STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **16**, August 2026, pp 1727–1745.
Published online in International Academic Press (www.IAPress.org)

Dual support M-method for solving linear programs with non-negative variables

Abdelhak Hebbache[1,2], Mohand Bentobache [1,2,*], Mohand Ouamer Bibi [1]

[1] *University of Bejaia, Faculty of Exact Sciences, LaMOS Research Unit (Modeling and Optimization of Systems), 06000 Bejaia, Algeria*
[2] *Laboratory of Pure and Applied Mathematics, University of Laghouat, 03000 Laghouat, Algeria*

Abstract This paper presents a version of the dual support method specifically designed to handle linear programming (LP) problems with non-negative variables. Since an initial support feasible solution is generally unavailable in advance, we introduce a dual support M-method to solve LP problems without requiring a prior starting point. Additionally, we develop efficient updating formulas for the inverse matrix and the pseudo-feasible solution utilized within the algorithm. To evaluate the performance of the proposed method against the dual simplex and interior-point methods, we implement the proposed algorithm in MATLAB. Computational experiments evaluating CPU time and the number of iterations across both randomly generated test problems and standard NETLIB benchmarks demonstrate the efficiency of the proposed approach, especially on solving dense problems.

Keywords Dual support method, Linear Programming, Big M, Updating Formulas, Numerical Experiments.

AMS 2010 subject classifications 90C32, 65K05.

DOI: 10.19139/soic-2310-5070-3541

1. Introduction

Linear Programming (LP) consists in maximizing a linear function subject to linear constraints. A large number of practical problems arising in applications can be modeled as linear programs, we can cite production management, resource allocation, biology, finance, and economy [8, 20]. In 1947, Dantzig proposed the Primal Simplex Method (PSM) for solving LP problems [7]; after that, several methods were developed, such as the dual simplex method, the primal-dual simplex method, active-set methods, support methods, interior-point methods, exterior-point method, steepest feasible direction method, hybrid direction methods [5, 6, 8, 9, 10, 11, 14, 16, 21, 22, 25, 24, 27].

It is well known that the primal simplex algorithm is initialized by an initial extreme point and moves from one extreme point to a better adjacent one until an optimal vertex is found. In the seventies, Gabasov and Kirillova [11] developed the Primal Support Method (PSupM) which is a generalization of PSM. Indeed, the PSupM improves the objective value by moving from the current support feasible solution to a new one until satisfying an optimality or suboptimality criterion. These solutions combine a support (a set of basic indices) with a feasible point that can lie on the boundary, in the interior, or at an extreme point of the feasible region. After that, Gabasov and Kirillova developed the Dual Support Method (DSupM) and the Adaptive Method (AM) for solving LP problems with bounded variables [11, 12]. Recently, a variant of AM with hybrid direction is extended for solving linear fractional programming problems [17].

The case where certain bounds can take infinite values is handled by introducing artificial finite bounds for these variables, then the bounded version of DSupM or AM is applied to solve the new problem. Finally, an analysis

*Correspondence to: Mohand Bentobache (Email: m.bentobache@lagh-univ.dz). Laboratory of Pure and Applied Mathematics, University of Laghouat, 03000, Laghouat, Algeria.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

of results is done when certain optimal components are equal to their artificial bounds (see page 198 in [12]). However, to our knowledge, there does not exist a version of the dual support M-method for handling problems with non-negative variables as they are presented in the original problem.

In [12], a full artificial basis technique is used to find a feasible starting solution for the adaptive method with finite bounded variables and in [2] the single artificial variable is adapted to initialize PSupM with bounded variables. In this work, being inspired by existing LP initialization procedures [1, 2, 3, 4, 8, 18, 19, 25], we propose a new dual support M-method for solving LP problems with non-negative variables without knowing in advance an initial solution. Next, it is well known that updating the inverse in each iteration of pivoting methods can lead to substantial gain in CPU time. In this work, we propose new updating formulas for efficiently computing the new inverse matrix and the new pseudo-feasible solution used in the dual support method.

Additionally, we conducted a numerical study on a set of randomly generated problems of varying densities and sizes, as well as benchmark LPs from the NETLIB library. This library contains practical LP test problems modeling a wide range of applications, such as resource allocation, nutrition, economics, and data fitting. We compared DSupM against the Dual Simplex Method (DSM) implemented in MATLAB, the Interior-Point Method (IPM) of MATLAB, and the open-source solver GLPK [15, 23].

In Section 2, we present the problem and recall some important results and definitions. In Section 3, we describe the non-negative version of the dual support method. In Section 4, the dual support M-method is introduced. In Sections 5 and 6, the proposed new updating technique is detailed and illustrated by a numerical example. Section 7 is devoted to numerical results that compare DSupM with the dual simplex method and the interior-point method. Finally, in the last section, we conclude the paper and suggest directions for future work.

2. Statement of the problem

The problem of linear programming with non-negative variables is presented in the following standard form:

$$\max z = c^T x, \tag{1}$$

$$Ax = b,\ x \geq 0, \tag{2}$$

where $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, A is a real matrix of order $(m \times n)$ with $rank(A) = m < n$. The symbol $(\ ^T)$ represents the transposition operation.

The dual problem associated with the primal problem (1)-(2) is given by:

$$\min \phi = b^T y, \tag{3}$$

$$A^T y - c - \delta = 0,\ \delta \geq 0,\ y \in \mathbb{R}^m. \tag{4}$$

Let us define the following sets of indices:

$$I = \{1, 2, \ldots, m\},\ J = \{1, 2, \ldots, n\},\ J = J_B \cup J_N,\ J_B \cap J_N = \emptyset,\ |J_B| = m.$$

So we can partition the vectors and the matrix A with respect to the partition (J_B, J_N), as follows:

$$x = \begin{pmatrix} x_B \\ x_N \end{pmatrix},\ c = \begin{pmatrix} c_B \\ c_N \end{pmatrix},\ A = (A_B, A_N),$$

with

$$x_B = x(J_B) = (x_j,\ j \in J_B),\ x_N = x(J_N) = (x_j,\ j \in J_N),\ c_B = c(J_B),\ c_N = c(J_N),$$

$$A_B = (a_j,\ j \in J_B),\ A_N = (a_j,\ j \in J_N),\ a_j = A(I, j) = (a_{ij}, i \in I) \text{ is the } j\text{-th column of } A.$$

- An index set $J_B \subset J$, such that $|J_B| = |I| = m$ is called a support if $\det(A_B) \neq 0$.

- A pair $\{x, J_B\}$ formed with a feasible solution x and a support J_B is called a Support Feasible Solution (SFS) for the primal problem (1)-(2).
- An SFS is said to be non-degenerate if $x_j > 0,\ j \in J_B$.
- Let J_B be a support. We define the vector of multipliers π and the reduced costs vector $E^T = (E_B^T, E_N^T)$, as follows:
$$\pi^T = c_B^T A_B^{-1},\ E_B = 0,\ E_N^T = \pi^T A_N - c_N^T. \tag{5}$$
- The support J_B is said to be a Dual Feasible Support (DFS), if the corresponding reduced costs vector satisfies the following condition:
$$E_N = A_N^T \pi - c_N \geq 0.$$
- An m-vector y is called a dual feasible solution if it satisfies the constraints of the dual problem (3)-(4).
- If y is a dual feasible solution, then the n-vector $\delta = A^T y - c$ is called the feasible co-solution corresponding to y.
- On the other hand, the pair $\{y, J_B\}$ comprising a dual feasible solution and a support J_B is called a support dual feasible solution for the problem (1)-(2), and the pair $\{\delta, J_B\}$ is called the support feasible co-solution.
- A support feasible co-solution $\{\delta, J_B\}$ is said to be non-degenerate if
$$\delta_j > 0,\ \forall j \in J_N = J \setminus J_B.$$
- A dual feasible solution y^* is called optimal if it minimizes the objective function of the dual problem (3)-(4). The vector $\delta^* = A^T y^* - c$ is then said to be an optimal co-solution.
- Finally, let $\{\delta, J_B\}$ be a support feasible co-solution of the dual problem (3)-(4) and consider the n-vector $\kappa = (\kappa_B, \kappa_N)$ defined by:
$$\kappa_j = 0,\ j \in J_N \text{ and } \kappa(J_B) = A_B^{-1} b.$$
Then κ is said to be a pseudo-feasible solution of the primal problem (1)-(2) and the pair $\{\kappa, J_B\}$ is called a support pseudo-feasible solution. Notice that we have $A\kappa = b$, so if in addition $\kappa_B \geq 0$, κ will be a feasible solution for the primal problem.

