STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **16**, September 2026, pp 2509–2525.
Published online in International Academic Press (www.IAPress.org)

The Dual Simplex Method for Solving Bounded-Variable Linear Programs

Khalil Djeloud *

Department of Mathematics-Higher Normal School of Laghouat, Algeria

Abstract In this paper, we propose a novel and robust dual simplex algorithm for solving bounded-variable linear programming based on a structured dual support framework and an enhanced long-step rule. The method enables stable and efficient updates of both the support and the associated co-solution. Its effectiveness is supported by a rigorous theoretical analysis: we prove a sign-preservation property of the support co-solutions and derive an explicit expression for the variation of the objective function, ensuring controlled and monotonic improvement and yielding strong convergence guarantees. Extensive experiments on NETLIB problems benchmarks demonstrate that the proposed approach is competitive with the dual simplex implementation of GLPK, confirming its efficiency and reliability.

Keywords Linear programming, dual simplex method, algorithm implementation, open-source solver GLPK, NETLIB library, numerical experiments.

AMS 2010 subject classifications 90C05, 65K05

DOI: 10.19139/soic-2310-5070-3489

1. Introduction

Linear programming (LP) is a mathematical technique for determining the best solution to a problem whose data and unknowns satisfy a series of linear equations and inequalities. Many economic and industrial phenomena can be modeled by mathematical systems of linear inequalities and equalities, which lead to linear optimization problems. In these linear optimization problems, one seeks to minimize or maximize a linear function under linear constraints on the variables of the problem.

In 1955, G.B. Dantzig developed, in collaboration with A. Orden and P. Wolfe, the primal dual simplex method [14]. In 1959, G.F. Hadley and M.A. Simonnard developed the two-phase method [26]. In [18], R. Gabasov and F. M. Kirillova generalized the simplex method by developing the support method that can be initialized by any feasible solution and any support (set of basic indices). This method uses a suboptimality criterion that stops the algorithm when an ϵ-optimal solution is obtained. Later, they developed an adapted method [19],[17],[16],[20] based on an improvement direction that depends on the current solution. The adapted method is applied to solve optimal control problems [20],[16],[28] and generalized to solve convex quadratic problems [21],[22],[5],[27].

In [9] and [10], the authors developed a new method with hybrid direction for linear programming (HDLP) for solving bounded-variable linear programming problems. This method uses the concept of hybrid direction, allowing a transition from one feasible solution to another with an improved objective function value.

Later, this algorithm was extended to solve linear fractional programming problems [23]. The stopping criterion of the HDLP algorithm relies on a quantity called the optimality estimate, thus ensuring convergence to an optimal solution. Moreover, the support update is performed using the long-step rule [15, 4, 8].

*Correspondence to: Khalil Djeloud (Email: k.djeloud@ens-lagh.dz).

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

However, although this method proved competitive with the dual simplex algorithm implemented in the GNU Linear Programming Kit (GLPK) for randomly generated linear problems, it showed inferior performance on NETLIB benchmark problems.

In this work, we propose a dual simplex algorithm for solving bounded-variable linear programming problems. The proposed method employs the long-step rule to compute the new support J_B, the co-solution δ, and the updated objective function value, while the set J_{BNO} is used as a stopping criterion to ensure optimality.

The main contribution of this paper lies in the development of a structured dual support framework within the dual simplex method that efficiently handles bounded-variable problems. Unlike existing approaches, the proposed algorithm provides a systematic mechanism for updating both the support and co-solution, leading to improved computational behavior and solution accuracy.

In addition, the proposed method is underpinned by a rigorous and carefully structured theoretical analysis that highlights its robustness and reliability. In particular, we establish and rigorously prove two fundamental results. Lemma 3 guarantees a sign-preservation property of the sequence of support co-solutions $\delta^{(k)}$, ensuring a stable and consistent evolution of the dual iterates and preventing undesirable oscillatory behavior. Lemma 4 provides a precise characterization of the variation of the objective function ψ between successive iterations through an explicit analytical expression, revealing a controlled and under standard assumptions, monotonic improvement of the objective value.

The complete proofs of Lemma 3 and Lemma 4 are provided in the paper, thereby offering a solid theoretical foundation that supports the stability, convergence, and overall reliability of the proposed algorithm.

To assess its performance, the algorithm is implemented in C++ and compared with the dual simplex method available in the open-source solver GLPK [3]. Numerical experiments on benchmark problems from the NETLIB library [25] demonstrate the effectiveness and competitiveness of the proposed approach.

The rest of the article is organized as follows: in Section 2, we present the dual problem and give some definitions. In Section 3, we present the proposed algorithm and we give a numerical example for illustration purposes. In Section 4, we present numerical results, that compare our algorithm with the dual simplex algorithm of the open-source solver GLPK. Finally, Section 5 concludes the paper by summarizing the main findings of the proposed approach.

2. The Dual Problem and Preliminaries

The linear programming problem with bounded variables has the following standard form :

$$\begin{aligned} \max\ & z = c^T x \\ \text{s.t.}\quad & Ax = b,\ l \le x \le u, \end{aligned} \tag{1}$$

where c, x, l, and $u \in \mathbb{R}^n$ such that $\|l\| < \infty$, $\|u\| < \infty$; $b \in \mathbb{R}^m$; A is an $(m \times n)$ matrix, with $rankA = m < n$.

Let us define the following sets of indices:

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

We can partition v, an arbitrary n-vector, and the matrix A as follows:

$$v^T = (v_B^T, v_N^T),\ \text{where } v_B = v(J_B) = (v_j,\ j \in J_B),\ v_N = v(J_N) = (v_j,\ j \in J_N),$$

$$A = (A_B, A_N),\ \text{where } A_B = A(I, J_B),\ A_N = A(I, J_N).$$

- The set J_B is called the support of the problem (1) if $\det(A_B) \neq 0$.
- A vector x satisfying the constraints of the problem (1) is called a feasible solution (FS).
- An FS x^0 is called optimal if it realizes the maximum of the functional $z(x)$ of the problem (1).
- A solution x^ϵ is called ϵ-optimal if $z(x^0) - z(x^\epsilon) \le \epsilon$, where x^0 is an optimal solution of the problem (1) and $\epsilon \ge 0$.

- The pair $\{x, J_B\}$ formed with the FS x and the support J_B is called a support feasible solution (SFS).

Let φ denote the indicator function of the set of feasible solutions of the problem (1).

Then the primal problem (1) is equivalent to

$$\max_x \ z(x) - \varphi(x).$$

We pose

$$\zeta(x, \gamma) = y^T(Ax - b) + v^T(l - x) + w^T(x - u).$$

where $\gamma = (y, v, w) \in \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R}^n_+$.

