Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 7** : 1–9.
ISSN-0037-8712
doi:10.5269/bspm.79556

The Vector $\overrightarrow{0}$ Algorithm for Solving Optimal Strategy Problems in Matrix Games and Experimental Computation in Matlab Environment

Van Hoan Tran

ABSTRACT: From the Min-Cone Method presented in [1], we have built a Vector $\overrightarrow{0}$ algorithm to solve the primitive standard linear programming problem with an objective function with non-negative coefficients. A computer implementation of this algorithm has been developed in the MATLAB environment to solve primal standard linear programming problems of arbitrary size. We present experimental results obtained from randomly generated data on the same problem, including applications to optimal strategy problems in matrix games, using the Vector $\overrightarrow{0}$ algorithm and comparing it with the Simplex method. The experimental results with random data show that, compared with the Simplex algorithm, the proposed algorithm requires significantly fewer iterations and substantially less computation time. We expect that this algorithm will be an effective approach for solving medium-and large-scale linear programming problems on commonly available computers today.

Key Words: Strategy problems, Min-Cone, Vector $\overrightarrow{0}$ algorithm, Matlab.

Contents

1. Introduction

As we know, most practical problems can be easily reduced to the primal standard linear programming problems with the objective function with non-negative coefficients through a simple elementary transformation. Therefore, in this section we will from the cone-min method presented in [1] to build a the algorithm to solve the primal standard linear programming problems with the objective function with non-negative coefficients.

We present the results of experimental calculations on common computers to solving with medium and large size linear programming problems, with random data in the Matlab environment to investigate the relationship between the number of iterations and calculation time.

2020 *Mathematics Subject Classification:* 90C05.
Submitted October 15, 2025. Published February 21, 2026

2. Preliminaries

Consider the primal standard linear programming problem with the objective function with non-negative coefficients:

$$(P): \begin{cases} f(x) = \langle C, x\rangle = \sum_{j=1}^{n} c_j \cdot x_j \to \min, \\ x \geq 0, & (2.1) \\ \langle C^i, x\rangle \geq d_i, \quad i = 1, 2, \ldots, N. & (2.2) \end{cases}$$

Where $x, C^i, C \in \mathbb{R}^n, x(x_1, x_2, \ldots, x_n), C^i(c_{i1}, c_{i2}, \ldots, c_{in}), C(c_1, c_2, \ldots, c_n)$ are the any vector, for $i = 1; 2; \ldots; N(N \geq 1), j = 1; 2; \ldots; n$. We rewrite the linear programming problem (P) in the following form:

$$(P_+): \begin{cases} f(x) = \langle C, x\rangle = \sum_{j=1}^{n} c_j x_j \to \min, \\ \langle A^i, x\rangle \geq b_i, \quad i = 1, 2, \ldots, m & (2.3) \end{cases}$$

Where

$$\begin{aligned} m &= n + N; \\ A^i &= e^i \quad (i = 1, \ldots, n); \quad b_i = 0 \quad (i = 1, \ldots, n); \\ A^{n+i} &= C^i \quad (i = 1, \ldots, N); \quad b_{n+i} = d_i \quad (i = 1, \ldots, N). \end{aligned}$$

e^i is a unit vector with the ith component receiving the value 1, and the other components receiving the value 0. Let us recall some basic concepts and definitions.

2.1. Concept of The Simple Polyhedral Cone

From the constraint system of problem (P_+), consider the set M is the set of points x that satisfy the constraint system:

$$\langle A^i, x\rangle \geq b_i, \forall i \in I \tag{2.4}$$

In which $I := \{i_1; i_2; \ldots; i_n\} \subset \{1; 2; \ldots; m\}$, $|I| = n$ ($|I|$ is the number of elements in set I) and A^i where $i \in I$ is a linearly independent system. The set M determined by (2.4) is a polyhedral convex set [4], is the intersection of n closed halfspaces and it has a unique extreme point x^M which is the solution (determined) satisfying the following system of equations:

$$\langle A^i, x\rangle = b_i, \forall i \in I \tag{2.5}$$

The set M is called a simple polyhedral cone of the problem (P_+) [1], [2]. Point x^M is called the vertex of cone M. The vector system A^i where $i \in I$ is called the basic of the cone M, also known as the basic of the vertex x^M. Set I is called the base index set of the M cone.

