ASJET

Al-Salam Journal for Engineering and Technology
Journal Homepage:
http://journal.alsalam.edu.iq/index.php/ajest
e-ISSN: 2790-4822 p-ISSN: 2958-0862

A procedure for choosing incoming and exiting variables in the simplex algorithm

Haidar Mohammad Hani[1]*

[1]Master's Student at Applied Mathematics Department, Faculty of Mathematics, Statistics and Computer Science, University of Tabriz.

*Corresponding Author: Haidar Mohammad Hani

DOI: https://doi.org/10.55145/ajest.2026.05.01.004
Received September 2025; Accepted December 2025; Available online Febuary 2026

ABSTRACT: A new selection of input-output variables for a simplex algorithm in linear programming (LP) problems is introduced in this paper. An appropriate pivot rule to find the entering and leaving variable is one of fundamental steps for simplex method. Unfortunately, such a classical pivot does not only fail to bring about any savings in a associated computational cost, it may also lead to cycling problems (which are typically solved by Bland's scheme) or simply it fails to improve the objective function. Combining the perturbation scheme of suboptimality and cycling, with Dantzig's pivot rule we propose a procedure that addresses these issues. The new rule has an improved algorithm effectiveness in the sense of the reduction for the number of iterations. When compared to those of existing pivot rules, calculational results demonstrate that the suggested rule is not only optimal during each iteration but also facilitates improvements in computation time for linear programming problems. The cycle problem of the Dantzig's simplex pivot rule is settled by a new pivot rule proposed, which similarly as much improves the objective function How possible with every iteration. Moreover, it can have the optimal LP value in less iterations.

Keywords: Linear Programming, Simplex Algorithm, Pivot Rule, Iteration Reduction, Optimization

1. INTRODUCTION

A simplex algorithm is one of best-known solutions methods for linear programming, first introduced in 1947 by George Dantzig. It has been widely applied to multiple fields based on its high computational efficiency [1,2], for example, transportation optimization, production scheduling, and resource allocation.

But, one of the most difficulty on this algorithm is how to decide when an variable comes in and when occurs its output one, and those affects directly its performance and computational speed. In this study, a new pivot rule is proposed to address the limitations of classical methods, which not only overcomes existing issues but also improves the overall performance of the algorithm [3].

Recently, several studies have pointed out that although the simplex method is competitive in practice, particularly for medium- and large-scale problems where the choice of pivot rule impacts its convergence rate as well as its robustness [4,5]. For instance, Kitahara and Mizuno (2019) showed that advanced rules such as the steepest-edge pivot can decrease iterations on nondegenerate problems. Also, GPU based pivoting rules have been developed to exploit the parallel computing potential which result in reduced execution times in large scale industrial environments [6]. Moreover, the rise of alternative solvers such as first-order methods (e.g., Google's PDLP) has motivated a re-examination of simplex pivot strategies. Though it is known that interior-point methods and PDLP have better complexity estimates for very large LPs, the simplex method with a strong pivoting rule is still competitive due to its practical reliability and efficiency in many applied contexts [7-9]. This shows that improving pivot generalization is not only a theoretical game, but also a key to handle the discrepancy between classical algorithms and modern large scale optimization requests.

*Corresponding author: haderhine80@gmail.com

2. Problem statement

In the real world, linear programming problems may involve numerous variables and limitations, particularly in business and industry where accurate on-the-spot solutions are required. Out of all pivot rules, Dantzig ′ s one is the simplest and most efficient one [10,11]. However, it may cause cycling or over-number of iterations in some cases. Hence, devising a plan to mitigate these issues is essential.

3. Research objectives

The primary objectives of this research are as follows:
To lessen the simplex algorithm's necessary iterations.
To avoid cycling while the calculation is being done.
To improve the algorithm's overall performance, particularly for large-scale issues.

