A Non-Iterative Basis Selection Algorithm for Linear Programming Using Directional Cosines

F. E Ukrakpor[1], H. O. Orugba[2*], D. Ukpenusiowho[3]

[1]Department of Mechanical Engineering, Delta State University, Abraka, Nigeria.
[2]Department of Chemical Engineering, Delta State University, Abraka, Nigeria.
[3]Department of Software Engineering, Southern Delta University, Ozoro, Nigeria.

ABSTRACT: This paper presents a geometry-based non-iterative method for solving linear programming (LP) problems that have either three constraints or three decision variables. Instead of using iterative procedures like the simplex, Interior-point, and Criss-Cross methods, the method finds a basic feasible solution consistent with optimal results for the class of problems considered. The algorithm measures the directional cosines alignment between each constraint vector and the resource vector, picks the three most aligned constraints to form a 3×3 Dantzig matrix, and solves that small linear system for the decision variables and objective value. The algorithm was tested on standard textbook examples, and the results match conventional solvers exactly, but are obtained without pivoting or tableau updates, just one matrix inversion. The approach is fast and provides a very clear geometric visualization, making it useful as a pedagogical tool or as an initialization step in larger routines.

KEYWORDS: Linear programming, non-iterative algorithm, directional cosine alignment, Dantzig matrix, geometric optimization

[Received Dec. 11, 2025; Revised Apr 10, 2026; Accepted June 18, 2026] Print ISSN: 0189-9546 | Online ISSN: 2437 2110

I. INTRODUCTION

Linear programming (LP) has been a potent analytical framework for decision-making in engineering, economics, logistics, production planning, and many other fields (Vasant, 2003; Srinivasan, 2014; Golden et al., 2024). Fundamentally, LP is the optimisation of a linear objective function under a set of linear constraints that specify a feasible area in a multidimensional space. A basic feasible point located at a vertex of this region, where all constraints are simultaneously satisfied, is the solution to an LP problem. This geometric foundation, though conceptually simple, serves as the basis for some of operations research's most adaptable and computationally reliable methods.

The Simplex algorithm, created by Dantzig in 1947 and still the mainstay of linear optimisation, serves as the foundation for the historical development of LP techniques. The elegance of the Simplex method lies in its methodical movement along the edges of the feasible polytope, efficiently locating an optimal vertex through a series of pivot operations (Loera et al., 2021). Notwithstanding its resilience, the approach is iterative by nature, necessitating numerous tableau updates (Pan, 2014), basis swaps, and feasibility modifications—particularly in cases of degeneracy or impracticability. Its implementation is made more difficult by the addition of auxiliary problems and artificial variables to determine feasibility, especially in pedagogical or small-scale computational contexts.

Later advancements in LP algorithms aimed to overcome the Simplex method's computational and iterative drawbacks. A significant advance was made when Karmarkar (1984) presented the interior-point algorithm, which provides a polynomial-time substitute that travels through the interior of the feasible region rather than its boundary. Subsequent modifications, like the least-pivoting index method (Li, 2010), attempted to improve convergence efficiency through geometric and algebraic refinements, while Zionts (1969) attempted to enhance convergence efficiency through geometric and algebraic refinements when he introduced the Criss-Cross algorithm. However, these methods continued to be iterative, involving several updates to achieve optimality and frequently involving computational complexity that does not scale well with the size of the problem (Ma et al., 2015; Sokolinsky & Sokolinskaya, 2023).

For small or structured LP systems, a number of researchers have looked at algebraic and geometric shortcuts in addition to traditional approaches. Despite its limited scope, the graphical approach for two-variable LPs provides a clear illustration of the geometric foundation of linear optimisation. For low-dimensional problems, computational methods have more recently investigated hybrid decomposition techniques, reduced-basis selection, and direct algebraic inversion (Fábián et al., 2013). However, the theoretical elegance and simplicity of these strategies are limited because the majority either maintain partial iteration or rely on artificial constructs to ensure feasibility. A straightforward, geometry-based technique that takes advantage of the inherent alignment between constraint vectors and the resource vector to find an ideal basis non-iteratively is still mainly unexplored.