2.2. The Concept of the Edge of the Simple Polyhedral Cone

The following concepts can be found in [1], [2], [3], [4]. Assume that the cone M is determined from (2.4) with the base index set I as above. For each $i \in I$, the set of points $x \in \mathbb{R}^n$ satisfying the system:

$$\langle A^r, x\rangle = b_r, \forall r \in I \setminus \{i\} \tag{2.6}$$

is called the line i of the cone M.

The set of points x satisfying the system

$$\begin{cases} \langle A^r, x\rangle = b_r, \forall r \in I \setminus \{i\} \\ \langle A^i, x\rangle \geq b_i, \end{cases}$$

is called edge i of the cone M.

For each $i \in I$, the vector $z^i_M (i \in I)$, determined from the system of equations:

$$\begin{cases} \langle A^r, z^i_M\rangle = 0, \forall r \in I, r \neq i, \\ \langle A^i, z^i_M\rangle = 1, \end{cases} \tag{2.7}$$

is called the direction vector of the edge i of the cone M.

The x^M vertex of the cone M can be determined from (2.5), in case we know the directional vector system $z^i_M (i \in I)$, we can use the following formula [1]:

$$x^M = \sum_{i\in I} b_i \cdot z^i_M \tag{2.8}$$

Theorem 2.1 *x^M is the vertex of the cone M determined from (2.5), and if the direction vector $z^i_M(i \in I)$ of the edge i of the cone M is determined from (2.7) then we can determine vertex x^M from the following formula:*

$$x^M = \sum_{i\in I} b_i \cdot z^i_M$$

2.3. Definition of Minimum Cone (Min-Cone)

Definition 2.1 *A simple polyhedral cone M of the problem (P_+), with vertex x^M is called the minimum (Min-Cone) of the objective function $f(x)$, if $f(x^M) \leq f(x)$ for all $x \in M$.*

It is easy to see, since the coefficient of the objective function of the problem (P_+) are non-negative, so the cone $M_0 = R^n_+$ is a cone-min, vertex is the origin of coordinate $\overrightarrow{0} = (0, 0, \ldots, 0)$, the vectors of directions of the edges are $z^i_{M_0} = e^i$ $(i = 1, 2, \ldots, n)$.

3. The Vector $\overrightarrow{0}$ algorithm and illustrative examples

From the Min-Cone Method presented in [1], we build the following the Vector $\overrightarrow{0}$ algorithm to solve the primal standard linear programming problem with the objective function with non-negative coefficients (P_+):

3.1. The Vector $\overrightarrow{0}$ algorithm

Initialization (iteration 0). Select the initial Min-Cone $M_0 = R^n_+$ is a cone-min, vertex is the origin of coordinate $\overrightarrow{0} = (0, 0, \ldots, 0)$, the base index set is $I_0 = \{1, 2, \ldots n\}$ the vectors of directions of the edges are $z^i_{M_0} = e^i$ $(i = 1, 2, \ldots, n)$.

Iteration $k(k = 0; 1; 2; \ldots)$. M_k is the Min-Cone (already built) of the (P_+) problem, with the set of base indices, vertices, and the direction vectors of the edges of the cone M_k are

$$I_k := \left\{i^k_1, i^k_2, ..., i^k_n\right\}; x^k = x^{M_k} = \sum_{i\in I_k} b_i.z^i_k \text{ and } z^i_k = z^i_{M_k}$$

Denoting

$$\Delta^k_i = \langle A^i, x^k\rangle - b_i, i = 1; 2; 3; \ldots; m \tag{3.1}$$

Determine set $I^-\left(x^k\right)$

$$I^-(x^k) := \left\{i \in \{1; 2; ...; m\} \,|\Delta^k_i < 0\right\}$$

We have two cases:

- Case 1: If $I^-\left(x^k\right) = \varnothing$ then stop, x^k is a solution to the problem (P),
- Case 2: If $I^-\left(x^k\right) \neq \varnothing$, we choose the index s_k entering into the base by one of the following three ways:

  We choose s_k as the satisfaction index

$$\Delta^k_{s_k} = \min\left\{\Delta^k_i < 0 | i \in I^-(x^k)\right\} \tag{3.2}$$

Or choose s_k is any index of $I^-\left(x^k\right)$ or we choose $s_k = \min\left\{i \mid i \in I^-\left(x^k\right)\right\}$ (called the min selection rule) or we choose $s_k = \max\left\{i \mid i \in I^-\left(x^k\right)\right\}$ (called the max selection rule) and determine:

$$I^{s_k} := \left\{i \in I_k \middle| \left\langle A^{s_k}, z_k^i\right\rangle \neq 0\right\}$$

$$I_+^{s_k} := \left\{i \in I^{s_k} \middle| \left\langle A^{s_k}, z_k^i\right\rangle > 0\right\} = \left\{i_{s_k 1}^k, i_{s_k 2}^k, ..., i_{s_k q_k}^k\right\} \tag{3.3}$$

We have two cases:

– Case 2.1: If $I_+^{s_k} = \emptyset$ then stop, deduce the problem (P) has no feasible point (The constraint domain of problem (P) is empty).

– Case 2.2: If $I_+^{s_k} \neq \emptyset$:
Denoting

$$V^{s_k} := \left\{ v \in I_+^{s_k} \mid \frac{\langle C, z_k^v\rangle}{\langle A^{s_k}, z_k^v\rangle} = \min_{i \in I_+^{s_k}} \left\{ \frac{\langle C, z_k^i\rangle}{\langle A^{s_k}, z_k^i\rangle} \right\} \right\} \tag{3.4}$$

We choose the index r_k going out the base by one of the following three ways:
We choose r_k as any index belonging to V^{s_k} or we choose

$$r_k = \min\{v \mid v \in V^{s_k}\} \quad \text{or} \quad r_k = \max\{v \mid v \in V^{s_k}\} \tag{3.5}$$

(The way to choose this index is called the min selection rule (or the max selection rule)).
We construct the new cone M_{k+1} from the Min-Cone M_k, the new base index set is $I_{k+1} = (I_k \cup \{s_k\}) \setminus \{r_k\}$; and the direction vectors z_{k+1}^i [1], [2]:

$$z_{k+1}^i = \begin{cases} \left(z_k^i - \dfrac{\langle A^{s_k}, z_k^i\rangle}{\langle A^{s_k}, z_k^{r_k}\rangle} \cdot z_k^{r_k} \right) & \text{if } i \in I_k, i \neq r_k \\ \dfrac{1}{\langle A^{s_k}, z_k^{r_k}\rangle} \cdot z_k^{r_k} & \text{if } i = s_k \end{cases} \tag{3.6}$$

Vertex x^{k+1} is determined according:

$$x^{k+1} = \sum_{i \in I_{k+1}} b_i \cdot z_{k+1}^i \tag{3.7}$$

Return to iteration k with $k := k+1$ ($k \leftarrow k+1$).
As with the simplex method, to facilitate the application of the Vector $\overrightarrow{0}$ algorithm to solve problem (P_+), we set up the following table:

Table 1: Vector $\overrightarrow{0}$ Algorithm

Base Index	b_j	$c_1, c_2, \ldots, c_j \ldots, c_n$						Δ_i^k, $k = 0, 1, 2, \ldots$	
1	b_1	a_{11}	a_{12}	$\ldots$	a_{1j}	$\ldots$	a_{1n}	Δ_1^k	
2	b_2	a_{21}	a_{22}	$\ldots$	a_{2j}	$\ldots$	a_{2n}	Δ_2^k	
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
(s_k)	b_{s_k}	$a_{s_k 1}$	$a_{s_k 2}$	$\ldots$	$a_{s_k j}$	$\ldots$	$a_{s_k n}$	$(\Delta_{s_k}^k)$	
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
m	b_m	a_{m1}	a_{m2}	$\ldots$	a_{mj}	$\ldots$	a_{mn}	Δ_m^k	