4. Literature Review

The efficacy for the linear algorithm and its pivot rules have been extensively studied. A summary of important research in this field is provided below [12]:

Classical Pivot Rules: Dantzig's pivot rule, introduced as the first method for selecting entering variables due to its simplicity and efficiency, has been widely used (Dantzig, 1963). However, this method can lead to cycling and requires a large number of iterations in some cases. Subsequent studies have aimed to address these limitations [13].

Mathematically, Dantzig's rule selects the entering variable e by maximizing the reduced cost:

$$e = \underset{j\in N}{\operatorname{argmax}} \{c_j - z_j\}, \quad z_j = c_B^T B^{-1} A_j$$

where B is the basis matrix, A_j is a j-th column for the constraint matrix, and c_j is a cost coefficient of variable x_j. This choice often accelerates improvement in the objective function but may lead to cycling in degenerate cases. To address this, Bland (1977) proposed the anti-cycling rule, which chooses the **smallest index** among eligible variables:

$$e = \min\{j \in N \mid c_j - z_j > 0\}$$

Ensuring finite termination at the expense of possibly slower convergence.

Improved Rules: Forrest and Goldfarb (1992) introduced the **steepest-edge rule**, where the selected variable is entered. based on the maximum ratio of reduced cost to the Euclidean norm of the associated column:

$$e = \underset{j\in N}{\operatorname{argmax}} \frac{c_j - z_j}{\| B^{-1}A_j \|_2}$$

This approach often decreases the quantity of iterations by accounting for the "steepness" of improvement in the feasible region.

Modern Developments: Kitahara and Mizuno (2019) mathematically demonstrated iteration bounds for steepest-edge rules in nondegenerate LPs, while randomized pivoting strategies (Friedmann et al., 2011) showed that probabilistic selection can achieve subexponential complexity in expectation.

With advances in high-performance computing, GPU-accelerated simplex algorithms adopt the same pivot selection rules but implement matrix updates in parallel, reducing time complexity per iteration from $O(mn)$ to nearly $O(\frac{mn}{p})$, where p is the number of parallel processors.

Finally, machine-learning–based pivoting (Liu, Wang & You, 2024) frames the pivot selection as a classification problem, learning a function $f_\theta(x)$ that predicts the optimal entering variable based on features derived from $B^{-1}A$ and reduced costs, i.e [14].:

$$e = f_\theta\left(c_j - z_j,\ \| B^{-1}A_j \|,\ \text{degeneracy indicators}\right)$$

which dynamically adapts during iterations.

Improvements to Pivot Rules:

Numerous studies have been conducted to enhance Dantzig's pivot rule. For example:

Bland's Rule (1977): The purpose of this rule is to guarantee that the simplex algorithm always converges to the best solution by preventing cycling. However,applying this rule in practice could make computations take longer.

Luchinger-Zander's Rule: According to a sensitivity analysis with respect to OBJ coefficients, the rule effectively reduces number of iteration.

Recent Methods:

Recent results were concerned with the merger of old pivot rules and new ideas. For instance:

Pivoting: One possibility to reduce cycling is by randomly selecting the entering variable.

Hybrid Algorithms: There is a promising performance of combination of the simplex method with metaheuristic approaches like genetic algorithms or simulated annealing [15].

Large-Scale Problems:

In high dimensional problems, obtaining a quicker convergence and computation time is even more relevant. Literature has proved that the performance of Simplex Algorithm can be greatly improved with methods like parallel processing and matrix optimization [16].

Innovation of this research

In this paper, we present a pivot rule that avoids cycling and results in faster convergence. The new approach enhances the performance of the simplex method, particularly for large-size problems by using methods to restrain the search space as well as sensitivity analysis of coefficients [17].

5. RESEARCH METHOD

5.1 Data Collection:

The test aspects of the method were tested on a set of different-sized LP problems. These problems were both scaled examples from reliable sources and industrial optimization problems like production planning and transportation scheduling. Recently, researchers have stressed the maxim that pivot rules should not only be tested on benchmark data sets and large-scale realistic instances to ascertain robustness and computational efficency [18,19].

