0 > b_j^i \geq b_i \), holds for all \( i \in I \).

Since \( i \) in Step 2 of Algorithm M1 is fixed, Algorithm M1 may cycle only if \( b_i^1 = b_i \). For that reason, if the minimum in \((2.18)\) is not unique, we choose \( j \) according to the Bland’s rule which guarantees that the simplex method always terminates [15] and [17, Theorem 3.3]. Therefore, according to Proposition 2.6, finite number of iterations value of \( b_i \) will start to increase or we will conclude that the problem is infeasible \((a_{i,j} \) are positive for all \( j = 1, \ldots, n \)).

In the sequel we show that our methods have better performances than the Nabli’s NFB method from [11]. Algorithms M1 and M2 are implemented in the software package MarPlex, reused from [13] and tested on some standard linear programming test problems.

### 3. Results of Comparison

We already mentioned that we were motivated with the paper [11] to consider the number of iterations needed to find the starting point. This yields an exact indicator of the quality of the applied linear programming solvers. In the paper [11] that the quotient of the total number of iterations required to determine the initial solution and the number of variables is equal to \( 0.334581 = 268/801 \) for the NFB, \( 0.483249 = 439/985 \) for the two-phase algorithm and \( 0.445685 = 476/985 \) for the Big-M method. It is obvious that the NFB algorithm is better than the Two-phases and the Big-M methods with respect to this criterion.

Note that the results of the paper [11] are obtained on a small sample of test problems in which the number of variables is relatively small. To resolve this drawback, we applied the NFB algorithm to the Netlib collection test problems. In addition, we tested our two algorithms, marked with M1 and M2. In this paper we consider the results of practical application of these algorithms in accordance with the methodologies used in the papers [11, 18].

The comparison of different optimization softwares from [11] is based on the number of iterative steps. In the tables below, columns labeled by \( n \) contain the number of variables in the problem, while the columns marked with NFB, M1, and M2 contain the number of iterations that is required for the corresponding algorithm. The names of the Netlib problems are given in the column LP. The number of iterations that are necessary to find initial basic feasible solutions in particular test examples is presented in Table 2.

The quotient of the total number of iterations required to determine the initial solution Simplex algorithm and the total number of variables is equal to \( 32856/52509 = 0.625721 \) for the NFB algorithm, for the Algorithm M1 is equal to \( 27005/52509 = 0.514293 \), and for the M2 method is \( 16118/52509 = 0.306957 \). For the NFB algorithm we get a higher value compared to the result from [11], as consequence of the larger size and the computational effort to solve the tested problems. Algorithm M1 shows a slightly better efficiency, while M2 produces significantly smaller quotient.

In Table 3 we give comparative results of the total number of iterations required to solve the same test problems.
Table 2: Number of iterations for generating the initial solutions computation.