1 2 $\dots$ (r_0) $\dots$ n	0 0 $\dots$ 0 $\dots$ 0	$1\ \ 0\ \dots\ 0\ \dots\ 0$ $0\ \ 1\ \dots\ 0\ \dots\ 0$ $\dots\ \dots\ \dots\ \dots\ \dots\ \dots$ $0\ \ 0\ \dots\ 1\ \dots\ 0$ $\dots\ \dots\ \dots\ \dots\ \dots\ \dots$ $0\ \ 0\ \dots\ 0\ \dots\ 1$	$\langle A^{s_0}, z_0^1\rangle$ $\dots$ $[\langle A^{s_0}, z_0^{r_0}\rangle]$ $\dots$ $\langle A^{s_0}, z_0^{2n}\rangle$	$\frac{\langle C, z_k^1\rangle}{\langle A^{s_0}, z_0^1\rangle}$ $\dots$ $\frac{\langle C, z_0^{r_0}\rangle}{\langle A^{s_0}, z_0^{r_0}\rangle}$ $\dots$ $\frac{\langle C, z_0^{2n}\rangle}{\langle A^{s_0}, z_0^{2n}\rangle}$
		$0, 0, \dots, 0, \dots, 0$		
i_1^k i_2^k $\dots$ $\left(r_k = i^k_{s_k p_k}\right)$ $\dots$ i_n^k	$b_{i_1^k}$ $b_{i_2^k}$ $\dots$ b_{r^k} $\dots$ $b_{i_n^k}$	$z_{k1}^{i_1^k}\ z_{k2}^{i_1^k}\ \dots\ z_{kj}^{i_1^k}\ \dots\ z_{kn}^{i_1^k}$ $z_{k1}^{i_2^k}\ z_{k2}^{i_2^k}\ \dots\ z_{kj}^{i_2^k}\ \dots\ z_{kn}^{i_2^k}$ $\dots\ \dots\ \dots\ \dots\ \dots\ \dots$ $z_{k1}^{r_k}\ z_{k2}^{r_k}\ \dots\ z_{kj}^{r_k}\ \dots\ z_{kn}^{r_k}$ $\dots\ \dots\ \dots\ \dots\ \dots\ \dots$ $z_{k1}^{i_n^k}\ z_{k2}^{i_n^k}\ \dots\ z_{kj}^{i_n^k}\ \dots\ z_{kn}^{i_n^k}$	$\langle A^{s_k}, z_k^{i_1^k}\rangle$ $\langle A^{s_k}, z_k^{i_2^k}\rangle$ $\dots$ $[\langle A^{s_k}, z_k^{r_k}\rangle]$ $\dots$ $\langle A^{s_k}, z_k^{i_n^k}\rangle$	$\frac{\langle C, z_k^{i^k_{s_{k1}}}\rangle}{\langle A^{s_k}, z_k^{i^k_{s_{k1}}}\rangle}$ $\dots$ $\left(\frac{\langle C, z_k^{r_k}\rangle}{\langle A^{s_k}, z_k^{r_k}\rangle}\right)$ $\dots$ $\frac{\langle C, z_k^{i^k_{s_k q_k}}\rangle}{\langle A^{s_k}, z_k^{i^k_{s_k q_k}}\rangle}$
Iteration $k =$ $0; 1; \dots$	x^k	$x_1^k\ \ x_2^k\ \dots\ x_j^k\ \dots\ x_n^k$		