If x is FS, we have $\zeta(x, \gamma) \leq 0$, so we can construct the function φ in the following way :

$$\varphi(x) = \max_\gamma \ \zeta(x, \gamma).$$

Since $\min(-z) = -\max\ z$, then the primal problem is equivalent to the problem

$$\max_x \min_\gamma \left(z(x) - \zeta(x, \gamma)\right).$$

The Lagrange function of problem (1) is $L(x, \gamma) = z(x) - \zeta(x, \gamma)$. We can then note that the problem (1), is written

$$\max_x \min_\gamma \ L(x, \gamma).$$

Definition 1
The following linear program is called dual problem of (1) : $\min_\gamma \max_x \ L(x, \gamma)$.

Definition 2
We call the dual function of (1), the function $\psi(\gamma) = \max_x L(x, \gamma)$.

Lemma 1
Let x and γ be, respectively, feasible solutions of the primal and dual problems. Then

$$\min_\gamma \ \max_x L(x, \gamma) \ \geq \ \max_x \ \min_\gamma L(x, \gamma).$$

Proof
For all x and γ, we have

$$L(x, \gamma) \ \geq \ \min_\gamma L(x, \gamma).$$

Taking the maximum over x yields

$$\max_x L(x, \gamma) \ \geq \ \max_x \min_\gamma L(x, \gamma).$$

Finally, taking the minimum over γ, we obtain

$$\min_\gamma \max_x L(x, \gamma) \ \geq \ \max_x \min_\gamma L(x, \gamma).$$

□

Corollary 1
Let x and λ be, respectively, feasible solutions of the primal and dual problems. Then $\psi(\lambda) \ \geq \ z(x)$.

Theorem 2
Let x and λ be, respectively, feasible solutions to the primal and dual problems. If

$$\psi(\lambda) = z(x),$$

then x and λ are optimal solutions to their respective problems.

Proof
Let $\bar{\lambda}$ be any feasible solution of the dual problem. By Corollary 1, we have

$$\psi(\bar{\lambda}) \ \geq \ z(x).$$

Since $\psi(\lambda) = z(x)$, it follows that

$$\psi(\bar{\lambda}) \ \geq \ \psi(\lambda).$$

Hence, λ is an optimal solution of the dual problem, since the latter is a minimization problem. A similar argument shows that x is an optimal solution of the primal problem. □

To obtain the function ψ, it is necessary to verify the first-order optimality conditions:

$$\frac{\partial L}{\partial x} = c^T - y^T A + v^T - w^T = 0.$$

Hence, the function ψ can be written as

$$\psi(\gamma) = y^T b - v^T l + w^T u.$$

Consequently, the dual problem associated with the primal problem (1) is given by

$$\min \ \psi(\gamma) = y^T b - v^T l + w^T u, \tag{2}$$

$$\text{s.t.} \quad c - A^T y + v - w = 0. \tag{3}$$

- The vector
$$\gamma = (y, v, w) \in \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R}^n_+$$
is called a Dual Feasible Solution (DFS) if it satisfies (3), and it is optimal if it satisfies (2)–(3).
- Define the dual feasible co-solution vector δ by
$$\delta = A^T y - c. \tag{4}$$
- The pair $\{\delta, J_B\}$, formed by a dual feasible co-solution δ and a support J_B, is called a support dual feasible co-solution of problem (2)–(3).
- A support dual feasible co-solution $\{\delta, J_B\}$ is said to be non-degenerate if
$$\delta_j \neq 0, \ j \in J_N.$$
- The vector δ^0 is said to be an optimal dual feasible co-solution if there exists a dual feasible solution $\gamma^0 = (y^0, v^0, w^0)$ that minimizes the functional ψ over problem (2)–(3), and such that
$$\delta^0 = A^T y^0 - c.$$

The problem (2)–(3) can be equivalently written as

$$\begin{aligned} \min \ \psi(\gamma) &= y^T b - v^T l + w^T u, \\ -\delta + v &= w. \end{aligned}$$

Substituting $w = -\delta + v$, we obtain

$$\begin{aligned} \min \ \psi(\gamma) &= y^T b - \delta^T u + v^T (u - l), \\ v - \delta &\in \mathbb{R}^n_+. \end{aligned}$$

Equivalently,

$$\begin{aligned} \min \ \psi(\gamma) &= y^T b - \delta^T u + \sum_{j \in J} v_j (u_j - l_j), \\ v_j - \delta_j &\in \mathbb{R}_+, \ j \in J. \end{aligned}$$

- For $y \in \mathbb{R}^m$, we have $\delta = A^T y - c$. A feasible dual solution $\lambda = (y, v, w)$ defined by

$$\begin{aligned} v_j = \delta_j,\ w_j = 0, \text{ if } \delta_j \geq 0, \\ v_j = 0,\ w_j = -\delta_j, \text{ if } \delta_j < 0,\ j \in J, \end{aligned} \tag{5}$$

  is called a dual feasible solution generated for problem (2)–(3).
- Finally, let $\{\delta, J_B\}$ be a support dual feasible co-solution of (2)–(3), and consider the vector $\kappa = (\kappa_B, \kappa_N)$ defined by

$$\begin{aligned} \kappa_j = l_j, \text{ if } \delta_j > 0, \\ l_j \leq \kappa_j \leq u_j, \text{ if } \delta_j = 0, \\ \kappa_j = u_j, \text{ if } \delta_j < 0,\ j \in J_N, \end{aligned} \tag{6}$$

  and

$$\kappa_B = A_B^{-1}\big(b - A_N \kappa_N\big). \tag{7}$$

  Then κ is called a support pseudo-feasible solution of problem (1).

Optimality criterion

Theorem 3
Let $\{\delta, J_B\}$ be a dual feasible support co-solution of problem (2)-(3), and let κ be the associated pseudo-solution. Then, the relations

$$\begin{aligned} \kappa_j = l_j, \text{ for } \delta_j > 0; \\ \kappa_j = u_j, \text{ for } \delta_j < 0; \\ l_j \leq \kappa_j \leq u_j, \text{ for } \delta_j = 0,\ j \in J_B. \end{aligned} \tag{8}$$

are sufficient for the optimality of $\{\delta, J_B\}$. Moreover, if the co-solution δ is non-degenerate, then conditions (8) are also necessary. In this case, the pseudo-solution κ is an optimal solution of problem (1).