Such an approach has a special opportunity in the geometry of low-dimensional LP problems, especially those with three constraints or three decision variables. The feasible region in these problems is located in a three-dimensional requirement space that allows for the explicit visualisation and quantification of geometric relationships between the objective vector and the constraints. A natural metric for assessing how well constraint vectors and resource vectors

*Corresponding author: orugbahenry@gmail.com
doi: http://doi.org/10.63746/njtd.v23i2.4261

align is provided by the idea of directional cosines. Without using artificial variables or iterative pivoting, it is feasible to directly construct a basis matrix whose inverse yields the basic feasible solution by choosing the three constraints that are most geometrically aligned with the direction of the resource vector.

A non-iterative basis selection algorithm for LP problems of small dimensionality is proposed in this study, building on this geometric concept. The technique determines the three vectors that are most closely aligned, calculates directional cosines between constraint vectors and the resource vector, and creates what could be called the Dantzig matrix. This matrix is then directly inverted to obtain the corresponding basic feasible solution, and systematic geometric adjustments are used to handle special cases involving ties, redundancy, or singularity.

Unlike traditional iterative solvers, this approach is straightforward, transparent, and easily interpretable geometric one-step computational framework. Because of its non-iterative nature, it is especially well-suited for sensitivity analysis, educational demonstration, and real-time or embedded applications where algorithmic clarity and computational speed are desired. The method provides a conceptual path towards generalising non-iterative and geometry-driven LP solvers for higher dimensions, despite being limited to problems with three constraints or decision variables.

Therefore, by reintroducing geometric reasoning into algorithmic design, the paper adds to the ongoing discussion in computational linear programming. It provides a unique but intuitively based viewpoint on LP computation by fusing basis selection theory with vector alignment principles. The theoretical background, algorithm mathematical formulation, and performance evaluation against standard LP problems are developed, which also reveal the algorithm's limitations and potential within the larger framework of contemporary optimisation theory.The method is therefore presented as a geometry-based basis selection approach whose effectiveness is demonstrated for specific classes of low-dimensional LP problems.

II. THEORETICAL BACKGROUND

Linear programming problems are generally formulated as the optimization of a linear objective function subject to a finite set of linear constraints (Corriou, 2021). In algebraic form, a standard LP problem can be expressed by Equation 1.

$$maximize\ z = C^T x \quad (1)$$

Subject to:

$$Ax = b \quad (2)$$

$$x \geq 0$$

where $x \in R^n$ represents the vector of decision variables, $A \in R^{m \times n}$ is the constraint matrix, $b \in R^m$ is the resource vector, and $c \in R^n$ contains the coefficients of the objective function. The feasible region defined by these linear constraints is a convex polyhedron, and the optimal solution lies at one of its extreme points or vertices (Dantzig, 1947). Each vertex corresponds to a basic feasible solution (BFS) obtained by selecting a subset of m linearly independent columns of A, forming the basis matrix B. The basic solution is then given by Equation 3.

$$x_B = B^{-1} b \quad (3)$$

$$x_N = 0$$

where x_B and x_N denote basic and non-basic variables, respectively. The Simplex algorithm navigates among these bases iteratively to find the one that maximizes (or minimizes) z.

The geometric interpretation of this process reveals crucial information. The feasible region is the intersection of all half-spaces enclosed by the hyperplanes that each constraint defines in n-dimensional space. A vertex of the feasible polyhedron is defined by the intersection of precisely m linearly independent hyperplanes, while the resource vector b specifies a direction from the origin towards the feasible region. The intersection that most closely matches the gradient of the objective function c is the optimal vertex. This geometric interpretation can be related to dual feasibility, where the active constraints at optimality satisfy complementary slackness conditions. Establishing this connection more formally could further strengthen the theoretical basis of the method. The non-iterative method suggested in this work is conceptually based on this interaction between constraint geometry and directional alignment.