Remark 1. Let J_B be a DFS and E be its corresponding reduced costs vector. Then the vector (y, δ), with $y^T = \pi^T = c_B^T A_B^{-1}$ and $\delta = E = A^T \pi - c$, is feasible for the dual problem. Indeed, we have
$$\delta_B = E_B = 0 \text{ and } \delta_N = E_N = A_N^T \pi - c_N \geq 0 \ \Rightarrow A^T \pi \geq c.$$

3. The dual support method

3.1. Increment formula of the dual function

Let $\{y, J_B\}$ be a support dual feasible solution, $\{\delta, J_B\}$ be its corresponding support co-solution and κ be a pseudo-feasible solution. Consider another arbitrary dual feasible solution $\overline{y}$ and the associated co-solution $\overline{\delta}$, such that
$$\overline{y} = y + \sigma s,\ \overline{\delta} = A^T \overline{y} - c,\ \overline{\delta} = \delta + \sigma t,\ \sigma \geq 0,\ s \in \mathbb{R}^m,\ t \in \mathbb{R}^n.$$
Let us calculate the increment of the dual objective function: we have
$$\overline{y} = y + \sigma s \Rightarrow \overline{\delta} = A^T \overline{y} - c = A^T y - c + \sigma A^T s \Rightarrow \overline{\delta} = \delta + \sigma A^T s \Rightarrow t^T = s^T A.$$

So, we deduce that

$$\phi(\overline{y}) - \phi(y) = \overline{y}^T b - y^T b = (\overline{y} - y)^T b = \sigma s^T b = \sigma s^T A\kappa = \sigma t^T \kappa = \sigma \sum_{j \in J} t_j \kappa_j.$$

Since the pseudo-feasible solution κ verifies $\kappa_j = 0,\ j \in J_N$, we obtain

$$\phi(\overline{y}) - \phi(y) = \sigma \sum_{j \in J_B} t_j \kappa_j. \tag{6}$$

3.2. Optimality criterion

In order to test the optimality of the support co-solution $\{\delta, J_B\}$, we have the following theorem [11]:

Theorem 3.1
Let $\{\delta, J_B\}$ be a support co-solution for the problem (3)-(4) and κ a pseudo-feasible solution associated to the support J_B. Then the relationships:

$$\begin{cases} \kappa_j = 0, & \text{for } \delta_j > 0; \\ \kappa_j \geq 0, & \text{for } \delta_j = 0; \end{cases} \quad j \in J_B, \tag{7}$$

are sufficient for the optimality of the support co-solution $\{\delta, J_B\}$. They are also necessary in the case of the nondegeneracy of the co-solution $\{\delta, J_B\}$. The pseudo-feasible solution κ corresponding to the optimal co-solution $\{\delta, J_B\}$ is then an optimal solution of the primal problem (1)-(2).

Proof
Sufficiency. Let $\{\delta, J_B\}$ be a co-solution verifying the optimality criterion, and let $\overline{\delta}$ an other arbitrary co-solution. By using the increment formula of dual function (6) and the relationships (7), we obtain

$$\phi(\overline{y}) - \phi(y) = \sigma \sum_{j \in J_B} t_j \kappa_j = \sigma \sum_{j \in J_B, \delta_j = 0.} t_j \kappa_j = \sum_{j \in J_B, \delta_j = 0.} \overline{\delta}_j \kappa_j.$$

Since $\kappa_j \geq 0$ for $\delta_j = 0$, and $\overline{\delta}_j \geq 0$, we will have

$$\phi(\overline{y}) - \phi(y) \geq 0 \Rightarrow \phi(\overline{y}) \geq \phi(y).$$

Therefore, the dual feasible solution y and its corresponding co-solution δ are optimal for the dual problem.

Now let us show that a pseudo-feasible solution κ which satisfies the optimality criterion (7) is an optimal solution of the primal problem (1)-(2). First, it is clear that κ is feasible for the primal problem because $A\kappa = b$ and $\kappa \geq 0$. Moreover, we have

$$z(\kappa) = c^T \kappa = (-\delta^T + y^T A)\kappa = -\delta^T \kappa + y^T b.$$

Since δ and κ verify (7), we get

$$\delta^T \kappa = \sum_{j \in J_B} \delta_j \kappa_j = 0 \Rightarrow z(\kappa) = c^T \kappa = y^T b = \phi(y).$$

According to the theory of duality in linear programming, the vector κ is therefore an optimal solution for the primal problem (1)-(2).
Necessity. Let $\{\delta, J_B\}$ be a non-degenerate optimal co-solution and suppose that the optimality criterion is not verified. Then there exists an index $j_1 \in J_B$, with

$$[\delta_{j_1} > 0 \text{ and } \kappa_{j_1} \neq 0] \text{ or } [\delta_{j_1} = 0 \text{ and } \kappa_{j_1} < 0].$$

Let us assume that $j_1 \in J_B$ verifies $\delta_{j_1} > 0$ and $\kappa_{j_1} \neq 0$.
Thus, we can construct a new feasible dual solution $\overline{y}$ and its corresponding co-solution $\overline{\delta}$, as follows

$$\overline{y} = y + \sigma^0 s,\ \overline{\delta} = \delta + \sigma^0 t,\ \sigma^0 > 0,\ s \in \mathbb{R}^m,\ t \in \mathbb{R}^n,$$

where t is defined by:

$$t_{j_1} = -\text{sign}(\kappa_{j_1});\ t_j = 0,\ j \neq j_1,\ j \in J_B;\ \ s^T = t_B^T A_B^{-1};\ \ t_N^T = s^T A_N.$$

In order to maintain the feasibility of the co-solution $\overline{\delta}$, the step-length σ^0 can be chosen in the interval $]0, \min\{\sigma_{j_1},\ \sigma_{j_0}\}]$, where:

$$\sigma_{j_1} = \begin{cases} \delta_{j_1}, & \text{if } \kappa_{j_1} > 0; \\ \infty, & \text{if } \kappa_{j_1} < 0; \end{cases} \text{ and } \sigma_{j_0} = \min_{j \in J_N} \{\sigma_j\},\ \text{with } \sigma_j = \begin{cases} -\delta_j / t_j, & \text{if} \quad t_j < 0; \\ \infty & \text{if} \quad t_j \geq 0. \end{cases}$$

By using the increment formula of the dual function (6) and the non-degeneracy assumption ($\sigma^0 > 0$), we obtain:

$$\phi(\overline{y}) - \phi(y) = \sigma^0 t^T \kappa = \sigma^0 \sum_{j \in J_B} t_j \kappa_j = -\sigma^0 |\kappa_{j_1}| < 0 \Rightarrow \phi(\overline{y}) < \phi(y).$$

This last inequality contradicts the optimality of the feasible dual solution y.
Similarly, we can show that the case $\delta_{j_1} = 0$ and $\kappa_{j_1} < 0$ leads also to a contradiction. □

3.3. The dual support algorithm

The dual support algorithm starts with a feasible dual solution y and a support J_B. We calculate the corresponding feasible co-solution $\{\delta, J_B\}$ and the pseudo-feasible solution κ, then we check the optimality of this pseudo-feasible solution.
- First we calculate the set of non-optimal basic indices:

$$J_{BNO} = \{j \in J_B :\ [\delta_j > 0 \text{ and } \kappa_j \neq 0] \text{ or } [\delta_j = 0 \text{ and } \kappa_j < 0]\}.$$

- If $J_{BNO} = \emptyset$, then the algorithm stops with κ an optimal solution to the primal problem.
- Otherwise, we will construct another better feasible co-solution $\overline{\delta}$.
An iteration of the algorithm therefore consists in passing from $\{\delta, J_B\}$ to $\left\{\overline{\delta}, \overline{J}_B\right\}$, with $\overline{\delta} = \delta + \sigma^0 t,\ \sigma^0 \geq 0$, where $t \in \mathbb{R}^n$ is called the dual direction and σ^0 the step-length along this direction.
The leaving index j_1 is chosen as follows:

$$|\kappa_{j_1}| = \max_{j \in J_{BNO}} |\kappa_j|.$$

The dual direction t is given by:

$$t_{j_1} = -\text{sign}(\kappa_{j_1});\ t_j = 0,\ j \neq j_1,\ j \in J_B;\ \ t_N^T = t_B^T A_B^{-1} A_N. \tag{8}$$

The step-length σ^0 along the dual direction t is computed so as to maintain the feasibility of the new co-solution $\overline{\delta}$:

$$\overline{\delta} \geq 0 \Rightarrow \delta + \sigma^0 t \geq 0 \Rightarrow \delta_j + \sigma^0 t_j \geq 0,\ j \in J_B \text{ and } \delta_j + \sigma^0 t_j \geq 0,\ j \in J_N.$$

Since $t_{j_1} = -\text{sign}(\kappa_{j_1})$ and $t_j = 0,\ j \in J_B \setminus \{j_1\}$, we obtain

$$\sigma^0 \text{sign}(\kappa_{j_1}) \leq \delta_{j_1};\ -\sigma^0 t_j \leq \delta_j,\ j \in J_N. \tag{9}$$