| LP         | n  | NFB | M1 | M2 |
|------------|----|-----|----|----|
| adlittle   | 97 | 77  | 21 | 57 |
| agg        | 615| 84  | 38 | 67 |
| agg3       | 758| 141 | 51 | 71 |
| beaconfd   | 295| 1   | 1  | 1  |
| brandy     | 303| 2248| 624| 1276|
| czprob     | 3562| 11261| 6824| 6933|
| etamacro   | 816| 185 | 162| 191|
| fit1d      | 1049| 1   | 1  | 1  |
| gfrd-pnc   | 1160| 240 | 126| 229|
| grow22     | 946| 1   | 1  | 1  |
| israel     | 316| 2   | 2  | 2  |
| lotfi      | 366| 339 | 111| 76 |
| sc105      | 163| 1   | 1  | 1  |
| sc50a      | 78 | 1   | 1  | 1  |
| scag7      | 185| 90  | 69 | 81 |
| scorpion   | 466| 114 | 70 | 90 |
| sctap2     | 2500| 739 | 246| 666|
| share2b    | 162| 123 | 92 | 135|
| ship04l    | 2166| 450 | 100| 8  |
| ship08l    | 4363| 461 | 144| 320|
| ship12l    | 5533| 938 | 232| 49 |
| sierra     | 2733| 65  | 82 | 75 |
| standata   | 1274| 76  | 146| 21 |
| stocfor1   | 165| 1   | 1  | 1  |
| afro       | 51 | 17  | 2  | 4  |
| agg2       | 758| 52  | 31 | 40 |
| bandm      | 472| 3128| 1495| 273|
| blend      | 114| 1   | 1  | 1  |
| capri      | 482| 214 | 1316| 251|
| e226       | 472| 5663| 395 | 215|
| finnis     | 1064| 276 | 809| 141|
| ganges     | 1706| 1   | 1  | 1  |
| grow15     | 645| 1   | 1  | 1  |
| grow7      | 301| 1   | 1  | 1  |
| kb2        | 68 | 1   | 1  | 1  |
| recipe     | 204| 8   | 9  | 9  |
| sc205      | 317| 1   | 1  | 1  |
| sc50b      | 78 | 1   | 1  | 1  |
| scf1x3m1   | 600| 2478| 311| 1133|
| sctap1     | 660| 496 | 131| 320|
| sctap3     | 3340| 618 | 369| 469|
| share1b    | 253| 366 | 368| 89 |
| shell      | 1777| 55  | 78 | 41 |
| ship04s    | 1506| 116 | 57 | 14 |
| ship08s    | 2467| 169 | 67 | 54 |
| ship12s    | 2869| 429 | 166| 55 |
| stair      | 614| 686 | 11066| 2450|
| standmps   | 1274| 260 | 752| 131|
| vtp.base   | 346| 179 | 430| 69 |
| Sum        | 52509| 32856| 27005| 16118|
Table 3: The total number of iterations.

| LP       | $n$  | NFB | M1  | M2  |
|----------|------|-----|-----|-----|
| adlittle | 97   | 115 | 76  | 101 |
| agg      | 615  | 115 | 63  | 89  |
| agg3     | 758  | 222 | 194 | 148 |
| beaconfd | 295  | 34  | 34  | 34  |
| brandy   | 303  | 2329| 714 | 1348|
| czprob   | 3562 | 11886| 7472| 7524|
| etamacro | 816  | 361 | 377 | 448 |
| fitd     | 1049 | 835 | 835 | 835 |
| gfrd-pnc | 1160 | 577 | 437 | 534 |
| grow22   | 946  | 3570| 3570| 3570|
| israel   | 316  | 159 | 159 | 159 |
| lotfi    | 366  | 397 | 248 | 204 |
| sc105    | 163  | 57  | 57  | 57  |
| sc50a    | 78   | 27  | 27  | 27  |
| scagr7   | 185  | 125 | 95  | 122 |
| scuron   | 466  | 151 | 140 | 128 |
| scrapt2  | 2500 | 934 | 816 | 942 |
| share2b  | 162  | 169 | 117 | 173 |
| ship04l  | 2166 | 574 | 474 | 259 |
| ship08l  | 4363 | 825 | 775 | 850 |
| ship12l  | 5533 | 2846| 1943| 1068|
| sierra   | 2733 | 490 | 392 | 401 |
| standata | 1274 | 174 | 262 | 159 |
| stocfor1 | 165  | 18  | 18  | 18  |
| afiro    | 51   | 22  | 11  | 12  |
| agg2     | 758  | 118 | 154 | 109 |
| bandm    | 472  | 3299| 1622| 432 |
| blend    | 114  | 733 | 733 | 733 |
| capri    | 482  | 352 | 1479| 371 |
| e226     | 472  | 6230| 759 | 533 |
| finnis   | 1064 | 584 | 1034| 516 |
| ganges   | 1706 | 421 | 421 | 421 |
| grow15   | 645  | 880 | 880 | 880 |
| grow7    | 301  | 241 | 241 | 241 |
| kb2      | 68   | 51  | 51  | 51  |
| recipe   | 204  | 37  | 37  | 39  |
| sc205    | 317  | 136 | 136 | 136 |
| sc50b    | 78   | 28  | 28  | 28  |
| scfxm1   | 600  | 2878| 434 | 1220|
| scapos   | 660  | 553 | 268 | 328 |
| sctap3   | 3340 | 865 | 1278| 721 |
| share1b  | 253  | 435 | 431 | 154 |
| shell    | 1777 | 334 | 356 | 317 |
| ship04s  | 1506 | 301 | 251 | 186 |
| ship08s  | 2467 | 408 | 339 | 312 |
| ship12s  | 2869 | 985 | 632 | 494 |
| stair    | 614  | 719 | 1234| 2492|
| standmps | 1274 | 415 | 824 | 240 |
| vtp.base | 346  | 250 | 477 | 124 |
| Sum      | 52509| 48265| 43405| 30288|
From Table 3 it is apparent that M2 is the best algorithm. It is interesting to note that the M1 algorithm requires less total number of iterations compared to the NFB, although the Simplex algorithm after applying Algorithm M1 requires more iterations. Hence the choice of algorithm for determining the starting point is very important for the overall efficiency of the software for solving the LP problems.