By iteratively exploring adjacent vertices under the guidance of pivoting rules based on reduced costs or feasibility conditions, the optimal basis is found in classical algorithms like the Simplex or criss-cross methods. However, when the dimensionality of the problem is low—specifically when m=3 or n=3—the entire geometric configuration of the feasible region can be directly visualized in three-dimensional space. Under these conditions, the optimal solution can be approached geometrically rather than through successive tableau operations.

Let the constraint matrix $A = [a_1, a_2, \ldots, a_n]$ consist of n column vectors $a_j \in R^3$, and let $b \in R^3$ represent the resource vector. The objective of the non-iterative algorithm is to identify the subset of three constraint vectors that form a basis most geometrically aligned with b. The degree of alignment between two vectors is conveniently quantified by the directional cosine defined by Equation 4.

$$cos\theta_j = \frac{a_j . b}{\|a_j\| . \|b\|} \quad (4)$$

where θ_j is the angle between the constraint vector a_j and the resource vector b. A higher cosine value (close to 1) indicates that a_j is more closely aligned with b. By ranking the constraint vectors according to their cosine values and selecting the three that exhibit the smallest angular separation from b, one can construct a Dantzig matrix $D = [a_{j1}, a_{j2}, a_{j3}]$ composed of the most aligned constraints. Provided that these vectors are linearly independent, D serves as the basis matrix for the optimal or near-optimal basic feasible solution. In this formulation, the vectors a_j are consistently treated as column vectors of the constraint matrix, representing constraint directions in the defined space. The geometric alignment is evaluated relative to the resource vector b, ensuring a consistent interpretation throughout the analysis.

The geometric property that, at the optimal vertex, the active constraints—those defining the intersection point—tend to be orientated in directions that collectively "enclose" the resource vector within the feasible region provides the

theoretical basis for this choice. These active constraints are equivalent to the dual variables that satisfy complementary slackness in the dual sense. Thus, the cosine-based criterion offers a geometric stand-in for the Simplex method's reduced cost test, substituting a single-step assessment of vector alignment for iterative updates.

When n=3 and the number of constraints m exceeds three, the same reasoning can be applied through the dual formulation of the LP problem. The dual expresses the same optimization structure from the perspective of constraints rather than variables, allowing a consistent application of the cosine selection rule to identify the three dual vectors most aligned with the dual resource vector. The resulting dual Dantzig matrix can then be inverted to obtain the dual basic solution, which is subsequently mapped back to the primal solution via the principle of complementary slackness.

In contrast to conventional iterative solvers, this method provides a one-step computational framework that is simple, transparent, and easily interpretable in geometric terms. Its non-iterative nature makes it particularly suited for educational demonstration, sensitivity exploration, and real-time or embedded applications where computational speed and algorithmic clarity are desirable. While the approach is restricted to problems with three constraints or decision variables, it opens a conceptual pathway toward generalizing non-iterative and geometry-driven LP solvers for higher dimensions. It is important to understand the theoretical limitations of the suggested geometric basis selection, even though it is an intuitively straightforward method for calculating simple feasible solutions. Because alignment by itself does not adequately capture the orientation of the objective function or the subtleties of feasibility in degenerate systems, the method does not always guarantee optimality. Nonetheless, the cosine alignment criterion provides a logical and computationally effective approximation for non-degenerate, small-dimensional problems, which closely resembles optimal basis selection in real-world scenarios.

Thus, the theoretical background provided here creates a link between geometric vector analysis and classical LP theory. It serves as the conceptual foundation for the non-iterative algorithm created in the next section, where the geometric selection process is codified into a computational process that manages degeneracy, redundancy, and singularity in a methodical manner without the need for iterative tableau transformations or artificial variables.