By calculating the different values that can take the step-length σ^0 in the relationships (9), we will have: $\sigma^0 = \min\{\sigma_{j_1}, \sigma_{j_0}\}$, where

$$\sigma_{j_1} = \begin{cases} \delta_{j_1}, & \text{if } \kappa_{j_1} > 0; \\ \infty, & \text{if } \kappa_{j_1} < 0; \end{cases} \quad \sigma_{j_0} = \min_{j \in J_N} \{\sigma_j\},\ \text{with } \sigma_j = \begin{cases} -\frac{\delta_j}{t_j}, & \text{if} \quad t_j < 0; \\ \infty, & \text{if} \quad t_j \geq 0. \end{cases}$$

- If $\sigma^0 = \infty$, then the dual problem is unbounded and according to the theory of duality, the set of feasible solutions of the primal problem (1)-(2) is empty. Else, we change the support J_B as follows:
- If $\sigma^0 = \sigma_{j_1}$, we put $\overline{J}_B = J_B$.
- If $\sigma^0 = \sigma_{j_0}$, we put $\overline{J}_B = (J_B \setminus \{j_1\}) \cup \{j_0\}$.
Notice that the passage from δ to $\overline{\delta}$ is accompanied by a decrease of the dual objective function equal to $-\sigma^0|\kappa_{j_1}|$. After that, we start a new iteration with the new support co-solution $\left\{\overline{\delta}, \overline{J}_B\right\}$.

Let J_B be an initial support, y be a starting dual feasible solution and $\delta = A^T y - c$ be its corresponding feasible co-solution. The dual support algorithm is summarized in the following steps:

Algorithm 1. (Dual Support Algorithm)

1. Calculate the pseudo-feasible solution κ associated with the co-solution δ, using the relationships:
$$\kappa_j = 0,\ j \in J_N \text{ and } \kappa(J_B) = A_B^{-1} b;$$
2. Test the optimality of the support pseudo-feasible solution $\{\kappa, J_B\}$:
- Determine the set of non-optimal basic indices:
$$J_{BNO} = \{j \in J_B :\ \delta_j > 0 \text{ and } \kappa_j \neq 0\} \cup \{j \in J_B :\ \delta_j = 0 \text{ and } \kappa_j < 0\}.$$
- If $J_{BNO} = \emptyset$, then the algorithm stops with $\{\kappa, J_B\}$ an optimal solution for the primal problem (1)-(2).
- Else, choose the index j_1 such that: $|\kappa_{j_1}| = \max\limits_{j \in J_{BNO}} |\kappa_j|$.
- Calculate the dual direction $t = (t_B, t_N)$ using the following formulas:
$$t_{j_1} = -\text{sign}(\kappa_{j_1});\ t_j = 0,\ j \neq j_1,\ j \in J_B;\ \ t_N^T = t_B^T A_B^{-1} A_N;$$
- Calculate the numbers σ_{j_1} and σ_{j_0}:
$$\sigma_{j_1} = \begin{cases} \delta_{j_1}, & \text{if } \kappa_{j_1} > 0; \\ \infty, & \text{if } \kappa_{j_1} < 0; \end{cases} \quad \sigma_{j_0} = \min_{j \in J_N} \sigma_j, \text{ with } \sigma_j = \begin{cases} -\delta_j / t_j, & \text{if} \quad t_j < 0; \\ \infty, & \text{if} \quad t_j \geq 0; \end{cases}$$
- Calculate the step-length along the dual direction: $\sigma^0 = \min\{\sigma_{j_1}, \sigma_{j_0}\}$;
3. Change of the co-solution δ and the support J_B:
- If $\sigma^0 = \infty$, then the dual problem is unbounded, so the set of feasible solutions of the primal problem (1)-(2) is empty.
- If $\sigma^0 = \sigma_{j_1}$, we put $\overline{J}_B = J_B$;
- If $\sigma^0 = \sigma_{j_0}$, we put $\overline{J}_B = (J_B \setminus \{j_1\}) \cup \{j_0\}$;
- Calculate $\overline{\delta} = \delta + \sigma^0 t$ and $\overline{\phi} = \phi - \sigma^0 |\kappa_{j_1}|$;
4. Set $\delta := \overline{\delta},\ \phi := \overline{\phi},\ J_B := \overline{J}_B$ and go to step 1;

4. The dual support M-method

When an initial support dual feasible solution is not available for the dual support method, we form a new problem called "M-problem" with a new variable and an additional constraint, which is given by:

$$\max\ w(\bar{x}) = \bar{c}^T \bar{x}, \tag{10}$$
$$\text{s.t. } \bar{A}\bar{x} = \bar{b},\ \bar{x} \geq 0, \tag{11}$$

with $\bar{I} = I \cup \{m+1\},\ \bar{J} = J \cup \{n+1\}$ and

$$\bar{A} = \bar{A}(\bar{I}, \bar{J}) = \begin{pmatrix} A & 0_{\mathbb{R}^m} \\ e_{\mathbb{R}^n}^T & 1 \end{pmatrix},\ \bar{b} = \begin{pmatrix} b \\ M \end{pmatrix},\ \bar{x} = \begin{pmatrix} x \\ x_{n+1} \end{pmatrix},\ \bar{c} = \begin{pmatrix} c \\ 0 \end{pmatrix},$$

where $e_{\mathbb{R}^n}$ is an n-vector of ones and M is a positive arbitrary number assumed to be very big. The dual problem associated with the problem (10)-(11) is given by:

$$\min\ \psi(\bar{y}) = \bar{b}^T\bar{y} = b^T y + M\bar{y}_{m+1},\ \text{s.t. } \bar{A}^T\bar{y} \geq \bar{c}, \tag{12}$$

with $\bar{y} = (y, \bar{y}_{m+1}) \in \mathbb{R}^m \times \mathbb{R}$, $\bar{A}^T = \begin{pmatrix} A^T & e_{\mathbb{R}^n} \\ 0^T_{\mathbb{R}^m} & 1 \end{pmatrix}$.

Then, we have the following result:

Proposition 1

Let J_B be a support for the original problem (1)-(2). Then, the pair $\{\bar{y}, \bar{J}_B\}$, with

$$\bar{J}_B = J_B \cup \{n+1\},\ \bar{y} = \begin{pmatrix} y \\ \bar{y}_{m+1} \end{pmatrix} = (y_1, y_2, \ldots, y_m, \bar{y}_{m+1})^T, \tag{13}$$

$$y = e_{\mathbb{R}^m} \text{ and } \bar{y}_{m+1} = \lambda_2 - m\lambda_1, \tag{14}$$

where

$$\lambda_1 = \min\{a_{ij},\ i \in I,\ j \in J\} \text{ and } \lambda_2 = \max\{\max\{c_j,\ j \in J\}, m\lambda_1\}, \tag{15}$$

is a support dual feasible solution for the problem (12).

Proof
We have

$$det\bar{A}_B = det\bar{A}(\bar{I}, \bar{J}_B) = det \begin{pmatrix} A_B & 0_{R^m} \\ e^T_{\mathbb{R}^m} & 1 \end{pmatrix} = 1 \times detA_B \neq 0.$$

Moreover, it is clear that

$$\bar{A}^T_{n+1} = (0^T_{\mathbb{R}^m}, 1),\ \bar{A}^T_{n+1}\bar{y} = \bar{y}_{m+1} = \lambda_2 - m\lambda_1 \geq 0 = c_{n+1}$$

and for all $j = 1, 2, \ldots, n$, we have

$$\bar{A}^T_j = (a_{1j}, a_{2j}, \ldots, a_{mj}, 1),\ \bar{A}^T_j\bar{y} = \sum_{i=1}^{m} a_{ij} + \lambda_2 - m\lambda_1 \geq \sum_{i=1}^{m} \lambda_1 + \lambda_2 - m\lambda_1 = \lambda_2 \geq c_j,$$

which implies that $\bar{A}^T\bar{y} \geq \bar{c}$, i.e., $\bar{y}$ is a feasible solution for the dual problem (12). □

Remark 2. If $m\lambda_1 \geq \max\{c_j, j \in J\}$, then $\lambda_2 = m\lambda_1$ and $\bar{y}_{m+1} = \lambda_2 - m\lambda_1 = 0$. Hence, $y = e_{\mathbb{R}^m}$ is a feasible solution for the dual of the original problem, so it is not necessary to solve the M-problem, we start directly solving the original problem with the initial dual feasible solution $y = e_{\mathbb{R}^m}$. Indeed, for $\bar{y} = (e_{\mathbb{R}^m}, 0)$, we have $\bar{A}^T\bar{y} \geq \bar{c} \Rightarrow A^T e_{\mathbb{R}^m} \geq c$.