From Tables 2 and 3 it follows that the total number of iterations spent in the Simplex algorithm to generate the optimal solution is equal to $15409,16400,14170$ for the NFB, M1, and M2 methods, respectively. Therefore, it is clear that the total number of iterations required for the initial basic feasible solution is greater than the number of iterations required by the Simplex algorithm to find the optimal solution. This fact is very important, especially if we observe from Table 2 that about 25% of the considered problems are suitable for determining the starting point in one step. Taking into account the results of the paper [11] from which it follows that the number of iterations of the classical two-phase method for determining the starting point is significantly higher than the number of iterations of algorithms that do not introduce artificial variables, the fact becomes more important. This means that finding the starting point represents a very important part in solving LP problems.

The better performances of the M2 method primarily (and particularly the M1 method), compared to the NFB method, can also be confirmed by using the so-called performance profile, introduced in [18]. The underlying metric is defined by the number of iterative steps. Following the notations given in the paper [18] we have that the number of solvers is $n_s = 3$ (the NFB, M1, and M2) and the number of numerical experiments is $n_p = 49$. For the performance metrics we use the number of iterative steps. By $i_{p,s}$ we denote the number of iterations required to solve problem $p$ by the solver $s$. The quantity

$$r_{p,s} = \frac{i_{p,s}}{\min\{i_{p,s} : s \in \{\text{NFB, M1, M2}\}\}$$

is called the performance ratio. Finally, the performance of the solver $s$ is defined by the following cumulative distribution function

$$\rho_s(\tau) = \frac{1}{n_p} \text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}, \quad s \in \{\text{NFB, M1, M2}\},$$

where $\tau \in \mathbb{R}$ and $\mathcal{P}$ represents the set of problems. Figure 1 shows the performance profiles for NFB, M1, and M2 regarding the number of iterations. Figure 1(a) (resp., Figure 1(b)) illustrates data arranged in Table 2 (resp., Table 3).

It is clear from Figure 1(a), that M1 and M2 methods show better performances compared to NFB. Comparison of M1 and M2 algorithms gives somewhat contradictory results. Namely, the probability of being the optimal solver regarding the number of iterative steps required for generating the starting point is in favor of the M1 algorithm, which is confirmed by $\rho_{M1}(1) = 0.64 > \rho_{M2}(1) = 0.60 > \rho_{NFB}(1) = 0.38$. On the other hand, the probability $\rho_{M2}(\tau)$ of Algorithm M2 is greater than the probability $\rho_{M1}(\tau)$ for all values $\tau > 1$. On the other hand, $\rho_{M2}(\tau) > \rho_{NFB}(\tau)$ is always satisfied.

From Figure 1(b), it is evident that the inequality $\rho_{M2}(\tau) \geq \max\{\rho_{NFB}(\tau), \rho_{M1}(\tau)\}$ is always satisfied, which means that Algorithm M2 has the highest probability of being the optimal solver with respect to the total number of iterations required to solve the problem.