III. METHODOLOGY

In order to solve low-dimensional linear programming problems, the proposed technique converts the geometric concepts of constraint alignment into a non-iterative computational framework. This method focuses on LP systems with either three decision variables and an arbitrary number of constraints or three constraints and an arbitrary number of decision variables. The same geometric structure is represented twice in three dimensions in these two instances.

A. Problem Standardization and Scope

Each LP problem must first be expressed in a standard form that can be understood geometrically in order to use the proposed algorithm correctly. This entails making sure the variables are not negative and rewriting all constraints in equality form. To keep the orientation of the feasible half-spaces consistent, inequality constraints of the form $a_i^T x \geq b_i$ are multiplied by −1, while those of the form $a_i^T x \leq b_i$ are left unchanged. To capture the same geometric boundary without adding extraneous variables, equality constraints are expressed as paired inequalities. This pre-processing step guarantees that the resource vector b and all constraint vectors a_i are located in the same directional frame of reference in R^3. The algorithm assumes that the problem is non-degenerate, meaning that no more than three constraints intersect at a single vertex in the feasible region. When degeneracy exists, the method still yields a valid basic feasible solution, but additional verification of optimality may be required.

B. Geometric Rationale

Each decision variable has a vector a_j that points from the origin in the direction of the variable's contribution to the constraints in the requirement space that the constraints define. The ideal combination of these contributions that meet the constraints is defined by the resource vector b. Finding the subset of three constraint vectors that are most geometrically aligned with b is the fundamental concept of the non-iterative method. When inverted, this subset creates a basis that yields the fundamental feasible solution that represents the intersection of the three most significant constraint planes.
The geometric alignment between any constraint vector a_j and the resource vector b is quantified using the directional cosine given in Equation 5.

$$cos\theta_j = \frac{a_j.b}{\|a_j\|.\|b\|} \quad (5)$$

where θ_j represents the angle between the two vectors. The closer $cos\theta_j$ is to unity, the smaller the angle between a_j and b, indicating a stronger geometric alignment. This metric forms the basis for selecting the optimal constraint set. This procedure transforms the abstract concept of geometric alignment into a measurable and rankable quantity, thereby enabling direct computational implementation without iterative search.

C. The Non-Iterative Algorithm

After standardising the problem and computing the directional cosines, the cosine values of the constraint vectors are used to rank them in descending order. The Dantzig matrix D= [a_{j1}, a_{j2}, a_{j3}] is a rough selection of the three vectors with the largest cosines, which correspond to the smallest angular separation from b. The chosen vectors' linear independence is then confirmed by evaluating the determinant of D. The matrix is invertible if |D| ≠ 0, and the fundamental feasible solution can be found using Equation 6.

$$x_B = D^{-1}b \quad (6)$$

with all non-basic variables set to zero. This operation yields a direct solution in a single computational step without iterative tableau transformations or pivoting sequences.

In cases where the determinant of D equals zero, indicating linear dependence among the selected vectors, one of the vectors is replaced with the next-ranked vector in the cosine list until a nonsingular matrix is obtained. This

substitution ensures that the chosen set spans the three-dimensional space and that the resulting intersection corresponds to a unique vertex of the feasible region. The calculated solutions and objective values were validated by comparing them with the results of well-known LP solvers (the Simplex, Interior-Point, and Criss-Cross algorithms), which were implemented in Python. The iteration counts, convergence behaviours, and final objective values of each solver were documented after they were run on identical benchmark problems.

The efficiency and dependability of the suggested algorithm in comparison to traditional iterative techniques were quantified by this comparative analysis.

The computational procedure of the proposed method can be summarised as follows:

Step 1: Convert the LP problem into standard form ensuring that all constraints are expressed consistently and variables are non-negative.

Step 2: Represent each constraint as a vector a_j and define the resource vector b.