Proof
Sufficiency. Let $\lambda = (y, v, w)$ be dual feasible and define $\delta = A^T y - c$, with

$$v = \max(\delta, 0), \quad w = \max(-\delta, 0).$$

Let $\bar{\lambda} = (\bar{y}, \bar{v}, \bar{w})$ be any dual feasible point and set

$$\Delta(\cdot) := \bar{(\cdot)} - (\cdot), \quad \bar{\delta} = \bar{v} - \bar{w}.$$

Using (2)–(3) and $A\kappa = b$, we obtain

$$\psi(\bar{\lambda}) - \psi(\lambda) = \sum_{j \in J} \Big[\Delta\delta_j \kappa_j - \Delta v_j l_j + \Delta w_j u_j\Big] = \sum_{j \in J} \Big[\Delta v_j(\kappa_j - l_j) + \Delta w_j(u_j - \kappa_j)\Big].$$

We distinguish three cases:

Case 1: $\delta_j > 0$**.** Then $v_j = \delta_j$ and $w_j = 0$, hence $\kappa_j = l_j$. In this case,

$$\Delta w_j = \bar{w}_j - 0 = \bar{w}_j \geq 0,$$

and

$$\psi(\bar{\lambda}) - \psi(\lambda) = \sum_{j \in J} \Delta w_j (u_j - l_j) \geq 0$$

Case 2: $\delta_j < 0$**.** Then $v_j = 0$ and $w_j = -\delta_j$, hence $\kappa_j = u_j$. In this case,

$$\Delta v_j = \bar{v}_j - 0 = \bar{v}_j \geq 0,$$

and

$$\psi(\bar{\lambda}) - \psi(\lambda) = \sum_{j \in J} \Delta v_j (u_j - l_j) \geq 0$$

Case 3: $\delta_j = 0$**,** $j \in J_B$**.** Then $v_j = w_j = 0$ and $l_j \leq \kappa_j \leq u_j$. Hence,

$$\psi(\bar{\lambda}) - \psi(\lambda) = \sum_{j \in J} \Delta v_j (\kappa_j - l_j) + \Delta w_j (u_j - \kappa_j) \geq 0,$$

Thus $\psi(\bar{\lambda}) - \psi(\lambda) \geq 0$ for all dual feasible $\bar{\lambda}$, and λ is optimal.

Necessity (non-degenerate case). Let $\{\delta, J_B\}$ be a non-degenerate optimal co-solution of the problem (2)–(3), and suppose that the relations (8) are not satisfied. Then, there exists an index $j_0 \in J_{BNO}$ (see Section 3, relation (11)), such that

$$\alpha_{j_0} \neq 0,$$

where

$$\alpha_j = \begin{cases} \kappa_j - l_j, & \text{if } \delta_j > 0 \text{ or else } \delta_j = 0 \text{ and } \kappa_j < l_j, \\ \kappa_j - u_j, & \text{if } \delta_j < 0 \text{ or else } \delta_j = 0 \text{ and } \kappa_j > u_j \end{cases}$$

We construct a perturbed feasible co-solution $\bar{\delta} = \delta + \sigma t$, where $\sigma > 0$ and $t = t(J)$ satisfies :

$$\begin{aligned} t_{j_0} &= -\operatorname{sign}(\alpha_{j_0}),\ t_j = 0,\ j \neq j_0,\ j \in J_B, \\ t_N &= t_B^T A_B^{-1} A_N, \end{aligned} \tag{9}$$

Then t defines a feasible direction that preserves the sign pattern of δ, i.e.,

$$\delta_j(\delta_j + \sigma t_j) \geq 0 \quad \text{for all } j \in J,$$

for all sufficiently small $\sigma > 0$. Using the expression of the dual objective variation, we have

$$\psi(\bar{\lambda}) - \psi(\lambda) = \sum_{j \in J} \Big[\Delta v_j (\kappa_j - l_j) + \Delta w_j (u_j - \kappa_j) \Big].$$

Using $v_j = \max(\delta_j, 0)$ and $w_j = \max(-\delta_j, 0)$, a direct case analysis yields

$$\Delta v_j = \begin{cases} \sigma t_j, & \delta_j > 0, \\ \max(\sigma t_j, 0), & \delta_j = 0, \\ 0, & \delta_j < 0, \end{cases} \qquad \Delta w_j = \begin{cases} 0, & \delta_j > 0, \\ \max(-\sigma t_j, 0), & \delta_j = 0, \\ -\sigma t_j, & \delta_j < 0. \end{cases}$$

$$\psi(\bar{\lambda}) - \psi(\lambda) = \sum_{\delta_j > 0, j \in J} \sigma t_j (\kappa_j - l_j) - \sum_{\delta_j < 0, j \in J} \sigma t_j (u_j - \kappa_j) + \sum_{\delta_j = 0, t_j > 0, j \in J} \sigma t_j (\kappa_j - l_j) + \sum_{\delta_j = 0, t_j < 0, j \in J} \sigma t_j (\kappa_j - u_j) \tag{10}$$

Since δ is non-degenerate, this reduces to

$$\psi(\bar{\lambda}) - \psi(\lambda) = \sigma t_{j_0} \alpha_{j_0} + \sum_{\delta_j > 0, j \in J_N} \sigma t_j (\kappa_j - l_j) - \sum_{\delta_j < 0, j \in J_N} \sigma t_j (u_j - \kappa_j).$$

From the definition of κ in (6), it follows that

$$\psi(\bar{\lambda}) - \psi(\lambda) = -\sigma |\alpha_{j_0}| < 0.$$

Therefore,

$$\psi(\bar{\lambda}) < \psi(\lambda),$$

which contradicts the optimality of δ.

Consequently, if $\{\delta, J_B\}$ is a non-degenerate optimal co-solution of the problem (2)-(3), then the relations (8) must necessarily hold. □

Lemma 2
If $y^T = c_B^T A_B^{-1}$ and $l_B \leq \kappa_B \leq u_B$, then κ is an optimal feasible solution to the problem (1).

Proof
If $y^T = c_B^T A_B^{-1}$, then $\delta_B = 0$. Hence, by Theorem 3, it follows that κ is an optimal FS to the problem (1). □

3. Algorithm Iteration

Let $\{\delta, J_B\}$ be a support dual feasible co-solution, and let κ denote the associated pseudo-solution. We begin by defining the set of non-optimal basic indices

$$J_{BNO} = \left\{ j \in J_B : \ (\delta_j > 0 \text{ and } \kappa_j \neq l_j) \text{ or } (\delta_j < 0 \text{ and } \kappa_j \neq u_j) \text{ or } (\delta_j = 0 \text{ and } \kappa_j \notin [l_j, u_j]) \ \right\}. \tag{11}$$

This set contains all basic indices that violate the optimality conditions and defines the candidates for leaving the basis. For each $j \in J_{BNO}$, we define

$$\alpha_j = \begin{cases} \kappa_j - l_j, & \text{if } \delta_j > 0, \text{ or if } \delta_j = 0 \text{ and } \kappa_j < l_j, \\ \kappa_j - u_j, & \text{if } \delta_j < 0, \text{ or if } \delta_j = 0 \text{ and } \kappa_j > u_j. \end{cases}$$

The leaving index j_0 is selected as

$$|\alpha_{j_0}| = \max_{j \in J_{BNO}} |\alpha_j|. \tag{12}$$

We then define the search direction t by

$$t_{j_0} = -\operatorname{sign}(\alpha_{j_0}), \ t_j = 0 \ (j \in J_B, \ j \neq j_0), \quad t_N = t_B^T A_B^{-1} A_N. \tag{13}$$

Finally, we introduce the index sets

$$J_0^+ = \{j \in J_N : \delta_j = 0, \ t_j > 0\}, \qquad J_0^- = \{j \in J_N : \delta_j = 0, \ t_j < 0\},$$

and define the scalar quantity

$$\alpha = -|\alpha_{j_0}| + \sum_{j \in J_0^+} t_j(\kappa_j - l_j) + \sum_{j \in J_0^-} t_j(\kappa_j - u_j). \tag{14}$$