In sequel, the total CPU time is the comparison criteria. The required CPU time for NFB, M1 and M2 methods is given in Table 4. The time ratio and the number of
Table 4: The CPU time, the CPU time ratio, and the number of iteration ratios.

| LP            | Time NFB | Time M1 | Time M2 | Time ratio M1/NFB | Time ratio M2/NFB | No. it. ratio M1/NFB | No. it. ratio M2/NFB |
|----------------|----------|---------|---------|-------------------|-------------------|----------------------|----------------------|
| adlittle       | 1.19     | 0.94    | 0.74    | 0.789             | 0.621             | 0.660                | 0.878                |
| agg            | 0.71     | 0.61    | 0.68    | 0.859             | 0.957             | 0.547                | 0.773                |
| agg3           | 0.69     | 0.58    | 0.48    | 0.840             | 0.695             | 0.873                | 0.666                |
| beacnf        | 0.63     | 0.63    | 0.64    | 1                 | 1.015             | 1                    | 1                    |
| brandy         | 1.45     | 1.28    | 1.39    | 0.882             | 0.958             | 0.306                | 0.578                |
| cprob          | 37.2     | 45.3    | 185.3   | 1.217             | 4.981             | 0.628                | 0.633                |
| etamacro       | 0.46     | 0.53    | 0.59    | 1.152             | 1.282             | 1.044                | 1.241                |
| fit1d          | 1.45     | 1.45    | 1.45    | 1                 | 1                 | 1                    | 1                    |
| gfrd-pnc       | 0.59     | 0.39    | 0.75    | 0.661             | 1.271             | 0.757                | 0.925                |
| grow22         | 180.2    | 180.2   | 180.2   | 1                 | 1                 | 1                    | 1                    |
| israel         | 0.51     | 0.51    | 0.51    | 1                 | 1                 | 1                    | 1                    |
| lotfi          | 0.62     | 0.53    | 0.45    | 0.854             | 0.725             | 0.624                | 0.513                |
| sc105          | 0.55     | 0.55    | 0.55    | 1                 | 1                 | 1                    | 1                    |
| sc50a          | 0.42     | 0.42    | 0.42    | 1                 | 1                 | 1                    | 1                    |
| scag7          | 0.44     | 0.38    | 0.35    | 0.863             | 0.795             | 0.76                 | 0.976                |
| scorpion       | 0.44     | 0.35    | 0.37    | 0.795             | 0.840             | 0.9271               | 0.847                |
| sctap2         | 4.2      | 3.1     | 15.1    | 0.738             | 3.590             | 0.873                | 1.008                |
| share2b        | 0.65     | 0.44    | 0.43    | 0.676             | 0.661             | 0.692                | 1.023                |
| ship04l        | 0.41     | 0.73    | 0.35    | 1.780             | 0.853             | 0.825                | 0.451                |
| ship08l        | 4.1      | 6.9     | 10.1    | 1.682             | 2.463             | 0.939                | 1.03                 |
| ship12l        | 17.2     | 16.1    | 14.9    | 0.936             | 0.866             | 0.682                | 0.375                |
| sierra         | 4.9      | 4.5     | 5.5     | 0.918             | 1.122             | 0.8                  | 0.818                |
| standata       | 0.39     | 0.41    | 0.46    | 1.051             | 1.179             | 1.505                | 0.913                |
| stocfor1       | 0.57     | 0.57    | 0.57    | 1                 | 1                 | 1                    | 1                    |
| afiro          | 0.42     | 0.29    | 0.36    | 0.690             | 0.857             | 0.5                  | 0.545                |
| agg2           | 0.39     | 0.49    | 0.38    | 1.256             | 0.974             | 1.305                | 0.923                |
| bandm          | 12.2     | 5.8     | 2.5     | 0.475             | 0.204             | 0.491                | 0.130                |
| blend          | 0.52     | 0.52    | 0.52    | 1                 | 1                 | 1                    | 1                    |
| capri          | 0.53     | 4.1     | 0.42    | 7.735             | 0.792             | 4.201                | 1.053                |
| e226           | 12.7     | 0.56    | 0.39    | 0.044             | 0.030             | 0.121                | 0.085                |
| finnis         | 0.62     | 0.95    | 0.41    | 1.532             | 0.661             | 1.770                | 0.883                |
| ganges         | 0.77     | 0.77    | 0.77    | 1                 | 1                 | 1                    | 1                    |
| grow15         | 26.3     | 26.3    | 26.3    | 1                 | 1                 | 1                    | 1                    |
| grow7          | 2.1      | 2.1     | 2.1     | 1                 | 1                 | 1                    | 1                    |
| kb2            | 0.51     | 0.51    | 0.51    | 1                 | 1                 | 1                    | 1                    |
| recipe         | 0.37     | 0.39    | 0.44    | 1.054             | 1.189             | 1.054                | 1                    |
| sc205          | 0.42     | 0.42    | 0.42    | 1                 | 1                 | 1                    | 1                    |
| sc50b          | 0.41     | 0.41    | 0.41    | 1                 | 1                 | 1                    | 1                    |
| scfxm1         | 8.3      | 0.45    | 4.8     | 0.054             | 0.578             | 0.150                | 0.423                |
| sctap1         | 0.67     | 0.34    | 0.38    | 0.507             | 0.567             | 0.484                | 0.593                |
| sctap3         | 4.1      | 4.9     | 14.3    | 1.195             | 3.487             | 1.477                | 0.833                |
| share1b        | 0.47     | 0.49    | 0.34    | 1.042             | 0.723             | 0.990                | 0.354                |
| shell          | 0.46     | 0.51    | 0.64    | 1.108             | 1.391             | 1.065                | 0.949                |
| ship04s        | 0.68     | 0.35    | 0.34    | 0.514             | 0.5              | 0.833                | 0.617                |
| ship08s        | 1.5      | 1.2     | 1.4     | 0.8               | 0.933             | 0.830                | 0.764                |
| ship12s        | 2.4      | 2.1     | 2.5     | 0.875             | 1.041             | 0.641                | 0.501                |
| stair          | 3.5      | 96.1    | 33.2    | 27.457            | 9.485             | 15.624               | 3.465                |
| standmps       | 0.44     | 0.53    | 0.36    | 1.204             | 0.818             | 1.985                | 0.578                |
iteration ratio, of Algorithms $M_1$ and $M_2$ with respect to the NFB algorithm are also given in Table 4.