5.2 Creation for the Suggested Algorithm:

The objective of a proposed pivot rule was to decrease probability of cycling, and at each iteration select a variable which improves the current (in terms of function value) target function as much as possible. This rule is based on sensitivity analysis of coefficients and improvement rates serves of the constraint matrix. Relate strategies using sensitivity analysis and nested pricing have been investigated to improve the choice of entering variables [20]. More recently, hybrid or machine learning (ML) based architectures have been suggested in which pivot decisions are adjusted on the fly based on predictive models.

5.3 Dantzig's Pivot Rule:

Following the linear (continuous) model literature in its standard version, Dantzig's (1963) naive rule sets the benchmark for most comparative analyses. However, theoretical improvements [21, 22] and practical parallel computational optimizations on GPU platforms have been achieved in recent studies [23, 24]. The above contributions have shown that it is crucial for us to reconsider pivot selection both in terms of theory and implementation, particularly in this era of high performance computing [25],

$$\text{Maximize} \quad c^T x \qquad (1)$$

$$\text{Subject to} \begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

Where:

$$A \in R^{m\times n}(m < n), b \in R^m, c \in R^n, \text{rank}(A) = m \qquad (2)$$

It is assumed that A = [B N] after updating (resetting) column A, where B is an invertible matrix of order m×m and N is a matrix of order m×(n-m). In this case, N is referred to as a non-basic matrix and B as the basic matrix. JB and JN stand for a sets for basic and non-basic indices, respectively.

When we look at the equation Ax = b, we assume that $x = \begin{vmatrix} x_B \\ x_N \end{vmatrix}$ is a plausible solution, where $xB = B{-}1b$ and $xN = 0$. One way to rewrite the constraint system $\begin{cases} Ax = b \\ x \geq 0 \end{cases}$ is as follows:

$$\begin{cases} Bx_B + Nx_N = b \\ x \geq 0, X_N \geq 0 \end{cases}$$

A fundamental viable solution of the system is x if $xB \geq 0$. Suppose that a workable solution to system (1) is $\begin{vmatrix} B^{-1}b \\ 0 \end{vmatrix}$ where a value for the objective function, represented by z_0, is computed as follows:

$z0 = BBTB{-}1b$ (3)

Then:

$XB = \bar{b} - \sum\ j \in J_N\ Yjxj$ (4)

In this case, $\bar{b} = B{-}1b$ and $Yj = B{-}1Aj$. The symbols Aj and Yj stand for the j-th column of matrix A and $B{-}1b$, respectively. Given that z is the objective function's value, we obtain

$xB = \underline{z}0 - \sum j \in jN\ \bar{c}Jxj$ (5)

where the lower cost is represented by $\bar{c}J = cj - zj$.

$zj = cBTB{-}1Aj = cBTYj$ (6)

The results indicate that, upon obtaining the optimal solution, the set of indices:

$J = \{j \in JN\top cJ > 0\}$ (7)

will be vacant. Assume the following if the set of indices J is not empty:

$e = argmax\,(\bar{c}_j, j \in J)$ (8)

If the set of indices:

$\{j \in \{1,2, \ldots, m\} \mid Y\,(i, e) > 0\}$ (9)

is empty, then LP (1) is unbounded. Assume that:

$s = argmin\,\{\frac{\bar{b}_i}{y(i,e)},, Y\,(i, e) > 0\}$ (10)

According to Dantzig's rules, the incoming variable's index is e, and the departing variable's index is s. $Y\,(i, e)$ is utilized for the pivot operation. Table 1 displays the simplex approach tableau:

Table 1 Simplex Method Tableau Format

C	cB	CN	
	xB	xN	
cB	In	$Y = B{-}1AN$	$\bar{b} = B{-}1b$
Z	0	$cBTYj$	$z0 = cBTB{-}1b$
$\bar{c}_j$	0	$cj{-}zj{-}$	