3.1. Updating the Dual Solution with the Short Step Rule

Let $\{\lambda, J_B\}$ be an initial feasible dual solution. The objective of the algorithm is to construct an optimal solution λ^0. At each iteration, the algorithm transitions from $\{\lambda, J_B\}$ to a new pair $\{\bar{\lambda}, \bar{J}_B\}$ such that

$$\psi(\bar{\lambda}) \leq \psi(\lambda).$$

To this end, a new feasible dual solution $\bar{\lambda}$ is defined by

$$\bar{\lambda} = \lambda + \sigma^0 t, \quad \sigma^0 \geq 0,$$

where $t \in \mathbb{R}^n$ is the dual direction and σ^0 is the step length along this direction. To minimize the increase given by (10), the step length σ^0 is chosen as large as possible. However, σ^0 must satisfy the feasibility condition

$$\bar{\delta}_j \, \delta_j \geq 0, \quad \forall j \in J,$$

taking into account that α, defined by relation (14), satisfies $\alpha \leq 0$. This yields the following constraints :

$$\delta_j^2 + \sigma^0 \delta_j t_j \geq 0, \quad \forall j \in J_N, \tag{15}$$

$$\delta_{j_0}^2 + \sigma^0 t_{j_0} \delta_{j_0} \geq 0. \tag{16}$$

Case of relation (16). The maximum admissible step length σ_{j_0} can be characterized without enumerating all individual cases. If $\delta_{j_0} = 0$, then $\sigma_{j_0} = +\infty$. Otherwise, for $\delta_{j_0} \neq 0$, we have

$$t_{j_0} = -\text{sign}(\alpha_{j_0}),$$

and

$$\sigma_{j_0} = \begin{cases} |\delta_{j_0}|, & \text{if } \text{sign}(\alpha_{j_0}) = \text{sign}(\delta_{j_0}), \\ +\infty, & \text{otherwise.} \end{cases} \tag{17}$$

Here, α_{j_0} is defined by relation (12).

Case of relations (15). Two cases may arise :

1. If $\alpha \leq 0$:
   - If $t_j \delta_j < 0$, then the maximal step length is
   $$\sigma_j = -\frac{\delta_j}{t_j}.$$
   - If $t_j \delta_j \geq 0$, then
   $$\sigma_j = +\infty.$$
2. If $\alpha > 0$: In this case, the quantity
   $$\alpha' = \sum_{j \in J_0^+} t_j(\kappa_j - l_j) + \sum_{j \in J_0^-} t_j(\kappa_j - u_j)$$
   is nonzero. To ensure that $\psi(\bar{\lambda}) \leq \psi(\lambda)$, this quantity must be eliminated. This is achieved by taking
   $$\sigma_j = 0, \quad \text{if } j \in J_0^- \text{ and } \kappa_j \neq u_j,$$
   $$\sigma_j = 0, \quad \text{if } j \in J_0^+ \text{ and } \kappa_j \neq l_j.$$

Finally, to guarantee that $\bar{\delta}_j \, \delta_j \geq 0$ for all $j \in J$, we set

$$\sigma^0 = \min\{\sigma_{j_0'}, \sigma_{j_0}\},$$

where

$$\sigma_{j_0'} = \min_{j \in J_N} \sigma_j.$$

The previous steps are summarized in the following procedure.

Procedure 1
(Computation of $\bar{\delta}$, $\bar{J}_B$, and $\bar{\psi}$ using the short step rule)

1. Compute α using (14).
2. Compute σ_{j_0} using (17).
3. Compute
   $$\sigma_{j_0'} = \min_{j \in J_N} \sigma_j,$$
   where
   $$\sigma_j = \begin{cases} -\dfrac{\delta_j}{t_j}, & \text{if } \delta_j t_j < 0, \\ 0, & \text{if } j \in J_0^-, \ \kappa_j \neq u_j, \ \alpha > 0, \\ 0, & \text{if } j \in J_0^+, \ \kappa_j \neq l_j, \ \alpha > 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{18}$$

4. Compute
$$\sigma^0 = \min\{\sigma_{j_0'}, \sigma_{j_0}\}.$$
   (a) If $\sigma^0 = +\infty$, then the problem (1) is infeasible.
   (b) If $\sigma^0 = \sigma_{j_0}$, set $\bar{J}_B = J_B$.
   (c) Otherwise, set
$$\bar{J}_B = (J_B \setminus \{j_0\}) \cup \{j_0'\}.$$
5. Compute
$$\bar{\delta} = \delta + \sigma^0 t, \quad \bar{\psi} = \psi + \sigma^0 \alpha.$$

3.2. Updating the Dual Solution with the Long-Step Rule

Compute σ_{j_0} using (17) and the values $\{\sigma_j : j \in J_N\}$ using (18). Next, sort the values $\{\sigma_j : j \in J_N \cup \{j_0\}\}$ in nondecreasing order:
$$\sigma_{i_1} \leq \cdots \leq \sigma_{i_{q+1}}, \text{ where } |J_N| = q.$$
Set
$$\sigma_{i_0} = 0, \quad \delta^{(0)} = \delta.$$
Define the index set
$$\mathcal{T} := \{r \in \mathbb{N} : \sigma_{i_r} < +\infty\} = \{0, 1, \ldots, p\}, \quad p \leq q + 1. \tag{19}$$
To each σ_{i_r}, $r \in \mathcal{T}$, associate the vector
$$\delta^{(r)} = \delta + \sigma_{i_r} t,$$
which represents a support co-solution, where t is the dual direction defined in (13).

Lemma 3
Let $\left(\delta^{(k)}\right)_{k\in\mathcal{T}}$ be a sequence of support co-solutions of problem (2)–(3). Then, for all $k \in \mathcal{T} \setminus \{p\}$,
$$\delta_j^{(k)}\, \delta_j^{(k+1)} \geq 0, \quad \forall j \in J.$$

Proof
We begin by observing that
$$\delta_j^{(0)}\, \delta_j^{(1)} \geq 0, \quad \forall j \in J.$$
Let $k \in \{1, \ldots, p-1\}$. For each $j \in J$, we have
$$\begin{cases} \delta_j\, \delta_j^{(k)} \leq 0, & \text{if } j = i_r,\ r < k,\ \sigma_{i_r} < +\infty, \\ \delta_j\, \delta_j^{(k)} \geq 0, & \text{otherwise.} \end{cases}$$
Moreover,
$$\delta_j^2\, \delta_j^{(k)}\, \delta_j^{(k+1)} \geq 0, \quad \forall j \in J.$$
Since $\delta_j^2 \geq 0$ for all $j \in J$, it follows that $\delta_j^{(k)}$ and $\delta_j^{(k+1)}$ have the same sign. Hence,
$$\delta_j^{(k)}\, \delta_j^{(k+1)} \geq 0, \quad \forall j \in J,\ \forall k \in \mathcal{T} \setminus \{p\}.$$