The situation in this case is not quite clear. Namely, comparing the time ratio with respect to the number of iterations ratio given in columns labeled $M_1$/NFB and $M_2$/NFB, we conclude the number of required iterations of $M_1$ and $M_2$ algorithms for problems $czprob$, $share1b$, $ship04l$, and $ship08l$ is smaller than the number of iterations of NFB algorithm but this is not in accordance with required CPU time ratio. This is the consequence of the greatest number of conditions that must be checked in the $M_1$ and $M_2$ algorithms with respect to the NFB algorithm. The difficulties of that type may occur especially when the LP problem has dense matrix. Note that for all others test problems accordance of the number of iterations ratio with respect to the required CPU time ratio does exist.

Figure 2 shows the performance profiles for NFB, $M_1$, $M_2$ regarding the required CPU time.
It is clear from Figure 2 that the probability of being the optimal solver regarding the CPU time required is in favor of the M2 algorithm, which is confirmed by the inequalities $\rho_{M2}(1) > \rho_{M2}(1) > \rho_{NFB}(1)$. On the other hand, further comparison when $\tau$ increases gives contradictory results. Unfortunately, M2 algorithm is not the best solver for all value $\tau$. In the case $1.5 \leq \tau \leq 4.5$ it follows that M1 algorithm is the best option except for $\tau = 2$ when NFB algorithm takes advantage. Further, for $5 \leq \tau \leq 9$ we have $\rho_{M2}(\tau) = \rho_{NFB}(\tau) = \rho_{M1}(\tau)$, which means that the considered solvers are equivalent. Finally, for values $\tau > 9$ Algorithm M2 is the best again.

4. Conclusion

The importance of selecting the starting point is known problem from the linear programming theory. In this paper, we investigated that problem in terms of the number of iterations required to find the initial point and the optimal point as well as in the terms of the required CPU time.