3.2. Illustrative example

Find the optimal strategy of the player P_1 and the player P_2 with the game matrix:
$A = \begin{bmatrix} 5 & 4 & 3 & 1 \\ 4 & 3 & 6 & 5 \end{bmatrix}$
Pair of dual problems of P_1 and P_2:

$$\begin{cases} f_1 = x'_1 + x'_2 \to \min \\ 5x'_1 + 4x'_2 \geq 1 \\ 4x'_1 + 3x'_2 \geq 1 \\ 3x'_1 + 6x'_2 \geq 1 \\ x'_1 + 5x'_2 \geq 1 \\ x'_1 \geq 0, x'_2 \geq 0 \end{cases}$$

$$\begin{cases} f_2 = y'_1 + y'_2 + y'_3 + y'_4 \to \max \\ 5y'_1 + 4y'_2 + 3y'_3 + y'_4 \leq 1 \\ 4y'_1 + 3y'_2 + 6y'_3 + 5y'_4 \leq 1 \\ y'_1 \geq 0, y'_2 \geq 0, y'_3 \geq 0, y'_4 \geq 0 \end{cases}$$

According to the the Vector $\overrightarrow{0}$ algorithm, we have the initial Min-Cone of the main problem which is a positive orthant cone with a vertex of the origin $\overrightarrow{0} = (0, 0)$ with a set of basic indices of $\{1; 2\}$ have a compact Min-Cone table for solving the problem after 2 iterations as follows:

Table 2: Computation table using the Vector $\overrightarrow{0}$ algorithm for Example

Base Index	b_j	1	1	Δ_i^0	Δ_i^1	Δ_i^2
1	0	1	0	0	0	$\frac{2}{17}$
2	0	0	1	0	$\frac{1}{5}$	$\frac{3}{17}$
3	1	5	4	-1	$-\frac{1}{5}$	$\frac{5}{17}$
4	1	4	3	-1	$(-\frac{2}{5})$	0
5	1	3	6	-1	$\frac{1}{5}$	$\frac{7}{17}$
6	1	1	5	(-1)	0	0
1	0	1	0	1	1	
2	0	0	1	[5]	$(\frac{1}{5})$	
Iter 0	x^0	0	0			
1	0	1	$-\frac{1}{5}$	$\left[\frac{17}{5}\right]$	$\left(\frac{4}{17}\right)$	
6	1	0	$\frac{1}{5}$	$\frac{3}{5}$	$\frac{1}{3}$	
Iter 1	x^1	0	$\frac{1}{5}$			
4	1	$\frac{5}{17}$	$-\frac{1}{17}$			
6	1	$-\frac{3}{17}$	$\frac{4}{17}$			
Iter 2	x^2	$\frac{2}{17}$	$\frac{3}{17}$			

If we solve this problem by the Dual Simplex Method [5], [6], [7], [8], we will return the original problem to its canonical form by adding 4 sub-unknowns $x'_3, x'_4, x'_5, x'_6 \geq 0$, we get the problem:

$$\begin{cases} f_1 = x'_1 + x'_2 \to \min \\ -5x'_1 \quad -4x'_2 \quad +x'_3 \qquad\qquad\qquad = -1 \\ -4x'_1 \quad -3x'_2 \qquad +x'_4 \qquad\qquad = -1 \\ -3x'_1 \quad -6x'_2 \qquad\qquad +x'_5 \qquad = -1 \\ -x'_1 \quad -5x'_2 \qquad\qquad\qquad +x'_6 = -1 \\ x'_j \geq 0, j = 1,2,3,4,5,6 \end{cases}$$

after 4 iterations to obtain the solution $x^* = \left(\frac{2}{17}; \frac{3}{17}; \frac{5}{17}; 0; \frac{7}{17}; 0\right)$.

4. Numerical Experimental Results

We have built a computer sample program for the Vector $\overrightarrow{0}$ algorithm to solve the primal standard linear programming problems with the objective function with non-negative coefficients in the Matlab environment. We have performed many experimental calculations to investigate the relation ship between the number of iterations and the size of the Linear Programming problem when solved by the Vector $\overrightarrow{0}$ algorithm proposed in the above section. The classes of experimental problems are classes the primal standard linear programming problems has form of optimal strategy problem in matrix game. Below are test calculation results with random data tables can be performed on any common computer today for form problem of the optimal strategy in matrix game [9], [10].