□

Let $\lambda^{(k)}$ be a dual solution associated with the co-solution $\delta^{(k)}$. For $k \in \mathcal{T} \setminus \{p\}$, we have
$$\delta^{(k+1)} = \delta^{(k)} + (\sigma_{i_{k+1}} - \sigma_{i_k})\, t.$$

By Lemma 3, it follows that for all $j \in J$,

$$\delta_j^{(k+1)} > 0 \;\Rightarrow\; \big(\delta_j^{(k)} = 0\big) \;\vee\; \big(\delta_j^{(k)} > 0\big),$$
$$\delta_j^{(k+1)} < 0 \;\Rightarrow\; \big(\delta_j^{(k)} = 0\big) \;\vee\; \big(\delta_j^{(k)} < 0\big).$$

By iterating this argument, we obtain for all $j \in J$:

$$\delta_j^{(k)} > 0 \;\Rightarrow\; \big(\delta_j^{(k-1)} = 0\big) \;\vee\; \big(\delta_j^{(k-2)} = 0\big) \;\vee\; \cdots \vee\; \big(\delta_j^{(1)} = 0\big) \;\vee\; \big(\delta_j = 0\big) \;\vee\; \big(\delta_j > 0\big), \tag{20}$$

$$\delta_j^{(k)} < 0 \;\Rightarrow\; \big(\delta_j^{(k-1)} = 0\big) \;\vee\; \big(\delta_j^{(k-2)} = 0\big) \;\vee\; \cdots \vee\; \big(\delta_j^{(1)} = 0\big) \;\vee\; \big(\delta_j = 0\big) \;\vee\; \big(\delta_j < 0\big). \tag{21}$$

Lemma 4
For $k \in \mathcal{T} \setminus \{p\}$, the variation of the function ψ between $\lambda^{(k)}$ and $\lambda^{(k+1)}$ is given by

$$\psi(\lambda^{(k+1)}) - \psi(\lambda^{(k)}) = (\sigma_{i_{k+1}} - \sigma_{i_k})\Bigg(- |\alpha_{j_0}| + \sum_{\substack{j \in J_N \\ \delta_j = 0,\ t_j > 0}} t_j(\kappa_j - l_j) + \sum_{\substack{j \in J_N \\ \delta_j = 0,\ t_j < 0}} t_j(\kappa_j - u_j) + \cdots$$
$$+ \cdots + \sum_{\substack{j \in J_N \\ \delta_j^{(k)} = 0,\ t_j > 0}} t_j(\kappa_j - l_j) + \sum_{\substack{j \in J_N \\ \delta_j^{(k)} = 0,\ t_j < 0}} t_j(\kappa_j - u_j)\Bigg).$$

Proof
Let

$$\lambda^{(k)} = \big(y^{(k)}, v^{(k)}, w^{(k)}\big)$$

be a dual feasible solution, and define

$$\delta^{(k)} = A^T y^{(k)} - c = \delta + \sigma_{i_k} t.$$

Then,

$$\Delta\delta^{(k)} := \delta^{(k+1)} - \delta^{(k)} = (\sigma_{i_{k+1}} - \sigma_{i_k})\, t.$$

so that for each $j \in J$,

$$\Delta\delta_j^{(k)} = (\sigma_{i_{k+1}} - \sigma_{i_k})\, t_j.$$

Recall that

$$v_j^{(k)} = \max(\delta_j^{(k)}, 0), \qquad w_j^{(k)} = \max(-\delta_j^{(k)}, 0).$$

Hence,

$$\Delta v_j^{(k)} = \begin{cases} \Delta\delta^{(k)}, & \delta_j^{(k)} > 0, \\ \max\big(\Delta\delta^{(k)}, 0\big), & \delta_j^{(k)} = 0, \\ 0, & \delta_j^{(k)} < 0, \end{cases} \qquad \Delta w_j^{(k)} = \begin{cases} 0, & \delta_j^{(k)} > 0, \\ \max\big(-\Delta\delta^{(k)}, 0\big), & \delta_j^{(k)} = 0, \\ -\Delta\delta^{(k)}, & \delta_j^{(k)} < 0. \end{cases}$$

By definition of ψ, we write

$$\psi(\lambda^{(k+1)}) - \psi(\lambda^{(k)}) = (\Delta y^{(k)})^T (A\kappa) - (\Delta v^{(k)})^T l + (\Delta w^{(k)})^T u$$
$$= \sum_{j \in J} \Big[(\Delta\delta_j^{(k)})\kappa_j - (\Delta v_j^{(k)}) l_j + (\Delta w_j^{(k)}) u_j\Big].$$

We denote

$$\Phi_j^{(k)} = (\Delta\delta_j^{(k)})\kappa_j - (\Delta v_j^{(k)}) l_j + (\Delta w_j^{(k)}) u_j,$$

so that

$$\psi(\lambda^{(k+1)}) - \psi(\lambda^{(k)}) = \sum_{j\in J} \Phi_j^{(k)}.$$

We decompose $J = J_B \cup J_N$. On the basic set, only the leaving index j_0 contributes. Using the definition of α_{j_0}, we obtain

$$\sum_{j\in J_B} \Phi_j^{(k)} = (\sigma_{i_{k+1}} - \sigma_{i_k})\, t_{j_0}\alpha_{j_0}.$$

Since $t_{j_0}\alpha_{j_0} = -|\alpha_{j_0}|$, it follows that

$$\sum_{j\in J_B} \Phi_j^{(k)} = -(\sigma_{i_{k+1}} - \sigma_{i_k})\, |\alpha_{j_0}|.$$

For each $j \in J_N$, the value of $\Phi_j^{(k)}$ depends on the signs of $\delta_j^{(k)}$ and $\delta_j^{(k+1)}$.

Case 1 : $\delta_j^{(k)} \geq 0$ and $\delta_j^{(k+1)} \geq 0$, in this case,

$$v_j^{(k)} = \delta_j^{(k)}, \quad v_j^{(k+1)} = \delta_j^{(k+1)}, \quad w_j^{(k)} = w_j^{(k+1)} = 0.$$

Hence

$$\Delta v_j^{(k)} = \Delta\delta_j^{(k)}, \quad \Delta w_j^{(k)} = 0,$$

and therefore

$$\Phi_j^{(k)} = \Delta\delta_j^{(k)}(\kappa_j - l_j) = (\sigma_{i_{k+1}} - \sigma_{i_k})\, t_j(\kappa_j - l_j).$$

Case 2 : $\delta_j^{(k)} < 0$ and $\delta_j^{(k+1)} < 0$, here,

$$w_j^{(k)} = -\delta_j^{(k)}, \quad w_j^{(k+1)} = -\delta_j^{(k+1)}, \quad v_j^{(k)} = v_j^{(k+1)} = 0,$$

so

$$\Delta w_j^{(k)} = -\Delta\delta_j^{(k)}, \quad \Delta v_j^{(k)} = 0,$$

which gives

$$\Phi_j^{(k)} = \Delta\delta_j^{(k)}(\kappa_j - u_j) = (\sigma_{i_{k+1}} - \sigma_{i_k})\, t_j(\kappa_j - u_j).$$