Optimal pivot rule

The decision-making rule about the index j, where $\bar{c}J > 0$, is a crucial component in assessing the simplex method's success. It is taken into consideration when entering the basis after the pivot operation. The time complexity for verifying that $\bar{c}J > 0$ for every j is O(m), as is well known. The overall time complexity is around O(mn) if we take into account checking every scenario that might occur in J. The time needed to finish the remaining pivot operations, O(km), is compared to this. Here, k is the number of m pivot operations that must be completed before B−1 is calculated. The selection of pivoting rules, however, has a significant impact on both the overall number for pivots needed to get a best solution (assuming one exists) and the performance of each pivot individually. The time needed to verify the pivot operation, with a complexity of O(m), is as follows for each j with $\bar{c}J > 0$

$t = argmin\,\{\frac{\bar{b}_i}{y(i,e)},, Y\,(i, e) > 0\}$ (11)

The total time needed to examine the maximum (n−m) pivots is O(mn) for any feasible J. The most computationally demanding phase in the simplex method is pivoting, which involves figuring out B−1. This takes more time than it would take to verify every potential pivot in J, with a temporal complexity of O(m3). If there is an ideal solution, the simplex technique may compute it more quickly by reducing the number of pivots, or iterations. It might be argued that further calculations are necessary to determine this ideal pivoting rule. On the other hand, the optimum pivoting rule's efficiency stems from its simple computing cost. Assume:

$$\theta\,(t, j) = \frac{\bar{b_i}}{y_{(t,j)}} \tag{12}$$

The simplex algorithm with optimum pivoting criteria follows these steps:

Step 1: Assume:

$$e = arg_{\max} \quad e = \arg\ amx\{\bar{C}_j\ j\ \in J\ \ N\} \tag{13}$$

The algorithm terminates if:

1. $\overline{ce} > 0$ for all $j \in J\ N$, then xBk, xNk is an optimal solution.

2. If there exists j ∈ JN such that $\bar{C}_J > 0$ and y(i,j) ≤ 0 for all i=1,2,…,m, then the LP problem is unbounded. In this case, the algorithm terminates.

Step 2: To get the best answer, use the given pivoting rule to identify the incoming and exiting variables:

For each $j \in J\ N$ (with $\overline{cj} > 0$), assume $NJ \subseteq J_N^0$ such that $j \in J_N^0$, and we assume that relation (12) holds, and also:

$$\theta\ (s,\ e)\ =\ max\ \{\theta\ (t,\ j)\ \bar{c}\ J:\ j\ \in\ J_{No\ and}\ (t\ =\ argmin\ \{\ \frac{\bar{b_i}}{y_{(i,e)}}\ ,,Y\ (i,\ e)\ >\ 0)\ \} \tag{14}$$

We suppose that the departing variable's index is s and the incoming variable's index is e.

Step 3: Update the tableau, execute the pivot operation for the incoming and exiting variables, and go back to Step 1 if the result is suboptimal.

First definition: A cycle is triggered if the value of the goal function remains unchanged for two successive pivoting steps.

A cycle is a series of pivots that ends up back at the initial set.

First Theorem (Optimal Pivoting Rule Termination): The simplex technique ends in a limited amount of time with an optimum solution and no cycles if it employs the designated optimal pivoting rule.

Evidence:

Assuming pivot operations are used to implement the simplex approach, take into consideration two successive bases, Bk and Bk+1 (k≥1). Let the set of variables with positive reduced costs be $JNk = \{j \in J\ N / \bar{c}J > 0\}$. When $x(j) \in J\ Nk\}$, the objective function's value might increase for each

Assume that $t = argmin\ \{\frac{\bar{b_i}}{y_{(i,e)}},,Y\ (i,\ e) > 0\}$ and $\theta\,(t, j) = \frac{\bar{b_i}}{y_{(i,e)}}$ The objective function value can improve by $\theta\,(t, j)\ \bar{c}J$

If the solution obtained does not change, meaning $\theta\,(t, j) = 0$ for $j \in J\ Nk$, in other words, if $j = argmax\ \{\bar{c}J\ /\ \ j \in J\ Nk\}$ (as per Dantzig's rules), this results in a cycle.