4.1. Experimental computational results comparing the Simplex method and the Vector $\overrightarrow{0}$ algorithm for solving of optimal strategy problems in matrix game

As we know, the optimal strategy problems in matrix games can be reduced to the following primal standard linear programming problem [1], [8]
$f = x'_1 + x'_2 + ... + x'_n \to min$. Subject to:

$$\begin{cases} a_{11}x'_1 + a_{12}x'_2 + \cdots + a_{1n}x'_n \geq 1 \\ a_{11}x'_1 + a_{12}x'_2 + \cdots + a_{1n}x'_n \geq 1 \\ \ldots \\ a_{m1}x'_1 + a_{m2}x'_2 + \cdots + a_{mn}x'_n \geq 1 \\ x'_i \geq 0, i = 1, 2, \ldots, n \end{cases}$$

The table below presents the experimental computational results on the same problem with random data to solve some primal standard linear programming problems has form of optimal strategy problems in matrix game using the Vector $\overrightarrow{0}$ algorithm compared with the Simplex method.

Table 3: Comparison of solving Optimal strategy problems using the Simplex algorithm and the Vector $\overrightarrow{0}$ algorithm.

Problem	**size**$A = m.n$ $A = \text{rand}(m, n)$	**Solved by Simplex Method**		**Solved by Vector $\overrightarrow{0}$ Algorithm**	
		Time (sec)	**iter**	**Time (sec)**	**iter**
1	A=rand(1000,50)	11.472550	1463	0.019955	244
2	A=rand(2000,60)	106.716192	3389	0.034841	383
3	A=rand(3000,70)	360.754671	5157	0.058998	557
4	A=rand(4000,80)	870.144531	7123	0.074320	591
5	A=rand(10000,100)	Timeout >> 1000 (sec)	timeout	0.256948	986
6	A=rand(150000000,5)	Timeout >> 1000 (sec)	timeout	54.918118	27

4.2. Experimental calculation with random data to solve some primal standard linear programming problems in the form of the problem of finding optimal strategy in matrix game by Vector $\overrightarrow{0}$ algorithm

The table 4 below is the experimental calculation results showing the effectiveness of the Vector $\overrightarrow{0}$ algorithm in terms of the number of iterations and the time to get the solution when solving problems in the form of problem of finding optimal strategy in matrix game.

Table 4: The experimental calculation results of the Vector $\overrightarrow{0}$ algorithm when solving Optimal strategy problems.

prob	**size(A)** $A = \text{rand}(m, n)$	**size(b)** $b = \text{ones}(1, m)$	**size(c)** $c = \text{ones}(1, n)$	**Time (sec)**	**iter**
1	A=rand(10000,100)	b=ones(1,10000)	c=ones(1,100)	0.262023	984
2	A=rand(50000,100)	b=ones(1,50000)	c=ones(1,100)	2.209728	1279
3	A=rand(100000,100)	b=ones(1,100000)	c=ones(1,100)	4.911921	1506
4	A=rand(200000,100)	b=ones(1,200000)	c=ones(1,100)	10.448424	1456
5	A=rand(300000,100)	b=ones(1,300000)	c=ones(1,100)	17.532213	1546
6	A=rand(400000,100)	b=ones(1,400000)	c=ones(1,100)	25.000031	1687
7	A=rand(500000,100)	b=ones(1,500000)	c=ones(1,100)	31.996817	1547
8	A=rand(1000000,200)	b=ones(1,1000000)	c=ones(1,200)	488.922097	5744
9	A=rand(1000000,200)	b=ones(1,1000000)	c=ones(1,200)	25.238192	3905
10	A=rand(500000,200)	b=ones(1,500000)	c=ones(1,200)	31.021444	2517
11	A=rand(300000,200)	b=ones(1,300000)	c=ones(1,200)	93.835376	3966
12	A=rand(400000,200)	b=ones(1,400000)	c=ones(1,200)	136.311651	4851
13	A=rand(500000,200)	b=ones(1,500000)	c=ones(1,200)	151.024494	5095
14	A=rand(1000000,200)	b=ones(1,1000000)	c=ones(1,200)	468.457011	5613
15	A=rand(1000000,200)	b=ones(1,1000000)	c=ones(1,200)	198.933564	5599
16	A=rand(1000000,200)	b=ones(1,1000000)	c=ones(1,200)	510.644141	5613
17	A=rand(1000000,200)	b=ones(1,1000000)	c=ones(1,200)	468.457011	5613
18	A=rand(150000000,5)	b=ones(1,150000000)	c=ones(1,5)	53.801432	22