Case 3 : $\delta_j^{(k)} < 0$ and $\delta_j^{(k+1)} \geq 0$, then by Lemma 3 we necessarily have

$$\delta_j^{(k+1)} = 0.$$

Thus

$$v_j^{(k+1)} = 0, \quad w_j^{(k+1)} = 0,$$

and a direct computation yields

$$\Phi_j^{(k)} = 0.$$

Case 4 : $\delta_j^{(k)} \geq 0$ and $\delta_j^{(k+1)} < 0$, then necessarily

$$\delta_j^{(k)} = 0.$$

Hence

$$\Phi_j^{(k)} = (\sigma_{i_{k+1}} - \sigma_{i_k})\, t_j(\kappa_j - u_j).$$

Combining all cases and using (20)–(21) together with (6), we obtain

$$\sum_{j\in J_N}\Phi_j^{(k)} = (\sigma_{i_{k+1}}-\sigma_{i_k})\Bigg(\sum_{\substack{j\in J_N\\ \delta_j=0,\ t_j>0}} t_j(\kappa_j-l_j)+\sum_{\substack{j\in J_N\\ \delta_j=0,\ t_j<0}} t_j(\kappa_j-u_j)+\cdots$$
$$+\cdots+\sum_{\substack{j\in J_N\\ \delta_j^{(k)}=0,\ t_j>0}} t_j(\kappa_j-l_j)+\sum_{\substack{j\in J_N\\ \delta_j^{(k)}=0,\ t_j<0}} t_j(\kappa_j-u_j)\Bigg).$$

Finally, we have

$$\begin{aligned}\psi(\lambda^{(k+1)})-\psi(\lambda^{(k)}) &= \sum_{j\in J_B}\Phi_j^{(k)}+\sum_{j\in J_N}\Phi_j^{(k)}\\ &= (\sigma_{i_{k+1}}-\sigma_{i_k})\Bigg(-|\alpha_{j_0}|+\sum_{\substack{j\in J_N\\ \delta_j=0,\ t_j>0}} t_j(\kappa_j-l_j)+\sum_{\substack{j\in J_N\\ \delta_j=0,\ t_j<0}} t_j(\kappa_j-u_j)+\cdots\\ &\quad+\cdots+\sum_{\substack{j\in J_N\\ \delta_j^{(k)}=0,\ t_j>0}} t_j(\kappa_j-l_j)+\sum_{\substack{j\in J_N\\ \delta_j^{(k)}=0,\ t_j<0}} t_j(\kappa_j-u_j)\Bigg).\end{aligned}$$

□

We have

$$\sum_{k=0}^{p-1}\big[\psi(\lambda^{(k+1)})-\psi(\lambda^{(k)})\big] = \psi(\lambda^{(p)})-\psi(\lambda).$$

Assume that there exists an index $s_0\in\{0,\dots,p-1\}$ such that

$$\sum_{k=0}^{s_0-1}\big[\psi(\lambda^{(k+1)})-\psi(\lambda^{(k)})\big]<0,$$

while

$$\sum_{k=0}^{s_0}\big[\psi(\lambda^{(k+1)})-\psi(\lambda^{(k)})\big]\geq 0.$$

In this case, we define a new dual solution by

$$\sigma^0=\sigma_{i_{s_0}},\quad \bar{\delta}=\delta+\sigma^0 t,$$

and the new support is given by

$$\bar{J}_B=(J_B\setminus\{j_0\})\cup\{i_{s_0}\}.$$

We now provide a summary of the main steps involved in computing the new co-solution vector $\bar{\delta}$, the updated support $\bar{J}_B$, and the associated function value $\bar{\psi}$ based on the long-step rule. The detailed procedure follows.

Procedure 2
(Computation of $\bar{\delta}$, $\bar{J}_B$, and $\bar{\psi}$ using the long-step rule)

1. Compute α using (14).
2. Compute σ_{j_0} using (17).

3. For each $j \in J_N$, define
$$\sigma_j = \begin{cases} -\frac{\delta_j}{t_j}, & \text{if } \delta_j t_j < 0, \\ 0, & \text{if } j \in J_0^-,\ \kappa_j \neq u_j,\ \alpha > 0, \\ 0, & \text{if } j \in J_0^+,\ \kappa_j \neq l_j,\ \alpha > 0, \\ +\infty, & \text{otherwise.} \end{cases}$$
4. Determine the index set $\mathcal{T}$ using (19), and sort $\{\sigma_{j_r} : r \in \mathcal{T}\}$ in nondecreasing order.
$$0 = \sigma_{i_0} \leq \sigma_{i_1} \leq \cdots \leq \sigma_{i_p}, \quad \text{where} \quad p \leq |J_N| + 1.$$
5. For each $r \in \mathcal{T} \setminus \{p\}$, compute
$$\Delta V_r = \sum_{j \in J_N\ \delta_j^{(r)}=0,\ t_j>0} t_j(\kappa_j - l_j) + \sum_{j \in J_N\ \delta_j^{(r)}=0,\ t_j<0} t_j(\kappa_j - u_j).$$
6. Compute
$$V_r = -|\alpha_{j_0}| + \sum_{k=0}^{r} \Delta V_k, \qquad r \in \mathcal{T}.$$
7. Let s_0 be the smallest index such that
$$V_{s_0} \geq 0.$$
8. Set $\sigma^0 = \sigma_{i_{s_0}}$ and :
   (a) If $\sigma^0 = +\infty$, then problem (1) is infeasible.
   (b) If $\sigma^0 = \sigma_{j_0}$, set
   $$\bar{J}_B = J_B, \quad \bar{\delta} = \delta^{(s_0)}.$$
   (c) Otherwise, set
   $$\bar{J}_B = (J_B \setminus \{j_0\}) \cup \{i_{s_0}\}, \quad \bar{\delta} = \delta^{(s_0)}.$$
9. Update
$$\bar{\psi} = \psi + \sum_{k=0}^{s_0} \left(\sigma_{i_{k+1}} - \sigma_{i_k}\right) V_k.$$

3.3. Dual Support Algorithm

Let $\{x, J_B\}$ be an initial SFS of problem (1), and let $y \in \mathbb{R}^m$. The dual support algorithm for solving linear programs with bounded variables is described by the following steps :

1. Compute the feasible co-solution using (4).
2. Compute the vectors $v, w \in \mathbb{R}^n$ using (5), and define
$$\psi = y^T b - v^T l + w^T u.$$
3. Compute the pseudo-solution κ using (6)–(7).
4. Perform the optimality test for the support feasible pseudo-solution $\{\kappa, J_B\}$:
   (a) Determine the set of non-optimal basic indices J_{BNO} using (11).
   (b) If $J_{BNO} = \emptyset$, then stop. The solution $\{\kappa, J_B\}$ is an optimal SFS for problem (1).
   (c) Otherwise, select the index j_0 using (12).
5. Compute the direction t using (13).
6. Compute the following quantities:

- the new support $\bar{J}_B$,
- the new co-solution $\bar{\delta}$,
- the new objective value $\bar{\psi}$,

using either the short step rule (Procedure 1) or the long step rule (Procedure 2).