Step 3: Compute the directional cosine between each constraint vector and the resource vector using the relation in Equation 4

Step 4: Rank all constraint vectors in descending order based on their cosine values.

Step 5: Select the top three vectors with the highest cosine values to form the Dantzig matrix D.

Step 6: Check linear independence of the selected vectors by evaluating

$$det\ (D)\ \neq\ 0$$

Step 7: Compute the basic feasible solution using the relation in Equation 6.

Step 8: Verify feasibility (non-negativity and constraint satisfaction). If violated, replace one vector with the next ranked vector and repeat the check.

Step 9: Compute the objective function value using the obtained solution.

This structured procedure ensures that the solution is obtained through a direct analytical pathway rather than successive approximations, distinguishing it from conventional iterative solvers.

D. Handling Special Cases

Numerical relationships between directional cosines, coplanar constraints, and geometric redundancies are examples of real-world LP problems. The algorithm includes particular rules for these situations to guarantee robustness. When two vectors have the same cosine value, the vector with the largest Euclidean norm is preferred because it usually indicates a constraint that has a larger geometric impact on the boundary of the feasible region. When two or more constraints are almost coplanar, the redundant constraints are swapped out for unit vectors along the coordinate axes, leaving only the most dominant constraint. For matrix D, this substitution stabilises the inversion process while preserving linear independence.

The procedures described in this section represent auxiliary steps for handling special or degenerate cases and are not part of the primary basis selection mechanism. An alternate set of vectors is tested from the ranked list if the chosen basis yields a solution that deviates from constraint feasibility or non-negativity. This validation guarantees that, within numerical tolerance, the final basic feasible solution satisfies all constraints. Although limited substitution may be required when infeasibility or singularity is detected, the method avoids systematic iterative pivoting and does not rely on tableau-based progression, thereby retaining its essentially non-iterative structure.

E. Dual Representation for N=3 Problems

The dual formulation follows the same procedure for problems with three decision variables and more constraints than three. Each primal constraint in the dual representation corresponds to a dual variable, and the same cosine alignment rule can be used to determine the ideal dual basis. The row vectors of the primal constraint matrix that show the best alignment with the dual resource vector are used to create the corresponding dual Dantzig matrix. To guarantee consistency between the two formulations, the dual solution is mapped back to the primal space using complementary slackness after it has been obtained.

F. Computational Efficiency and Complexity

For small-dimensional LP problems, this method's non-iterative nature significantly lowers computational overhead. Evaluating the directional cosines for each constraint vector is the main computational task, with a time complexity of O(n). This is followed by the inversion of a 3×3 matrix, which has a constant complexity of O(1). Apart from this analytical process, the study assessed performance metrics from the Simplex, Interior-Point, and Criss-Cross algorithms for direct comparison, including solver iteration counts and convergence rates.

In contrast to multiple iterative updates in traditional solvers, the suggested algorithm usually only requires a single matrix inversion, achieving identical objective values with significantly fewer computational steps, as demonstrated by this multi-solver benchmarking. On the other hand, the Simplex algorithm necessitates iterative tableau operations whose complexity increases in direct proportion to the number of variables and constraints. Therefore, the suggested method uses a fraction of the computational effort while achieving comparable accuracy for low-dimensional problems. However, the method's effectiveness depends on precisely identifying feasible and linearly independent constraint sets. The directional cosine criterion may produce less-than-ideal or impractical bases in degenerate configurations or poorly conditioned systems. As a result, the method works best for small-dimensional non-degenerate LP problems, where its geometric simplicity and computational immediacy can be fully utilised.

IV. RESULTS AND DISCUSSION

Fifteen LP problems from standard references (Sharma, 2009; Stroud, 2011) were used to assess the suggested geometry-based, non-iterative algorithm. Primal and dual formulations involving three constraints with multiple decision variables or three variables with multiple constraints

are included in these problems. To illustrate the computation procedure, confirm accuracy, and contrast the outcomes with current LP techniques, four representative examples are provided.