Now, we solve the M-problem (10)-(11) with the dual support method (Algorithm 1), starting with the support dual feasible solution $\{\bar{y}, \bar{J}_B\}$ given by (14)-(15). If the dual support method stops with $\sigma^0 = \infty$, then the dual of the M-problem is unbounded. Otherwise, let $\bar{\kappa}^* = (\kappa^*, \bar{\kappa}^*_{n+1})$, $\bar{y}^* = (y^*, \bar{y}^*_{m+1})$ be the obtained optimal solutions for the M-problem (10)-(11) and its dual, respectively. Let $\bar{\delta}^* = (\delta^*, \bar{\delta}^*_{n+1})$ be the corresponding optimal co-solution. Thus, we have the following proposition.

Proposition 2 (i) If $\sigma^0 = +\infty$, then the original problem is infeasible; otherwise

(ii)) If $\bar{\delta}^*_{n+1} = 0$, then κ^* is an optimal solution for the original problem.

(iii) If $\bar{\delta}^*_{n+1} > 0$, then the original problem is unbounded.

Proof
Case (i). Assume that the dual support method stops with $\sigma^0 = +\infty$. Thus, the dual of the M-problem is unbounded because we have found a dual feasible direction t along which the dual function decreases infinitely. So according to the duality theory, the primal M-problem is infeasible. Now let us show that the infeasibility of the primal M-problem implies the infeasibility of the original problem. Indeed, let us assume that the primal M-problem is infeasible and the original primal problem has at least one feasible solution $x = (x_1, x_2, \ldots, x_n)^T$, so we see clearly that

$$\bar{x} = (x_1, x_2, \ldots, x_n, M - \sum_{i=1}^{n} x_i)^T$$

satisfies the constraints of the M-primal problem and its feasible region is not empty. This contradicts the fact that the primal M-problem is infeasible.
Cases (ii) and (iii). Let us assume that the primal M-problem has an optimal solution. Since $c_{n+1} = 0$ and $\bar{A}_{n+1}^T = (0_{\mathbb{R}^m}^T, 1)$, we have

$$\bar{\delta}_{n+1}^* = \bar{A}_{n+1}^T y^* - c_{n+1} = \bar{y}_{m+1}^*.$$

(ii) If $\bar{\delta}_{n+1}^* = 0$, then $\bar{y}_{m+1}^* = 0$. Since $(\bar{\kappa}^*, \bar{y}^*)$ is an optimal primal-dual pair for the M-problem, according to the duality theory, we have

$$w(\bar{\kappa}^*) = \psi(\bar{y}^*) = \bar{b}^T \bar{y}^* = b^T y^* + M\bar{y}_{m+1}^* = b^T y^* = \phi(y^*).$$

In the other hand, since $c_{n+1} = 0$, we get

$$w(\bar{\kappa}^*) = \bar{c}^T \bar{\kappa}^* = c^T \kappa^* = z(\kappa^*) \Rightarrow z(\kappa^*) = \phi(y^*).$$

Since $\bar{\kappa}^*$ and $\bar{y}^*$ are optimal, we have $A^T y^* \geq c$, $A\kappa^* = b$ and $\kappa^* \geq 0$, which implies that κ^* and y^* are feasible for the primal original problem and its dual, respectively. Therefore, according to the duality theory, the equality $c^T \kappa^* = b^T y^*$ implies that the n-vectors κ^* and y^* are optimal solutions for the primal original problem and its dual, respectively.

(iii) If $\bar{\delta}_{n+1}^* > 0$, then $\bar{y}_{m+1}^* > 0$. Hence, the dual of the original problem is infeasible. Indeed, assume that the dual of the original problem has a feasible solution $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m)^T$, so $\tilde{y} = (\hat{y}, 0)$ is feasible for the dual problem (12). Since M is a sufficiently large positive number and $\bar{y}_{m+1}^* > 0$, we have

$$\psi(\tilde{y}) = b^T \hat{y} < b^T y^* + M\bar{y}_{m+1}^* = \psi(\bar{y}^*),$$

contradicting the optimality of $\bar{y}^*$ for the dual of the M-problem. Therefore, the dual of the original problem is infeasible and according to the duality theory, the original problem is unbounded.

□

5. Efficient updating techniques

It is well known that updating the inverse in each iteration of pivoting methods can lead to substantial gain in CPU time. In this section, we propose an updating procedure for efficiently computing the new inverse matrix and the new pseudo-feasible solution in terms of the old ones. Assume without loss of generality that

$$J_B = \{1, 2, \ldots, m\} \text{ and } J_N = \{m+1, m+2, \ldots, n\}.$$

Let $A_B = A(I, J_B)$ the current basis matrix and $\bar{A}_B$ be the new basis matrix obtained by replacing the s-th column of A_B by the r-th column of A_N.

5.1. Updating the inverse matrix

Let

$$X = (X_i,\ i \in I) = (x_{ji},\ j \in J_B,\ i \in I) = A_B^{-1} = A_B^{-1}(J_B, I),$$
$$\overline{X} = (\overline{X}_i,\ i \in I) = \bar{A}_B^{-1},\ \alpha = (\alpha_j,\ j \in J_B) = A_B^{-1}a_r,\ \text{with } \alpha_s = A_B^{-1}(s, I)a_r \neq 0,$$

where X_i and $\overline{X}_i$ denote here the i-th column of X and the i-th column of $\overline{X}$, respectively. We know that [1, 26]:

$$\overline{X} = \bar{A}_B^{-1} = EA_B^{-1} = EX,$$

where E represents the identity matrix of order m with the s-th column replaced by the vector η, computed as follows:

$$\text{for } j \in J_B,\ \eta_j = \begin{cases} \frac{1}{\alpha_s}, & \text{if } j = s; \\ -\frac{\alpha_j}{\alpha_s}, & \text{if } j \neq s. \end{cases}$$

Let us define the vector $\bar{\eta} = (\bar{\eta}_j,\ j \in J_B) = \eta - e_s$, i.e.,

$$\text{for } j \in J_B,\ \bar{\eta}_j = \begin{cases} \eta_j - 1, & \text{if } j = s; \\ \eta_j, & \text{if } j \neq s, \end{cases}$$

where e_s represents the s-th unitary vector of $\mathbb{R}^m$.

In the following Lemma, we will show that the i-th column of the new inverse is equal to the sum of the i-th column of the old one and the vector $\bar{\eta}$ multiplied by x_{si}, where x_{si} represents the i-th component of the the s-th row of the current inverse matrix. Therefore if $x_{si} = 0$, then the i-th column of the new inverse $\overline{X}$ is equal to the i-th column of the old inverse matrix.

Lemma 5.1
The columns of the new inverse matrix can be updated as follows:

$$\overline{X}_i = X_i + x_{si}\bar{\eta},\ \forall i \in I. \tag{16}$$

Proof
Consider the $(m \times m)$-matrix L which is equal to the zero-matrix of order m with the s-th column replaced by the vector $\bar{\eta}$. So, we can write:

$$E = \begin{pmatrix} 1 & 0 & \dots & \eta_1 & \dots & 0 \\ 0 & 1 & \dots & \eta_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \eta_s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \eta_m & \dots & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \eta_1 & \dots & 0 \\ 0 & 1 & \dots & \eta_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & (\eta_s - 1) + 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \eta_m & \dots & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{pmatrix} + \begin{pmatrix} 0 & \dots & \bar{\eta}_1 & \dots & 0 \\ 0 & \dots & \bar{\eta}_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \bar{\eta}_s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \bar{\eta}_m & \dots & 0 \end{pmatrix} = I_m + L.$$

Thus we have

$$\overline{X} = EX = (I_m + L)X = X + LX.$$

Hence,

$$\begin{aligned}
\overline{X} &= \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{s1} & x_{s2} & \dots & x_{sm} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mm} \end{pmatrix} + \begin{pmatrix} 0 & \dots & \bar{\eta}_1 & \dots & 0 \\ 0 & \dots & \bar{\eta}_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \bar{\eta}_s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \bar{\eta}_m & \dots & 0 \end{pmatrix} \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{s1} & x_{s2} & \dots & x_{sm} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mm} \end{pmatrix} \\
&= \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ x_{s1} & x_{s2} & \dots & x_{sm} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mm} \end{pmatrix} + \begin{pmatrix} \bar{\eta}_1 x_{s1} & \bar{\eta}_1 x_{s2} & \dots & \bar{\eta}_1 x_{sm} \\ \bar{\eta}_2 x_{s1} & \bar{\eta}_2 x_{s2} & \dots & \bar{\eta}_2 x_{sm} \\ \vdots & \vdots & \vdots & \vdots \\ \bar{\eta}_s x_{s1} & \bar{\eta}_s x_{s2} & \dots & \bar{\eta}_s x_{sm} \\ \vdots & \vdots & \vdots & \vdots \\ \bar{\eta}_m x_{s1} & \bar{\eta}_m x_{s2} & \dots & \bar{\eta}_m x_{sm} \end{pmatrix} \\
&= \begin{pmatrix} x_{11} + \bar{\eta}_1 x_{s1} & x_{12} + \bar{\eta}_1 x_{s2} & \dots & x_{1m} + \bar{\eta}_1 x_{sm} \\ x_{21} + \bar{\eta}_2 x_{s1} & x_{22} + \bar{\eta}_2 x_{s2} & \dots & x_{2m} + \bar{\eta}_2 x_{sm} \\ \vdots & \vdots & \vdots & \vdots \\ x_{s1} + \bar{\eta}_s x_{s1} & x_{s2} + \bar{\eta}_s x_{s2} & \dots & x_{sm} + \bar{\eta}_s x_{sm} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} + \bar{\eta}_m x_{s1} & x_{m2} + \bar{\eta}_m x_{s2} & \dots & x_{mm} + \bar{\eta}_m x_{sm} \end{pmatrix}.
\end{aligned}$$