7. Update:
$$\delta \leftarrow \bar{\delta}, \quad \psi \leftarrow \bar{\psi}, \quad J_B \leftarrow \bar{J}_B,$$
and return to Step 3.

3.4. Numerical Example

Consider the following linear programming problem with bounded variables:

$$\begin{aligned} \max \quad & z = c^T x \\ \text{s.t.} \quad & Ax = b, \quad l \le x \le u, \end{aligned} \tag{22}$$

where

$$A = \begin{pmatrix} 6 & -3 & 1 & 0 \\ 1 & 9 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} \frac{31}{2} \\ 37 \end{pmatrix}, \quad c = \begin{pmatrix} 8 \\ 1 \\ 0 \\ 0 \end{pmatrix}, l = \begin{pmatrix} 2 \\ 2 \\ \frac{1}{2} \\ 14 \end{pmatrix}, \quad u = \begin{pmatrix} 5 \\ 5 \\ \frac{25}{2} \\ 20 \end{pmatrix}.$$

Iteration 1

1. The initial support is $J_B = \{2, 3\}$ and $J_N = \{1, 4\}$.
2. Let $y = \left(0, \frac{1}{9}\right)^T$.
3. The co-solution is $\delta = A^T y - c = \left(\frac{-71}{9}, 0, 0, \frac{1}{9}\right)^T$.
4. The vectors $v = \left(0, 0, 0, \frac{1}{9}\right)^T$, $w = \left(\frac{71}{9}, 0, 0, 0\right)^T$, and the dual objective value $\psi = 42$.
5. The pseudo-solution is $\kappa = \left(5, 2, \frac{-17}{2}, 14\right)^T$.
6. The set of non-optimal basic indices is $J_{BNO} = \{3\}$, thus $j_0 = 3$.
7. The direction vector is $t = \left(\frac{19}{3}, 0, 1, \frac{1}{3}\right)^T$.
8. The step length σ^0 is computed as $\alpha = -9$, $\sigma_{j_0} = +\infty$, $\sigma_{j_0'} = \sigma_1 = \frac{71}{19}$. Hence, $\sigma^0 = \min \sigma_{j_0'}, \sigma_{j_0} = \frac{71}{19}$.
9. The updated quantities are $\bar{\delta} = \left(0, 0, \frac{71}{57}, \frac{10}{19}\right)^T$, $\bar{J}_B = \{1, 2\}$, $\bar{J}_N = \{3, 4\}$, $\bar{\psi} = \frac{585}{19}$.

Iteration 2

1. The current support is $J_B = \{1, 2\}$ and $J_N = \{3, 4\}$.
2. The co-solution is $\delta = \left(0, 0, \frac{71}{57}, \frac{10}{19}\right)^T$, with dual objective value $\psi = \frac{585}{19}$.
3. The pseudo-solution is $\kappa = \left(\frac{68}{19}, \frac{41}{19}, \frac{1}{2}, 14\right)^T$.
4. Since $J_{BNO} = \emptyset$, the algorithm terminates. Therefore, $x = \left(\frac{68}{19}, \frac{41}{19}, \frac{1}{2}, 14\right)^T$ is an optimal solution of problem (22), with $z = \frac{585}{19}$.

4. Experimental Results

In order to perform a numerical comparison between the dual simplex algorithm implemented in the GNU Linear Programming Kit (GLPK) [3] and the algorithm proposed in Section 3.3, a dedicated implementation was developed in the C++ programming language.

All algorithms were executed on a personal computer equipped with an Intel Core i7-4790 processor (3.60 GHz) and 8 GB of RAM, running under Windows 10.

The numerical experiments were conducted on a subset of NETLIB test problems [25] involving bounded variables. The main characteristics of these problems are reported in Table 1, where :

- $nrows$: number of constraints,
- $ncols$: number of variables,
- neq: number of equality constraints,
- $nonzeros$: number of nonzero elements in the constraint matrix,
- obj: optimal objective function value.

Problem	**ncols**	**nrows**	**neq**	**nonzero**	**obj**
boeing1	384	351	9	3865	-3.3521×10^2
boeing2	143	166	4	1339	-3.1502×10^2
etamacro	688	400	272	2489	-7.5572×10^2
finnis	614	497	47	2714	1.7279×10^5
fit1p	1677	627	627	10894	9.1464×10^3
fit1d	1026	24	1	14430	-9.1464×10^3
fit2d	10500	25	1	138018	-6.8464×10^4
fit2p	13525	3000	3000	50284	6.8464×10^4
recipe	180	91	67	752	-2.6662×10^2
nesm	2923	662	480	13988	1.4076×10^7
ganges	16816	1309	1284	7021	-1.0959×10^5
gfrd-pnc	1092	616	548	3467	6.9022×10^6
grow15	645	300	300	5665	-1.0687×10^8
grow22	946	440	440	8318	-1.6083×10^8
grow7	301	140	140	2633	-4.7788×10^7
kb2	41	43	16	291	-1.7499×10^3
pilot	3652	1441	233	43220	-5.5749×10^2
seba	1028	515	507	4874	1.3331×10^4
shell	1775	536	534	4900	1.2088×10^9

Table 1. NETLIB bounded variable problems data

The implementation of the proposed method relies on the following components:

1. The support set J_B is extracted directly from the GLPK simplex basis using the routine **`glp_get_bhead`** [3]. For each row $i = 1, \ldots, m$, this routine returns the index of the basic variable occupying the i-th position in the basis. These indices are used to construct the set J_B, which defines the basis matrix A_B.
2. The dual vector y is computed by solving the linear system

$$A_B^T y = c_B,$$

using the GLPK routine **`glp_btran`** [3], which performs a backward transformation based on the LU factorization of the basis matrix.
3. The vector κ_B is obtained by solving the linear system

$$A_B \kappa_B = z, \quad \text{where} \quad z = b - A_N \kappa_N.$$

4. The vector t_N is computed as
$$t_N = z^T A_N,$$
where z is the solution of the system
$$A_B^T z = t_B.$$

The proposed algorithm was compared with the dual simplex method implemented in GLPK. The results are reported in Table 2, where :

- CPU: computation time in seconds,
- nit: number of iterations,
- DS-SSR: algorithm of Section 3.3 with the short step rule,
- DS-LSR: algorithm of Section 3.3 with the long step rule,
- DS-GLPK: dual simplex algorithm from GLPK [3].