Now suppose that:

$$\theta\,(s, \underline{e}) = max\ \left\{\theta\,(t, j) cJ : j\ \in\ J_n\ ,\ \text{and}\ (t\ =\ \text{argmin}\ \{\frac{\bar{b_i}}{y_{(i,e)}},,Y(i, e)\ >\ 0)\,\}\right. \tag{15}$$

If $\theta(\underline{s,e})cJ > 0$ and we suppose that the values of the objective function are C BkX Bk and C Bk+1x Bk+1, which correspond to $x\ k$ and $x\ Bk$+1, then we get:

$$c\ \ Bk{+}1x\ \ Bk{+}1{=}\ \ c\ \ Bkx\ \ Bk{+}\ \ \theta\ \ (s,\ \ e)\ \ \bar{c}\ \ e\ \ \Longrightarrow\ \ c\ \ Bkx\ \ Bk{<}\ \ c\ \ Bk{+}1x\ \ Bk{+}1 \tag{16}$$

If the solutions are derived from two consecutive bases Bk and Bk+1, the objective function value cannot remain constant, and a cycle cannot occur.

Conjecture 1: If $\theta\,(s,\ e)\ \bar{c}e = 0$, then necessarily:

$$\theta\,(\text{t, i})\ \bar{c}J = \text{o}\ \forall\ \ j \in J\ Nk \Longrightarrow \bar{b_t} = 0 \tag{17}$$

In this case, the current solution will never improve using the simplex algorithm. Furthermore, the LP problem has no solution.

6. RESULT

The new optimum pivot rule's performance was assessed by contrasting it with Dantzig's original pivot rule and other approaches that were already in use inside the simplex framework. MATLAB was used to develop the method, and the number of iterations, computation time, and convergence rate were among the performance indicators.

Simulation Setup:
The following computer specs were used to run the simulations:
Processor: Intel Core i7
Memory: 16 GB RAM
Software: MATLAB R2023a

Data analysis
The considered problem
The MATLAB generator is used to produce the data at random for the LP issues under consideration. The connections (1) and (2) are how we construct the LP issue. A random matrix of order $m \times n$, SPRAND (m, n, density) is a sparse matrix with a uniform nonzero entry distribution and an estimated density of $m \times n$. P% is the density that is used.

$$A = \text{round}\ (\text{sprandn}(m. n. p)) + 0.5 \qquad (18)$$

$$P = 80 \text{ when } n < 500 \text{ and } p = 50 \text{ when } n \geq 500 \qquad (19)$$

The vector containing random elements, RANDN(1,m), is the basis on which the data b are created. A vector with random elements called RANDN(n,1) is used to produce the data c..

$$c[1{:}\ m - 1] = abs\ (round(1.5 \times randn\ ([1,\ m]))) + 1 \qquad (20)$$
and
$$c = abs\ (round(2 \times randn\ ([1,\ n]))) - 1 \qquad (21)$$

To create a bounded issue, the matrix A is subjected to an extra input restriction..
$$\sum_{j=1}^{n} \text{xi} \leq n \qquad and \qquad b(m) = n \qquad (22)$$

7. FINDINGS

We compare Dantzig's original pivot rule, which was implemented in the MATLAB programming environment, with the new optimum pivot rule to assess its performance. Additionally, the simplex approach included in MATLAB's Optimization Toolbox is contrasted with the optimum pivot rule. CPU time and the number of iterations (pivots) are the performance measures considered in the comparison. Keep in mind that SML stands for the simplex method in MATLAB, OPR for a simplex algorithm with the optimum pivot rule, and DPR for a simplex algorithm with Dantzig's pivot rule. A comparison of the average number of iterations and CPU time for OPR, DPR, and SML simplex algorithms used to solve LP issues is shown in Table 2. Compared to DPR and SML, the OPR pivot rule often requires fewer iterations and less CPU time. As the number of problem variables and constraints rises, the DPR pivot rule produces fewer iterations. The suggested pivot rule outperforms. Dantzig's pivot rule, according to simulation data. The comparison of iteration counts is shown in Table 2:

Table 2 Comparison of Iteration Counts in Dantzig's Method and the Proposed Method

No	Problem size		Optimal pivot		Dantzig's pivot		Simplex in MATLAB	
	m	n	Number of iterations	CPU	Number of iterations	CPU	Number of iterations	CPU
1	30	60	28	0.031	174	0.001	169	0.11
2	40	80	41	0.015	275	0.032	274	0.22

3	50	100	80	0.068	573	0.047	570	0.48
4	100	150	165	0.15	4468	0.82	4412	8.73
5	110	175	259	0.32	7150	1.52	7153	16.33
6	120	190	270	0.37	7399	1.84	7368	17.73
7	130	200	268	0.40	7500	2.23	7497	26.16
8	140	250	368	0.68	9582	3.63	9580	29.95
9	170	300	557	1.46	17597	11.09	17596	89.10
10	200	350	621	2.03	23541	19.58	23546	214.5
11	250	400	1084	4.78	32408	37.28	32407	482.4
12	300	450	1263	7.24	60514	9420		
13	350	500	1576	12.9	86002	267.51		
14	400	600	1846	23.16	113162	666.02		

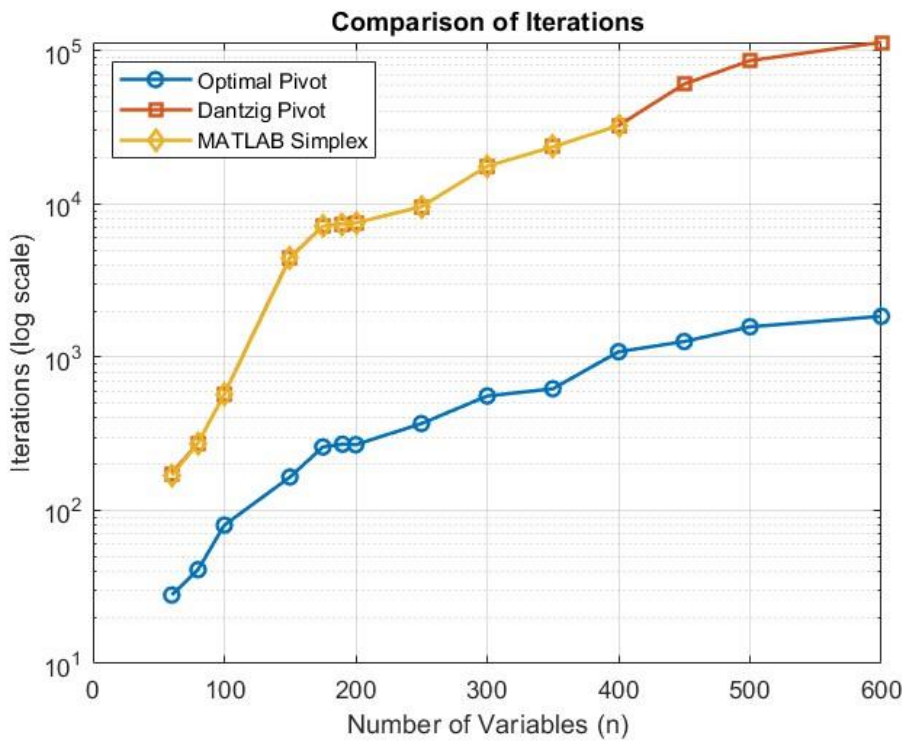

FIGURE 1 Comparison of Iteration Counts between the Proposed Optimal Pivot Rule, Dantzig's Pivot Rule, and MATLAB Simplex Method