Therefore,

$$\overline{X}_1 = X_1 + x_{s1}\bar{\eta}, \dots, \overline{X}_m = X_m + x_{sm}\bar{\eta}, \text{i.e., } \overline{X}_i = X_i + x_{si}\bar{\eta},\ i \in I.$$

□

5.2. Updating the pseudo-feasible solution

Instead of calculating $\bar{\kappa}_B$ in the dual support method with the formula $\bar{\kappa}_B = \bar{A}_B^{-1} b$, which needs a multiplication of a matrix by a vector, we write the new pseudo-feasible solution in terms of the old one by using an addition of only two vectors. Thus, we have the following Lemma:

Lemma 5.2
The new pseudo-feasible solution can be updated as follows:

$$\bar{\kappa}_B = \kappa_B + \kappa_s \bar{\eta}. \tag{17}$$

Proof
We have

$$\bar{\kappa}_B = \bar{A}_B^{-1} b = E A_B^{-1} b = (I_m + L) A_B^{-1} b = (A_B^{-1} + L A_B^{-1}) b = A_B^{-1} b + L A_B^{-1} b = \kappa_B + L\kappa_B.$$

So,

$$\begin{aligned}
\bar{\kappa}_B &= \kappa_B + L\kappa_B \\
&= \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_s \\ \vdots \\ \kappa_m \end{pmatrix} + \begin{pmatrix} 0 & \dots & \bar{\eta}_1 & \dots & 0 \\ 0 & \dots & \bar{\eta}_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \bar{\eta}_s & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \bar{\eta}_m & \dots & 0 \end{pmatrix} \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_s \\ \vdots \\ \kappa_m \end{pmatrix} \\
&= \begin{pmatrix} \kappa_1 \\ \kappa_2 \\ \vdots \\ \kappa_s \\ \vdots \\ \kappa_m \end{pmatrix} + \begin{pmatrix} \bar{\eta}_1\kappa_s \\ \bar{\eta}_2\kappa_s \\ \vdots \\ \bar{\eta}_s\kappa_s \\ \vdots \\ \bar{\eta}_m\kappa_s \end{pmatrix} \\
&= \kappa_B + \kappa_s\bar{\eta}.
\end{aligned}$$

□

Remark 3. The inverse updating formula (16) needs to compute only m_{nz} sums of two vectors, where m_{nz} represents the number of nonzero elements of the s-th row of A_B^{-1}. This formula can save much CPU time especially when m_{nz} is small. Furthermore, the updating formula (17) needs only the addition of two vectors, instead of multiplying the new inverse by the right-hand side.

6. Numerical example

Consider the following linear programming problem with non-negative variables:

$$\begin{gathered}
\max\ z = 3x_1 + 2x_2, \\
\text{s.t. } x_1 + x_2 + x_3 = 15, \\
2x_1 + 5x_2 + x_4 = 50, \\
x_j \geq 0,\ j = 1, 2, 3, 4.
\end{gathered}$$

We have

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 5 & 0 & 1 \end{pmatrix},\ b = (15, 50)^T,\ c = (3, 2, 0, 0)^T.$$

The M-problem corresponding to this problem is given by:

$$\begin{gathered}
\max w = 3x_1 + 2x_2, \\
\text{s.t. } x_1 + x_2 + x_3 = 15, \\
2x_1 + 5x_2 + x_4 = 50, \\
x_1 + x_2 + x_3 + x_4 + x_5 = M, \\
x_j \geq 0,\ j = 1, 2, \dots, 5,
\end{gathered} \tag{18}$$

where M is an arbitrary positive number assumed to be very large. We have

$$\lambda_1 = \min\{a_{ij}, i = 1, 2, j = 1, 2, 3, 4\} = 0,$$

$$\max\{c_j, j = 1, 2, 3, 4\} = c_1 = 3,\ \lambda_2 = \max\{c_1, m\lambda_1\} = 3.$$

So $\bar{y}_3 = \lambda_2 - m\lambda_1 = 3$ and the starting dual feasible solution is $\bar{y} = (1,1,3)^T$.
First iteration:
Let us work with the initial support for the M-problem (18): $J_B = \{3,4,5\}$ and $J_N = \{1,2\}$. So we have

$$c = (3,2,0,0,0)^T,\ A = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 5 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix},\ A_B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix},\ A_N = \begin{pmatrix} 1 & 1 \\ 2 & 5 \\ 1 & 1 \end{pmatrix}.$$

The value of the dual objective function is $\psi = 65 + 3M$.
The feasible co-solution is: $\delta = A^T\bar{y} - c = (3,7,4,4,3)^T$.
The pseudo-solution κ associated with the co-solution δ is:

$$\kappa = \begin{pmatrix} \kappa_N \\ \kappa_B \end{pmatrix},\ \kappa_N = 0,\ \kappa_B = A_B^{-1}b = (15,50,M-65)^T,\ \text{with}$$

$$A_B^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix} = X = (X_1, X_2, X_3).$$

We test the optimality of the pseudo-solution κ:
The set of non-optimal support indices is:

$$J_{BNO} = \{j \in J_B :\ [\delta_j > 0 \text{ and } \kappa_j \neq 0] \text{ or } [\delta_j = 0 \text{ and } \kappa_j < 0]\} = \{3,4,5\} \neq \emptyset.$$

So the pseudo-solution κ is not optimal. We have

$$|\kappa_{j_1}| = \max_{j \in J_{BNO}} \{|\kappa_j|\} = |\kappa_5| \Rightarrow j_1 = 5.$$

The dual direction t is:

$$t_B = (0,0,-1)^T,\ t_N = t_B^T A_B^{-1} A_N = (2,5)^T,\ t = (2,5,0,0,-1)^T.$$

The dual step-length σ^0 is computed as follows: $\sigma^0 = \min\{\sigma_{j_1}, \sigma_{j_0}\}$, where

$$\sigma_{j_1} = \sigma_5 = 3,\ \sigma_{j_0} = \min\{\sigma_1, \sigma_2\} = \infty \Rightarrow \sigma^0 = \sigma_{j_1} = 3.$$

The new co-solution $\bar{\delta}$ and the new support $\bar{J}_B$ are:

$$\bar{J}_B = J_B,\ \bar{\delta} = \delta + \sigma^0 t = (9,22,4,4,0)^T.$$

The new value of the dual function is $\bar{\psi} = \psi - \sigma^0|\kappa_{j_1}| = 260$.
In the second iteration, we find:

$$J_B = \{3,4,5\},\ J_N = \{1,2\},\ \kappa_B = (15,50,M-65)^T,\ \delta = (9,22,4,4,0)^T;$$

$$J_{BNO} = \{3,4\},\ j_1 = 4,\ t = (-2,-5,0,-1,0)^T,\ \sigma^0 = \sigma_{j_1} = 4;$$

$$\bar{J}_B = J_B,\ \bar{\delta} = (1,2,4,0,0)^T,\ \bar{\psi} = 60.$$

In the third iteration, we find:

$$J_B = \{3,4,5\},\ J_N = \{1,2\},\ \kappa_B = (15,50,M-65)^T,\ \delta = (1,2,4,0,0)^T;$$

$$J_{BNO} = \{3\},\ j_1 = 3,\ t = (-1,-1,-1,0,0)^T,\ \sigma^0 = \sigma_{j_0} = 1 \Rightarrow j_0 = 1;$$

$$\bar{J}_B = \{1,4,5\},\ \bar{\delta} = (0,1,3,0,0)^T,\ \bar{\psi} = 45.$$

Note that in this iteration $\sigma^0 = \sigma_{j_0}$, so the change of support must be performed and the inverse of the matrix is made as follows: we have

$$A_N = (a_1, a_2) = \begin{pmatrix} 1 & 1 \\ 2 & 5 \\ 1 & 1 \end{pmatrix},\ A_B = (a_3, a_4, a_5) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix},$$

$$A_B^{-1} = X = (X_1, X_2, X_3) = \begin{pmatrix} x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}.$$