Problem	**DS-GLPK**		**DS-SSR**		**DS-LSR**	
	nit	**CPU (s)**	**nit**	**CPU (s)**	**nit**	**CPU (s)**
boeing1	872	0.234	870	0.230	870	0.232
boeing2	248	0.094	280	0.150	248	0.100
etamacro	935	0.218	1240	0.290	856	0.210
finnis	357	0.213	320	0.190	308	0.180
fit1p	3499	1.310	4240	1.780	1430	0.550
fit1d	568	0.343	41	0.024	41	0.026
fit2d	5875	10.840	88	0.160	62	0.110
fit2p	33359	55.661	29737	42.610	29737	42.600
recipe	56	0.125	54	0.120	34	0.080
nesm	3779	1.622	5218	2.250	1905	0.830
ganges	1645	0.702	1523	0.640	1523	0.640
gfrd-pnc	667	0.250	667	0.250	666	0.250
grow15	3378	1.123	2287	0.770	2287	0.760
grow22	5824	1.794	5517	1.690	5517	1.700
grow7	1270	0.234	1333	0.250	1330	0.245
kb2	93	0.094	108	0.130	108	0.110
pilot	8940	12.433	9837	12.680	3180	3.960
seba	391	0.218	392	0.222	392	0.220
shell	830	0.312	857	0.320	830	0.310

Table 2. Number of iterations and CPU time of the three algorithms (DS-GLPK, DS-SSR, and DS-LSR) on NETLIB problems.

The experimental results demonstrate the effectiveness of the proposed algorithm, which performs particularly well on large-scale problems with many bounded variables. Notably, the long-step rule algorithm (DS-LSR), despite its higher implementation complexity and potential numerical challenges, proves highly competitive with the dual simplex implementation of the GNU Linear Programming Kit (GLPK), often delivering substantial reductions in both iteration counts and computation time.

5. Conclusion

This paper presented an efficient algorithm for solving bounded-variable linear programming problems within the dual simplex framework. The proposed method demonstrates solid theoretical foundations and effectively handles bounded-variable structures. Numerical results confirm its efficiency and competitiveness, particularly for large-scale problems, where the long-step rule achieves significant improvements in computational performance compared to GNU Linear Programming Kit (GLPK). Overall, the proposed method represents a valuable contribution to linear optimization, combining theoretical rigor with practical efficiency. Future work may focus on improving numerical stability and further enhancing the algorithm's performance.

REFERENCES

1. P.-Q. Pan, *Linear Programming Computation*, Springer, 2014.
2. M. Huang, Y. Zhong, H. Yang, J. Wang, F. Zhang, B. Bai, and L. Shi, *Simplex initialization: A survey of techniques and trends*, arXiv preprint arXiv:2111.03376, 2021.
3. A. Makhorin, *GLPK (GNU Linear Programming Kit)*, Available at: https://www.gnu.org/software/glpk/, 2008.
4. E. Kostina, *The long step rule in the bounded-variable dual simplex method: Numerical experiments*, Mathematical Methods of Operations Research, vol. 55, pp. 413–429, 2002.
5. E. A. Kostina and O. I. Kostyukova, *An algorithm for solving quadratic programming problems with linear equality and inequality constraints*, Computational Mathematics and Mathematical Physics, vol. 41, no. 7, pp. 960–973, 2001.
6. N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Proceedings of the 16th Annual ACM Symposium on Theory of Computing, pp. 302–311, 1984.
7. L. G. Khachiyan, *A polynomial algorithm for linear programming*, Doklady Akademii Nauk SSSR, vol. 244, pp. 1093–1096, 1979.
8. M. Bentobache and M. O. Bibi, *Numerical Methods of Linear and Quadratic Programming*, French Academic Presses, Germany, 2016.
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, e5836, 2021.
10. K. Djeloud, *Une nouvelle direction pour l'optimisation linéaire et application au contrôle optimal*, PhD thesis, 2021.
11. E. R. Bixby, *Implementing the simplex method: The initial basis*, ORSA Journal on Computing, vol. 4, no. 3, pp. 267–284, 1992.
12. E. R. Bixby and A. Martin, *Parallelizing the dual simplex method*, INFORMS Journal on Computing, vol. 12, no. 1, pp. 45–56, 2000.
13. G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, 2016.
14. G. B. Dantzig, A. Orden, and P. Wolfe, *The generalized simplex method for minimizing a linear form under linear inequality constraints*, Pacific Journal of Mathematics, vol. 5, no. 2, pp. 183–195, 1955.
15. F. M. Kirillova, R. Gabasov, and O. I. Kostyukova, *A method of solving general linear programming problems*, Doklady AN BSSR, vol. 23, pp. 197–200, 1979.
16. R. Gabasov, F. M. Kirillova, and S. V. Prischepova, *Optimal Feedback Control*, Springer, 1995.
17. R. Gabasov and F. M. Kirillova, *Adaptive method of solving linear programming problems*, Preprint, University of Karlsruhe, Institute for Statistics and Mathematics, 1994.
18. R. Gabasov and F. M. Kirillova, *Methods of Linear Programming, Parts 1–3: Special Problems*, Minsk, Belarus, 1977–1980.
19. R. Gabasov, F. M. Kirillova, and E. A. Kostina, *Adaptive methods for solving minimax problems*, Optimization, vol. 50, no. 1–2, pp. 61–92, 2001.
20. R. Gabasov and F. M. Kirillova, *Optimal controllers and their applications*, IFAC Proceedings Volumes, vol. 33, no. 16, pp. 113–118, 2000.
21. R. Gabasov, F. M. Kirillova, and V. M. Raketskii, *On methods for solving the general problem of convex quadratic programming*, Doklady Akademii Nauk, vol. 258, no. 6, pp. 1289–1293, 1981.
22. R. Gabasov, F. M. Kirillova, and O. I. Kostyukova, *Solution of linearly quadratic extremal problems*, Doklady Akademii Nauk, vol. 280, no. 3, pp. 529–533, 1985.
23. M. A. Hakmi, M. Bentobache, and M. O. Bibi, *A hybrid direction method for linear fractional programming*, Statistics, Optimization & Information Computing, vol. 13, no. 3, pp. 922–948, 2025.
24. A. Hebbache, M. Bentobache, and M. O. Bibi, *Dual support M-method for solving linear programs with non-negative variables*, Statistics, Optimization & Information Computing, vol. 16, no. 2, pp. 1727–1745, 2026.
25. M. D. Gay, *Electronic mail distribution of linear programming test problems*, Mathematical Programming Society COAL Newsletter, vol. 13, pp. 10–12, 1985.
26. G. F. Hadley and M. A. Simonnard, *A simplified two-phase technique for the simplex method*, Naval Research Logistics Quarterly, vol. 6, no. 3, pp. 221–226, 1959.
27. B. Brahmi and M. O. Bibi, *Dual support method for solving convex quadratic programs*, Optimization, vol. 59, no. 6, pp. 851–872, 2010.
28. M. A. Zaitri, M. O. Bibi, and M. Bentobache, *A hybrid direction algorithm for solving optimal control problems*, Cogent Mathematics & Statistics, vol. 6, no. 1, 1612614, 2019.