All computations were implemented in Python. The algorithm's performance was compared with three established LP solution techniques: the Simplex method, the Interior-Point method, and the Criss-Cross algorithm. These represent, respectively, the classical vertex-based approach, a polynomial-time interior traversal algorithm, and a combinatorial pivoting method known for handling infeasibility without artificial variables. Furthermore, a real-world engineering application example on optimal resource allocation has been included to demonstrate the engineering application of the proposed method and establish its usefulness in engineering decision-making.

A. Representative Test Cases

The following examples correspond to those presented in classical LP texts and were used to evaluate the accuracy and efficiency of the proposed algorithm.

Example 1: Three Constraints and Three Variables (Stroud, 2011, p. 961)

$$maximize \quad P = 3x_1 + 4x_2 + 5x_3$$

$$\text{Subject to:} \begin{cases} 2x_1 + 4x_2 + 3x_3 \leq 80 \\ 4x_1 + 2x_2 + x_3 \leq 48 \\ x_1 + x_2 + 2x_3 \leq 40 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

The constraint vectors were expressed as

$a_1 = [2,4,1]^T, \quad a_2 = [4,2,1]^T, \quad a_3 = [3,1,2]^T, \quad b = [80,48,40]^T$

The Dantzig matrix D=[a_1, a_2, a_3] was formed and inverted to yield

$x_B = D^{-1}b = [5,7,14]^T$

$P_{max} = 3(5) + 4(7) + 5(14) = 113$

The Simplex, Interior-point and Criss-Cross methods produce the same optimum; iterative solvers required multiple steps whereas the proposed method obtains the same result by a single selection.

Example 2: Four constraints, three variables (Stroud, 2011, pg 979, Example 2)

$$maximize \quad P = 84x_1 + 72x_2 + 52x_3$$

$$\text{Subject to} \begin{cases} 2x_1 + x_2 + 2x_3 \leq 98 \\ x_1 + x_2 + x_3 \leq 60 \\ 3x_1 + 4x_2 + 2x_3 \leq 145: \\ 4x_1 + 3x_2 + 2x_3 \leq 160 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

For this over-determined system, the dual formulation was used. After directional-cosine ranking and redundancy handling, the dual active set a_4, a_3, a_1 was chosen. Solving the corresponding (dual) Dantzig system yields

$x_B = [23,8,22]^T$

$P_{max} = 84(23) + 72(8) + 52(22) = 3652$

Conventional solvers agree with this value; the dual approach with the proposed selection recovers the same primal optimum without iterative pivoting in the primal tableau.

Example 3: Three constraints, four variables (Sharma, 2009, pg 134, Example 3)

$$maximize \quad P = 10x_1 + 8x_2 + 7x_3 + 10x_4$$

$$\text{Subject to:} \begin{cases} 2x_1 + x_2 + x_3 + x_4 \leq 50 \\ x_1 + 2x_2 + x_3 + x_4 \leq 50 \\ x_1 + x_2 + 2x_3 + x_4 \leq 50 \\ x_1, x_2, x_3, x_4 \geq 0 \end{cases}$$

The cosine test ranked the constraints and selected a_4, a_2, a_3 as the active set. Solving $D^{-1}b$ produces

$x_B = [0,50,50,0]^T$

$P_{max} = 10(0) + 8(50) + 7(50) + 10(0) = 750$

This result matches the Simplex and Interior-Point outputs; the algorithm handles the underdetermined primal case by selecting the appropriate basis and computing the basic variables directly.