The entering index is $j_0 = 1$ and the leaving index is $s = j_1 = 3$. We want to derive the inverse of the new basis matrix $\overline{X} = \bar{A}_B^{-1} = \bar{A}_B^{-1}(\bar{J}_B, I)$, where

$$\bar{J}_B = \{1, 4, 5\},\ \bar{A}_B = (a_1, a_4, a_5) = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

Following the results obtained in Section 5, we have

$$\alpha = (\alpha_3, \alpha_4, \alpha_5)^T = A_B^{-1} a_1 = (1, 2, -2)^T,\ \alpha_s = \alpha_3 = 1,$$

$$\eta = (\eta_3, \eta_4, \eta_5) = (\frac{1}{\alpha_3}, -\frac{\alpha_4}{\alpha_3}, -\frac{\alpha_5}{\alpha_3})^T = (1, -2, 2)^T,\ \bar{\eta} = (\eta_3 - 1, \eta_4, \eta_5) = (0, -2, 2)^T.$$

Therefore, following the updating formula (16), the inverse of the new basis matrix is computed as follows:

$$x_{31} = 1 \neq 0 \Rightarrow \overline{X}_1 = X_1 + x_{31}\bar{\eta} = (1, 0, -1)^T + 1 \times (0, -2, 2)^T = (1, -2, 1)^T,$$

$$x_{32} = 0 \Rightarrow \overline{X}_2 = X_2,\ x_{33} = 0 \Rightarrow \overline{X}_3 = X_3.$$

Therefore,

$$\bar{A}_B^{-1} = \overline{X} = (\overline{X}_1, \overline{X}_2, \overline{X}_3) = (X_1 + x_{31}\bar{\eta}, X_2, X_3) = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix}.$$

Finally, we deduce the vector $\bar{\kappa}$ using the updating Formula (17):

$$\bar{\kappa}_B = (\kappa_3, \kappa_4, \kappa_5)^T = \kappa_B + \kappa_3\bar{\eta} = (15, 50, M - 65)^T + 15 \times (0, -2, 2)^T = (15, 20, M - 35)^T.$$

In the fourth iteration, we have:

$$J_B = \{1, 4, 5\},\ J_N = \{2, 3\},\ \delta = (0, 1, 3, 0, 0)^T,\ \kappa_B = (15, 20, M - 35)^T.$$

The set of non-optimal basic indices is: $J_{BNO} = \emptyset$. Hence, the pseudo-feasible solution κ is optimal for the M-problem. Since $\bar{\delta}^*_{n+1} = \bar{\delta}^*_5 = 0$, following Proposition 5.1, the solution $\kappa^* = (15, 0, 0, 20)^T$ is optimal for the original problem, with $z^* = 45$.

7. Numerical experiments

We have developed an implementation of the Dual Support M-method (DSupM) under the MATLAB programming language version R2018a. We compared DSupM with the Dual Simplex Method implemented in MATLAB (DSM_LINPROG), the Dual Simplex Method of GLPK (DSM_GLPK) [15, 23] and the Interior-Point Method implemented in MATLAB (IPM_LINPROG). All the solvers DSM_GLPK, DSM_LINPROG and IPM_LINPROG are used with their default parameters.

Table 1. Numerical results for test problems of size $m \times m$ and density 10%

$d = 10, m \times m$	DSupM		DSM_GLPK		DSM_LINPROG		IPM_LINPROG	
m	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
200	0.08	603.7	0.02	397.1	0.02	189.8	0.04	13.7
400	0.66	1905.0	0.08	906.5	0.04	520.2	0.26	15.6
600	1.80	3584.8	0.23	1461.6	0.11	1237.5	0.88	17.0
800	6.05	5880.5	0.50	2006.4	0.82	1959.2	1.97	18.5
1000	18.47	8317.7	1.07	2696.1	1.25	2250.6	4.01	19.4

Table 2. Numerical results for test problems of size $m \times m$ and density 50%

$d = 50, m \times m$	DSupM		DSM_GLPK		DSM_LINPROG		IPM_LINPROG	
m	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
200	0.02	360.4	0.03	290.9	0.02	109.9	0.07	15.8
400	0.10	778.9	0.14	656.7	0.05	215.8	0.50	18.1
600	0.56	1397.9	0.37	942.9	0.10	315.5	1.82	18.9
800	1.55	1952.1	0.68	1305.2	0.18	450.5	4.13	20.4
1000	4.53	2847.6	1.42	1652.8	0.29	519.3	8.40	20.8

The different algorithms are executed on a PC with an Intel Core i7-4790 processor CPU 3.60 GHz and 8GB of RAM working under Windows 10. In our implementation of DSupM, we have used the updating formulas (16), (17) to compute the new inverse matrix and the new pseudo-feasible solution, respectively. In order to maintain a certain numerical stability, we reinvert the basis from scratch after every 50 iterations. In the numerical study, two classes of LP test problems are considered: randomly generated test problems of different densities and sizes, and some LPs taken from the NETLIB library [13] with characteristics shown in Table 7.

The randomly generated linear programs that have been solved are of the following form:

$$\{\max z = c^T x,\ Ax \leq b,\ x \geq 0\},$$

where $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and A is a matrix of order $(m \times n)$, with $m \leq n$. The generated LP problems have a constraints matrix A of density 10%, 50% and 100%. Also, two sizes are considered: $m \times m$ and $m \times 2m$. Each class of LPs with a given dimension and a given size contains 10 different instances. Similarly to [25], the coefficients of the constraints matrix A and those of the right-hand-side vector b and the costs vector c are randomly generated in the intervals $[50, 400]$, [10,100] and [-300,700], respectively. Totally we have generated 150 LP test problems of size $m \times m$ and 150 LP test problems of size $m \times 2m$. In DSupM, the initial support is set to $J_B = \{n+1, n+2, \ldots, n+m\}$ for solving the generated test problems and it is computed using the QR factorization of the augmented matrix formed with the equalities and the inequalities. Moreover, we have set in DSupM, the parameter M to 10^5 for solving the generated test problems and it is set to 10^8 for solving the NETLIB test problems.

The numerical results are reported in Tables 1 to 6 for the generated test problems and in Tables 7-8 for the NETLIB test problems, where CPU and ITER represent respectively the mean CPU time in seconds and the average number of iterations of the ten generated problems for each size; CPU_0 is the CPU time necessary to find an initial basis in DSupM (for NETLIB test problems only); E in Tables 7-8 (error) represents the absolute difference between the known optimal value of the NETLIB test problem and the optimal value found by the solver; n, meq, $mineq$ and z^* in Table 7 represent the number of variables, the number of equality constraints, the number of inequality constraints and the optimal value of the NETLIB test problem, respectively. Finally, we plot the CPU time for the different algorithms in Figures 1, 2 and 3.

From the different tables and graphs, we can clearly remark that for test problems of density 10%, DSM_GLPK, DSM_LINPROG and IPM_LINPROG outperforms largely DSupM. For LPs of density 50%, DSM_GLPK, and DSM_LINPROG outperforms DSupM and DSupM is more efficient than IPM_LINPROG. However, for test

Table 3. Numerical results for test problems of size $m \times m$ and density 100%

$d = 100, m \times m$	DSupM		DSM_GLPK		DSM_LINPROG		IPM_LINPROG	
m	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
200	0.01	240.2	0.03	195.7	0.02	48.7	0.13	15.7
400	0.06	507.7	0.15	394.0	0.07	80.6	1.03	18.3
600	0.12	743.8	0.41	578.2	0.15	124.5	2.44	19.0
800	0.19	1012.7	1.03	775.1	0.29	130.4	8.31	21.1
1000	0.66	1311.3	1.42	975.3	0.51	176.4	13.88	21.6

Table 4. Numerical results for test problems of size $m \times 2m$ and density 10%

$d = 10, m \times 2m$	DSupM		DSM_GLPK		DSM_LINPROG		IPM_LINPROG	
m	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
200	0.06	1048.6	0.05	746.5	0.02	348.6	0.05	14.4
400	0.88	3350.8	0.25	1624.4	0.09	1501.9	0.32	17.1
600	5.79	6754.4	0.86	2802.1	0.91	2147.7	1.01	19.1
800	12.35	9945.1	1.53	4067.3	4.67	3235.1	2.32	20.5
1000	27.19	10000	3.39	5476.5	5.91	3991.6	4.77	20.7