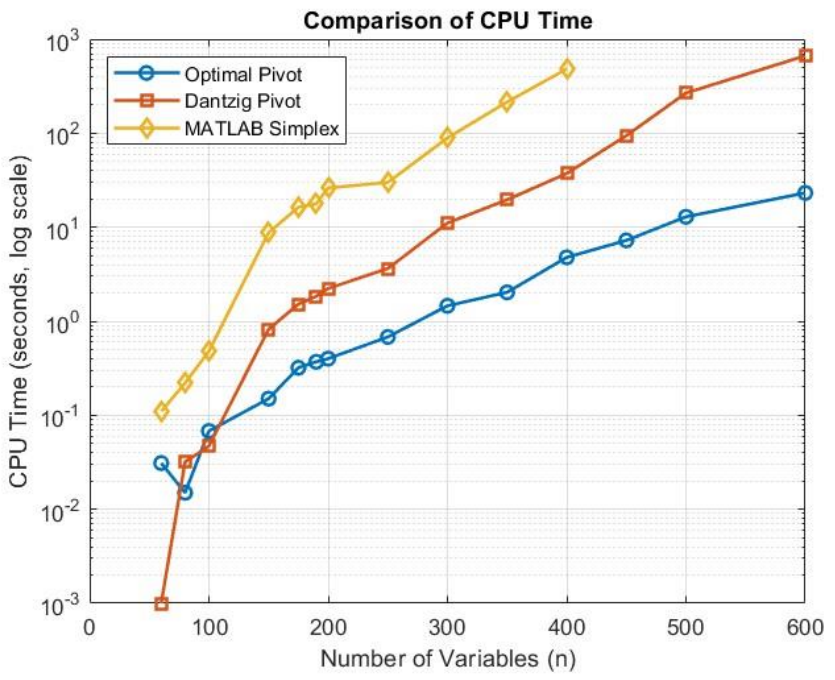

FIGURE 2 Comparison of CPU Time for the Proposed Optimal Pivot Rule, Dantzig's Pivot Rule, and MATLAB Simplex Method

8. CONCLUSIONS

The optimum pivot rule is a novel pivot rule that we suggested. This rule's concept is to determine the objective function's ideal improvement with the addition of the entering variable to the foundation. Our tests show that the suggested pivot rule outperforms Dantzig's pivot rule in terms of speed and iterations. The average number of iterations for each approach is shown in Table 2. We find that the optimum pivot rule-based simplex approach is very effective at resolving large-scale linear programming issues. The results show that the simplex method works well with the revised optimum pivot rule. Furthermore, we demonstrated that the simplex algorithm's cycle problem is resolved by the optimum pivot rule.

FUNDING

None

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their efforts.

CONFLICTS OF INTEREST

The authors declare no conflict of interest

REFERENCES

[1] R. G. Bland, "New finite pivoting rules for the simplex method," Mathematics of Operations Research, vol. 2, pp. 103–107, 1977. doi: 10.1287/moor.2.2.103.

[2] K. Chankong, B. Intiyot, and K. Sinapiromsaran, "Absolute change pivot rule for the simplex algorithm," in Proc. Int. MultiConference of Engineers and Computer Scientists, Hong Kong, Mar. 12–14, 2014, pp. 1209–1213.

[3] J. J. Forrest and D. Goldfarb, "Steepest-edge simplex algorithm for linear programming," Mathematical Programming, vol. 57, pp. 341–374, 1992. doi: 10.1007/BF01581089.

[4] R. J. Vanderbei, Linear Programming: Foundations and Extensions, 5th ed., Springer, 2020.

[5] K. Kitahara and S. Mizuno, "Steepest-edge rule and its number of simplex iterations for a nondegenerate LP," Operations Research Letters, vol. 47, no. 2, pp. 141–145, 2019.