With the randomly data of the problems in the table above, if solved by the Simplex algorithm and the Interior point algorithm, we must reduce the problems to canonical form with a very large number of variables, so the solution will not be obtained within the allowed time.

5. Conclusion

In this paper, we have presented the Vector $\overrightarrow{0}$ algorithm to solve the primitive standard linear programming problem with the objective function having non-negative coefficients. We have built a computer sample program for this algorithm to solve this problems with size any on the Matlab environment. The paper presents the experimental results with random data to solve the standard primal linear programming problems in the form of optimal strategy problems in matrix games using the Vector $\overrightarrow{0}$ algorithm compared with the simplex method with medium and large sizes. The results show that the number of iterations and the calculation time are less and significantly faster. We continue to conduct experimental calculations and hope that the Vector $\overrightarrow{0}$ algorithm will be one of the effective algorithms to solve the primal standard linear programming problems with medium and large sizes on any common computer today.

References

1. Tran, V.H., Nguyen, A.T., *A New Algorithm for Solving the Primal Standard Linear Programming Problem and Numerical Experimental Results*, In: Le Thi, H.A., Pham Dinh, T., Le, H.M. (eds), Modelling, Computation and Optimization in Information Systems and Management Sciences, Lecture Notes in Networks and Systems, vol. 1688, Springer, Cham, (2026). https://doi.org/10.1007/978-3-032-08381-4 26.

2. Tran, V.H., *A new algorithm for solving Primal Standard Two Variables Linear Programming Problem and Numerical Experimental Result*, Proceedings in "Lecture Note and Systerms", Springer-Verlag, 3rd International Conference on Data Analytics and Insights, 12 pages, (2026).

3. Tuy, H., *Convex Analysis and Global Optimization*, Kluwer, (1998).

4. Tuy, H., *Programming Theory, Volume 1*, Science and Technology Publishing House, 202 pages, (1968).

5. Dantzig, G.B., Thapa, M.N., *Linear Programming, 1: Introduction*, Springer-Verlag, New York, (1997).

6. Klee, V., Minty, G.J., *How Good is the Simplex Algorithm?*, In: Shisha, O. (ed.), Inequalities III, Academic Press, New York, pp. 159–175, (1972).

7. Khanh, P.Q., Nuong, T.H., *Linear Programming*, Education Publisher, 457 pages, (2002).

8. Sinh, T.X., *Economic Mathematics (used for economic and engineering branches)*, Hanoi National University Publisher, 156 pages, (2007).

9. Ploskas, N., Samaras, N., *Linear Programming Using MATLAB*, Springer Optimization and Its Applications, vol. 127, Springer, (2017). https://doi.org/10.1007/978-3-319-65919-0.

10. Ferris, M.C., Mangasarian, O.L., Wright, S.J., *Linear Programming With MATLAB*, MPS-SIAM Series on Optimization, Series No. 7, SIAM, (2008).

Van Hoan Tran,
Lac Hong University, Tran Bien Ward, Dong Nai Province
VietNam.
E-mail address: `hoantran@lhu.edu.vn`