Example 4: Three constraints, five variables (Stroud, 2011, pg 985, Example 5)

$$maximize \quad P = 5x_1 + 3x_2 + 2x_3 + 4x_4 + 6x_5$$

$$\text{Subject to:} \begin{cases} x_1 + 2x_2 + x_3 + x_4 + x_5 \leq 100 \\ 2x_1 + x_2 + x_3 + x_4 + 2x_5 \leq 80 \\ x_1 + x_2 + 2x_3 + x_4 + x_5 \leq 60 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases}$$

Directional cosines indicated selection of a_3, a_1, a_4. Solving the Dantzig system yields

$x_B = [20,0,30,10,0]^T$

$P_{max} = 5(20) + 3(0) + 2(30) + 4(10) + 6(0) = 200$

Again, standard solvers report the same optimum; the proposed method computes it with a single selection and inversion.

B. Comparative Performance

All solvers produced the same objective values as presented in Table 1.

Table 1 Comparative performance of some common LP solvers

Method	Example 1 (Stroud, 2011)	Example 2 (Stroud, 2011)	Example 3 Sharma (2009)	Example 4 (Stroud, 2011)
Variables × Constraints	3 × 3	3 × 4	4 × 3	5 × 3
Simplex (Iterations)	multiple pivots	multiple pivots	multiple pivots	multiple pivots
Interior-Point (Steps)	multiple steps	multiple steps	multiple steps	multiple steps
Criss-Cross (Pivots)	multiple pivots	multiple pivots	multiple pivots	multiple pivots
Proposed algorithm (selection+inversion)	1	1	1	1
Objective value	113	3652	750	200

As revealed in Table 1, for LPs with three constraints or three decision variables, these illustrative examples show that the proposed technique consistently finds active constraint sets that produce the best basic feasible solution. The proposed algorithm computes the result in a single analytic step (selection followed by 3×3 inversion), providing a definite computational and interpretive advantage for low-dimensional problems. In each case, the computed solution matched those from well-known iterative solvers. In this context, "single step" refers specifically to the direct computation of the solution through matrix inversion once the basis has been selected. Preprocessing operations such as cosine evaluation, ranking, and feasibility checks are not considered iterative in the classical sense.

In near-degenerate configurations (as used in Examples 2 and 3), the technique necessitates tie-breaking and redundancy handling. In practical deployments, it must be used independently to check for unboundedness and infeasibility. Considering these drawbacks, the algorithm is not a perfect substitute for full-scale iterative solvers but rather works best as a direct basis-selection method for small LPs or as a quick initialization/preprocessing tool for larger optimisation workflows. The method may encounter limitations in near-degenerate or poorly conditioned systems, where cosine-based ranking may not uniquely identify the optimal basis. In such cases, additional validation or alternative basis selection may be required, highlighting the importance of applying the method within its intended scope.

C. Engineering Application Example

To demonstrate the practical utility of the proposed algorithm in an engineering context, a typical product-mix optimization problem encountered in mechanical and chemical engineering industries was given.

Example 5: Optimal Resource Allocation in a Manufacturing Plant

A certain manufacturing plant produces three different components (P1, P2, and P3) for industrial equipment. The production process is constrained by three limited resources: raw material availability, machine hours on CNC equipment, and skilled labour hours. The goal is to determine the optimal production quantities that maximize daily profit without exceeding available resources.

The linear programming model is formulated as follows:

The LP formulation is:

$$maximize \quad Z = 12x_1 + 11x_2 + 13x_3$$

Subject to:

$$2x_1 + x_2 + x_3 \leq 20 \ (raw\ material, units)$$
$$x_1 + 2x_2 + x_3 \leq 20 \ (machine\ capacity, hour)$$
$$x_1 + x_2 + 2x_3 \leq 20 \ (labour, hours)$$
$$x_1, x_2, x_3 \geq 0$$

The constraint vectors were expressed as

$a_1 = [2,1,1]^T, \quad a_2 = [1,2,1]^T, \quad a_3 = [1,1,2]^T,$

$b = [20,20,20]^T$

$$D = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

Inverting D and computing,

$x_B = D^{-1}b$ yields the basic feasible solution $x_1 = 5, x_2 = 5, x_3 = 5$

$Z_{max} = 180$

The Simplex, Interior-Point, and Criss-Cross methods produce exactly the same optimum. Conventional iterative solvers require multiple pivots or steps, whereas the proposed method obtains the identical result by a single matrix inversion.