We observed that our Algorithm M2 achieved the lowest total number of iterations for finding the starting point for the Simplex method as well as the minimal total number of iterations for determining the optimal point. On the other hand, Algorithm M1 has the highest probability to achieve the minimal number of iterations to find the initial basic feasible solution. Algorithm M2 has the highest probability of being the optimal solver with respect to the total number of iterations required to solve the problem.

Comparison with respect to the required CPU time gives that the probability of being the optimal solver is in favor of the M2 algorithm for $\tau = 1$ and for $\tau \geq 9.5$. It is clear that CPU time generally depends on the number of iterations, but it is obvious that there are some others important factors. Further research will be in that direction.

Acknowledgments

The authors would like to thank the anonymous referees for useful suggestions. P. S. Stanimirović, N. V. Stojković, and M. D. Petković gratefully acknowledge support from the Research Project 174013 Serbion Ministry of Education and Science.

References

[1] P. S. Stanimirović, N. V. Stojković, and V. V. Kovačević-Vujčić, “Stabilization of Mehrotra’s primal-dual algorithm and its implementation,” European Journal of Operational Research, vol. 165, no. 3, pp. 598–609, 2005.
[2] N. V. Stojković and P. S. Stanimirović, “Initial point in primal-dual interior point method,” Facta Universitatis, Series Mechanical Engineering, vol. 3, pp. 219–222, 2001.
[3] N. V. Stojković and P. S. Stanimirović, “Two direct methods in linear programming,” European Journal of Operational Research, vol. 131, no. 2, pp. 417–439, 2001.
[4] H. W. Corley, J. Rosenberger, W. C. Yeh, and T. K. Sung, “The cosine simplex algorithm,” International Journal of Advanced Manufacturing Technology, vol. 27, no. 9-10, pp. 1047–1050, 2006.
[5] P. Q. Pan, “The most-obtuse-angle row pivot rule for achieving dual feasibility: a computational study,” European Journal of Operational Research, vol. 101, no. 1, pp. 164–176, 1997.
[6] W.-C. Yeh and H. W. Corley, “A simple direct cosine simplex algorithm,” Applied Mathematics and Computation, vol. 214, no. 1, pp. 178–186, 2009.
[7] H. Arsham, “A big-M free solution algorithm for general linear programs,” International Journal of Pure and Applied Mathematics, vol. 32, no. 4, pp. 37–52, 2006.
[8] H. Jian-Feng, “A note on ‘an improved initial basis for the simplex algorithm’,” *Computers & Operations Research*, vol. 34, no. 11, pp. 3397–3401, 2007.
[9] H. V. Junior and M. P. E. Lins, “An improved initial basis for the Simplex algorithm,” *Computers & Operations Research*, vol. 32, no. 8, pp. 1983–1993, 2005.
[10] N. Khan, S. Inayatullah, M. Imtiaz, and F. H. Khan, “New artificial-free phase 1 simplex method,” *International Journal of Basic and Applied Sciences*, vol. 9, no. 10, pp. 97–114, 2009.
[11] H. Nabli, “An overview on the simplex algorithm,” *Applied Mathematics and Computation*, vol. 210, no. 2, pp. 479–489, 2009.
[12] J. K. Strayer, *Linear Programming and Its Applications*, Springer, 1989.
[13] N. V. Stojkovic, P. S. Stanimirovic, and M. D. Petkovic, “Modification and implementation of two-phase simplex method,” *International Journal of Computer Mathematics*, vol. 86, no. 7, pp. 1231–1242, 2009.
[14] E. D. Nering and A. W. Tucker, *Linear Programs and Related Problems*, Academic Press, New York, NY, USA, 1993.
[15] R. G. Bland, “New finite pivoting rules for the simplex method,” *Mathematics of Operations Research*, vol. 2, no. 2, pp. 103–107, 1977.
[16] J. P. Ignizio, *Linear Programming in Single-Multiple-Objective Systems*, Prentice Hall, Englewood Cliffs, NJ, USA, 1982.
[17] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA, 2001.
[18] E. D. Dolan and J. J. Moré, “Benchmarking optimization software with performance profiles,” *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.