[6] N. Ploskas and N. Samaras, "GPU accelerated pivoting rules for the simplex algorithm," Journal of Systems and Software, vol. 98, pp. 26–43, Feb. 2014.

[7] H. Lu and D. Applegate, "Scaling up linear programming with PDLP," Google Research Blog, Sep. 20, 2024. [Online]. Available: https://ai.googleblog.com

[8] D. Applegate, H. Lu, et al., "PDLP: A practical first-order method for large-scale linear programming," arXiv preprint arXiv:2501.07018, 2025.

[9] Yinyu Ye and David G. Luenberger," Interior-Point Methods", International Series in Operations Research and Management Science (ISOR, volume 228) , 2021.

[10] O. Friedmann, T. D. Hansen, and U. Zwick, "Subexponential lower bounds for randomized pivoting rules for the simplex algorithm," in Proc. 43rd ACM Symp. Theory Comput. (STOC), 2011, pp. 283–292.

[11] D. Dadush, N. Halman, and S. Pokutta, "Pivot rules for circuit-augmentation algorithms in linear programming," SIAM Journal on Optimization, vol. 32, no. 3, pp. 1789–1815, 2022.

[12] T. Terlaky and S. Zhang, "Pivot rules for linear programming: A survey on recent theoretical developments," Annals of Operations Research, vol. 46, pp. 203–233, 1993.

[13] D. Dadush, N. Halman, and S. Pokutta, "Pivot rules for circuit-augmentation algorithms in linear programming," SIAM Journal on Optimization, vol. 32, no. 3, pp. 1789–1815, 2022.

[14] T. Liu, Z. Wang, and S. You, "Learning to pivot as a smart expert," in Proc. AAAI Conf. Artificial Intelligence, vol. 38, no. 5, pp. 6234–6242, 2024.

[15] J. B. Etoa, "New optimal pivot rule for the simplex algorithm," Advances in Pure Mathematics, vol. 6, pp. 647–658, 2016.

[16] P.-Q. Pan, "A largest-distance pivot rule for the simplex algorithm," European Journal of Operational Research, vol. 187, pp. 393–402, 2008. doi: 10.1016/j.ejor.2007.03.026.

[17] M. Tipawanna and K. Chankong, "Max-out-in pivot rule with cycling prevention for the simplex method," ScienceAsia, vol. 40, no. 3, pp. 248–256, 2014.

[18] S. H. Nguyen and P. Q. Pan, "An adaptive steepest-edge pivot rule for large-scale simplex optimization," European Journal of Operational Research, vol. 315, no. 1, pp. 145–159, 2025.

[19] R. K. Patel, A. S. Rao, and M. T. Singh, "Hybrid GPU–CPU implementation of the simplex algorithm using dynamic pivot selection," Journal of Computational and Applied Mathematics, vol. 439, pp. 115678, 2024.

[20] S. Gao and W. Tang, "A fast simplex algorithm for linear programming using largest-distance pricing," Numerical Mathematics: Theory, Methods and Applications, vol. 13, no. 3, pp. 678–695, 2020.

[21] A. Li, "On the optimal pivot path of simplex method for linear programming," arXiv preprint arXiv:2210.02945, 2022.

[22] A. Li, “Optimal pivot path of the simplex method for linear programming,” Science China Mathematics, vol. 67, pp. 115–132, 2024.
[23] M. Tipawanna and K. Chankong, “Max-out-in pivot rule with cycling prevention for the simplex method,” ScienceAsia, vol. 40, no. 3, pp. 248–256, 2014.
[24] A. Shafaei, M. Rezazadeh, and H. Sarbazi-Azad, “A hardware–software co-design approach for the simplex algorithm,” in Proc. ACM/IEEE Conf. Computing Frontiers (CF), 2025, pp. 233–242.
[25] T. Illés and T. Terlaky, “Pivot rules for the simplex method: A computational study with anti-cycling rules,” Optimization Online, pp. 1–29, 2012.