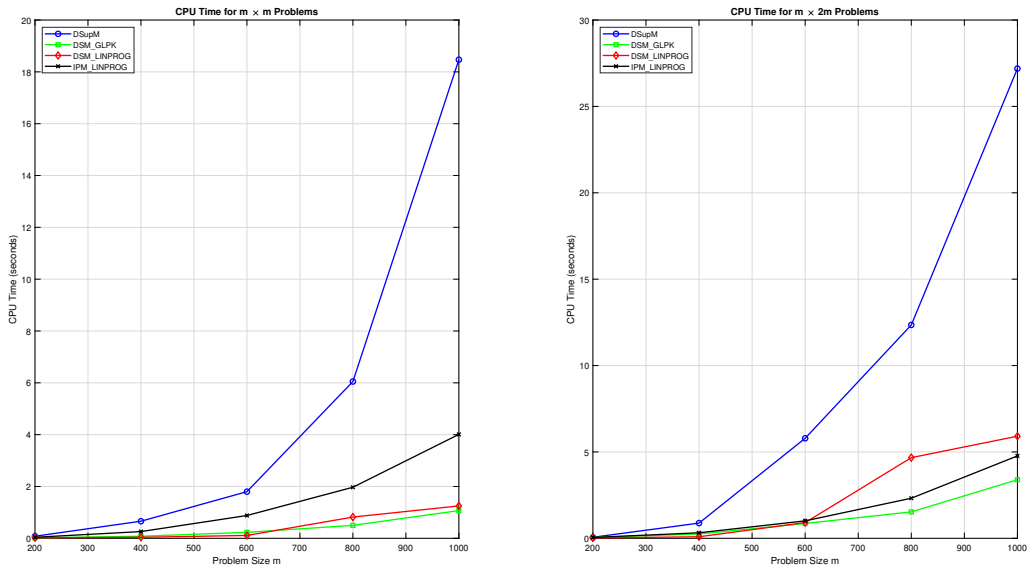

Figure 1. The CPU time of the different algorithms for density 10%

problems of density 100%, we note that the dual support M-method is faster than DSM_GLPK and IPM_LINPROG and its is competitive with DSM_LINPROG.

Note also that the dual simplex and IPM algorithms implemented in GLPK and MATLAB outperform largely our implementation of the dual support M-method for solving the considered NETLIB test problems. The efficiency of DSM on solving sparse LP problems, such as those of the NETLIB library and the generated problems of small densities comes from the fact that the considered implementations of the dual simplex method are designed especially to handle efficiently sparse LPs. Indeed, a variety of efficient computational algorithmic techniques (scaling and presolving techniques, finding advanced initial basis, updating the basis inverse to maintain sparsity,

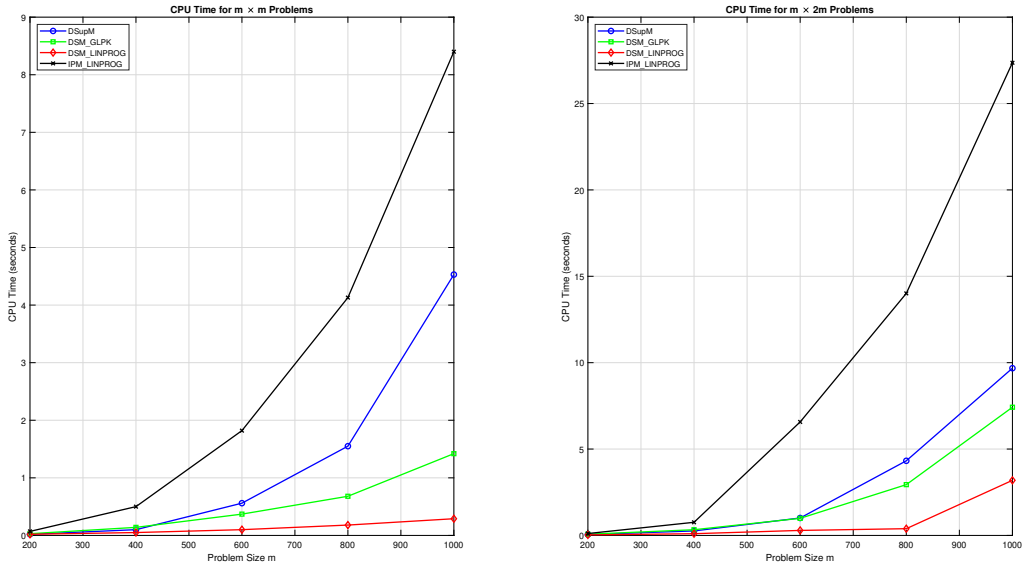

Figure 2. The CPU time of the different algorithms for density 50%

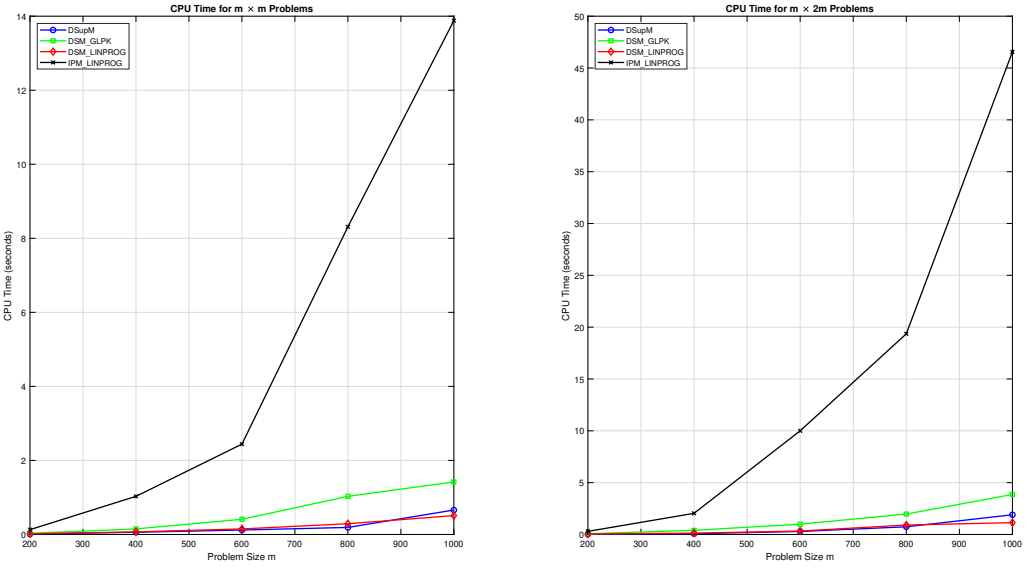

Figure 3. The CPU time of the different algorithms for density 100%

efficient selection of entering variable, etc) are included on those professional solvers to solve large scale and bad-conditioned sparse problems.

Finally, we notice that the value of 10^5 for the M parameter allowed us to find the same optimal values as those of the other solvers, however we have increased M to 10^8 in order to increase the number of the solved NETLIB

Table 5. Numerical results for test problems of size $m \times 2m$ and density 50%

$d = 50, m \times 2m$	DSupM		DSM_GLPK		DSM_LINPROG		IPM_LINPROG	
m	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
200	0.05	479.1	0.07	546.9	0.03	169.4	0.11	17.3
400	0.26	1014.4	0.33	1101.1	0.10	336.0	0.76	20.2
600	1.01	2033.3	0.99	1944.7	0.29	507.7	6.57	22.9
800	4.32	3100.9	2.94	2647.2	0.39	838.8	14.01	23.7
1000	9.68	4268.0	7.42	3865.4	3.19	1380.2	27.36	24.4

Table 6. Numerical results for test problems of size $m \times 2m$ and density 100%

$d = 100, m \times 2m$	DSupM		DSM_GLPK		DSM_LINPROG		IPM_LINPROG	
m	CPU	ITER	CPU	ITER	CPU	ITER	CPU	ITER
200	0.02	266.3	0.07	371.0	0.04	66.2	0.30	18.0
400	0.07	525.6	0.40	728.7	0.13	115.6	2.04	21.0
600	0.29	897.1	0.99	1096.9	0.33	171.6	10.00	22.9
800	0.74	1201.4	1.97	1647.3	0.91	199.1	19.36	24.4
1000	1.90	1548.9	3.86	2394.9	1.14	223.0	46.56	25.3

Table 7. Numerical results for some NETLIB test problems

						DSupM			
No.	Name	n	meq	$mineq$	z^*	E	CPU_0	CPU	ITER
1	adlittle	97	15	41	225494.963160	3.90E-04	0.0007	0.1928	218
2	agg	163	36	452	-35991767.286580	7.75E-05	0.0013	3.6228	692
3	agg2	302	60	456	-20239252.355270	6.59E-04	0.0038	9.1294	1152
4	agg3	302	60	456	10312115.935100	2.40E-05	0.0014	11.6093	1069
5	afiro	32	8	19	-464.753143	4.01E-07	0.0006	0.0128	33
6	bandm	472	305	0	-158.628018	2.33E-04	0.0041	5.1424	1401
7	blend	83	43	31	-30.812150	1.95E-04	0.0033	0.1036	175
8	beaconfd	262	140	33	33592.485807	1.69E-04	0.0020	0.1919	179
9	e226	282	33	190	-18.751929	8.36E-04	0.0004	2.7910	1254
10	sc50a	48	20	30	-64.575077	3.28E-07	0.0006	0.0429	78
11	sc50b	48	20	30	-70.000000	1.52E-07	0.0006	0.0519	70
12	sc105	103	45	60	-52.202061	7.11E-07	0.0002	0.3663	369
13	sc205	203	91	114	-52.202061	1.90E-06	0.0002	2.2527	818
14	scagr7	140	84	45	-2331389.824297	1.95E-05	0.0002	0.2634	282
15	scagr25	500	300	171	-14753433.060750	4.83E-04	0.0014	15.2857	1359
16	share2b	79	13	83	-415.732240	7.98E-04	0.0002	0.5163	709
17	stocfor1	111	63	54	-41131.976219	7.90E-05	0.0002	0.1360	211

test problems, because a small value of M can impact the obtained optimal value and a very big value of M can lead to a numerical instability in the resolution process.

8. Conclusion

In this work, we have described a new version of the dual support method for solving LP problems with non-negative variables. When an initial feasible solution is not known in advance, we have proposed the dual support M-method for solving the original problem. In order to test the performance of the proposed algorithm, we

Table 8. Numerical results for some NETLIB test problems, continued

	DSM_GLPK			DSM_LINPROG			IPM_LINPROG		
No	E	CPU	ITER	E	CPU	ITER	E	CPU	ITER
1	2.38E-06	0.0065	144	2.38E-06	0.0167	75	2.38E-06	0.0086	11
2	3.52E-06	0.0083	180	3.49E-06	0.0129	108	3.49E-06	0.0203	20
3	7.07E-04	0.0187	464	7.07E-04	0.0146	224	7.07E-04	0.0502	25
4	1.08E-05	0.0194	477	1.08E-05	0.0133	192	1.08E-05	0.0490	24
5	5.68E-14	0.0059	18	0.00E+00	0.0623	10	6.54E-07	0.0175	7
6	1.21E-10	0.0176	345	1.21E-10	0.0401	346	7.48E-08	0.0153	15
7	8.28E-12	0.0075	74	7.88E-12	0.0142	68	6.47E-12	0.0087	8
8	5.00E-08	0.0083	104	5.00E-08	0.0119	45	4.93E-08	0.0076	13
9	1.04E-09	0.0147	445	1.04E-09	0.0151	280	6.03E-06	0.0103	13
10	9.62E-08	0.0056	34	9.62E-08	0.0302	39	9.45E-08	0.0084	8
11	1.42E-14	0.0076	31	0.00E+00	0.0751	33	7.67E-13	0.0162	8
12	2.77E-12	0.0044	72	2.85E-12	0.0119	89	3.57E-08	0.0072	10
13	2.77E-12	0.0064	127	2.74E-12	0.0135	145	9.25E-12	0.0080	13
14	3.40E-05	0.0050	124	3.40E-05	0.0109	132	3.35E-05	0.0059	12
15	1.85E-05	0.0172	444	1.85E-05	0.0185	483	5.23E-07	0.0081	16
16	5.13E-07	0.0055	95	5.13E-07	0.0122	83	5.13E-07	0.0065	11
17	6.40E-09	0.0042	64	6.41E-09	0.0105	40	3.46E-09	0.0050	10

have developed an implementation with the MATLAB programming language. In this implementation, efficient techniques for updating the inverse matrix and the pseudo-feasible solution are used. The numerical comparison of the proposed implementation with the dual simplex method of MATLAB and GLPK as well as the interior-point method of MATLAB has shown the efficiency of our method on solving randomly generated dense problems. In the future, we intend to integrate in our implementation advanced computational techniques such as scaling, presolving, implementing an adaptive choice for M to ensure numerical stability in order to solve more difficult problems. Moreover, we will adapt the proposed support M-method for solving convex quadratic programming problems with non-negative variables.

Acknowledgments The authors are grateful to the editor and reviewers for useful suggestions which improved the contents of this paper.

REFERENCES

1. M. S. Bazaraa, and J. J. Jarvis, *Linear programming and network flows*, John Wiley & Sons, New York, 1977.
2. M. Bentobache, and M. O. Bibi, *A two-phase support method for solving linear programs: Numerical experiments*, Mathematical Problems in Engineering, 28 pages, Article ID 482193, doi:10.1155/2012/482193, 2012.
3. M. Bentobache, *On mathematical methods of linear and quadratic probgramming*, PhD Dissertation, University of Bejaia, Bejaia, Algeria, 2013 (in French).
4. M. Bentobache, and M. O. Bibi, *Numerical methods of linear and quadratic programming*, French Academic Press, Germany, 2016 (in French).
5. M. O. Bibi, and M. Bentobache, *The adaptive method with hybrid direction for solving linear programming problems with bounded variables*, Proceedings of Colloque sur l'Optimisation et les Systèmes d'Information, COSI'2011, pp. 80–91, University of Guelma, Algeria, 24-28 April 2011.
6. M. O. Bibi, and M. Bentobache, *A hybrid direction algorithm for solving linear programs*, International Journal of Computer Mathematics, vol. 92, no. 2, pp. 200–216, 2015.
7. G. B. Dantzig, *Maximization of a linear function of variables subject to linear inequalities*, in T. C. Koopmans, Ed., Activity Analysis of Production and Allocation, Wiley & Chapman-Hall, New York, pp. 339–347, 1951.
8. G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, N.J., 1963.
9. K. Djeloud, M. Bentobache, and M. O. Bibi, *A new method with hybrid direction for linear programming*, Concurrency and Computation, Practice and Experience, vol. 33, no. 1, doi: 10.1002/cpe.5836, 2021.
10. I. I. Dikin, *Iterative solution of the problems of linear and quadratic programming*, Soviet Mathematics Doklady, vol. 8, pp. 674–675, 1967.

11. R. Gabasov, and F. M. Kirillova, *Methods of linear programming, Vol. 1, 2 and 3*, Edition of the Minsk University, 1977, 1978, 1980 (in Russian).
12. R. Gabasov, F. M. Kirillova, and S. V. Prischepova, *Optimal Feedback Control*, Springer-Verlag, London, 1995.
13. M. Gay, *Electronic mail distribution of linear programming test problems*, Mathematical Programming Society COAL, Bulletin no. 13, pp. 10–12, 1985. Data available at http://www.netlib.org/lp/data.
14. P. E. Gill, W. Murray, and M. H. Wright, *Numerical linear algebra and optimization*, Addison-Wesley Publishing Company, Redwood City, CA, 1991.
15. N. Giorgetti, *GLPKMEX: A MATLAB MEX interface for the GLPK library*, https://glpkmex.sourceforge.net/, 2007.
16. R. Guerbane, and M. O. Bibi, *Primal-dual method for a linear program with hybrid direction*, International Journal of Mathematics in Operational Research, vol. 23, pp. 316–343, 2022.
17. M. A. Hakmi, M. Bentobache, and M. O. Bibi, *A hybrid direction method for linear fractional programming*, Statistics, Optimization and Infomation Computing, vol. 13, pp. 922–948, 2025.
18. A. Hebbache, *Two-phase adaptive method for solving linear optimization problems*, Master thesis, Higher Normal School of Laghouat, 2017 (in French).
19. A. Hebbache, M. Bentobache, and M. O. Bibi, *An initialization technique for the dual support method of linear programming with nonnegative variables*, in K. Barkaoui, N. Gmati, and M. Roche (eds) African Conference on Research in Computer Science and Applied Mathematics. CARI 2024. Trends in Mathematics. Birkhäuser, Cham. https://doi.org/10.1007/978-3-031-90510-0_27, 2025.
20. L. V. Kantorovich, *Mathematical methods in the organization and planning of production*, Publication House of the Leningrad State University, 1939. Translated in management science, vol. 6, pp. 366–422, 1960.
21. N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica, vol. 4, pp. 373–395, 1984.
22. L. G. Khachian, *A polynomial algorithm for linear programming*, Soviet Mathematics Doklady, vol. 20, pp. 191–194, 1979.
23. A. Makhorin, *GNU Linear Programming Kit, Reference Manual Version 4.9. Draft Edition*, Software available at www.gnu.org/software/glpk/glpk.html, 2006.
24. S. Mehrotra, *On the implementation of a (primal-dual) interior point method*, SIAM Journal on Optimization, vol. 2, pp. 575–601, 1992.
25. K. Paparrizos, N. Samaras, and G. Stephanides, *A new efficient primal dual simplex algorithm*, Computer & Operations Research, vol. 30, pp. 1383–1399, 2003.
26. N. Polskas, and N. Samaras, *Linear programming with MATLAB*, Springer-Verlag, New York, 2017.
27. B. C. Wolde, and T. Larsson, *A steepest feasible direction method for linear programming. Derivation and embedding in the simplex method*, Operations Research and Decisions, vol. 34, No. 2, pp. 163–182, 2024, DOI: 10.37190/ord240210