XIII. CONCLUSION

In order to solve linear programming problems with three constraints or three decision variables, this study presented a non-iterative algorithm based on geometry. The technique provides a direct computational route to a basic feasible solution without the need for tableau construction or iterative pivoting by employing directional cosine alignment to identify the most relevant constraint vectors and constructing a Dantzig matrix for direct inversion. The suggested algorithm consistently reproduced the same objective values as the Simplex, Interior-point, and Criss-Cross methods, according to results from standard benchmark problems taken from Stroud (2011) and Sharma (2009).

The algorithm was able to reproduce optimal solutions in a single matrix-inversion operation for the class of problems considered, whereas traditional solvers needed multiple iterative steps. This demonstrates that the suggested method for small-dimensional LP systems is accurate and computationally efficient. The algorithm's simplicity and transparency make it an effective analytical and pedagogical tool, especially for rapidly validating solutions to low-dimensional problems and for demonstrating geometric relationships in LP. Currently, its use is restricted to non-degenerate systems with an invertible 3x3 Dantzig matrix. Its reliability under degeneracy and near-collinearity conditions still needs theoretical validation, and it does not naturally handle infeasibility, redundancy, or unboundedness.

The algorithm's extension to higher dimensions, formalisation of its optimality and feasibility proofs, and integration of automated tie-breaking and degeneracy checks will be the main goals of future research. In order to combine its geometric efficiency with the robustness of full-scale optimisation engines, integration with current solver frameworks as an initialisation module or pre-processing stage will also be investigated. The proposed approach should be regarded as a heuristic basis selection method unless formal optimality guarantees are established. The method is best suited for low-dimensional LP problems, pedagogical applications, and as a rapid initialization tool within larger optimisation frameworks. It is not intended to replace established LP solvers but to complement them by providing a fast and interpretable basis selection mechanism.

REFERENCES

Corriou, J. P. (2021). *Linear programming*. In *Numerical methods and optimization* (Vol. 187, pp. 1–24). Springer, Cham. https://doi.org/10.1007/978-3-030-89366-8_10

Dantzig, G. B. (1947). *Linear programming*. Princeton University Press.

Fábián, C. I., Papp, O., & Eretnek, K. (2013). Implementing the simplex method as a cutting-plane method, with a view to regularization. *Computational Optimization and Applications, 56*(2), 343–368. https://doi.org/10.1007/s10589-013-9565-9

Golden, B., Schrage, L., Shier, D. Apergi L.A. (2024). The unexpected power of linear programming: an updated collection of surprising applications. *Annals of Operation Research,* 343, 573–605. https://doi.org/10.1007/s10479-024-06245-5

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica, 4*(4), 373–395. https://doi.org/10.1145/800057.808695

Li, W. (2010). Practical criss-cross method for linear programming. In L. Zhang, B. L. Lu, & J. Kwok (Eds.), *Advances in neural networks – ISNN 2010* (Lecture Notes in Computer Science, Vol. 6063, pp. 234–243). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13278-0_29

Loera, J., Kafer, S., & Sanità, L. (2021). On the simplex method for 0/1-polytopes. *Mathematics of Operations Research, 50*(4), 1398–1420. https://doi.org/10.1287/moor.2021.0345

Ma, Y., Zhang, L., & Pan, P. (2015). Criss-cross algorithm based on the most-obtuse-angle rule and deficient basis. *Applied Mathematics and Computation, 268*, 439–449. https://doi.org/10.1016/j.amc.2015.06.080

Pan, P. Q. (2014). Interior-point method. In *Linear programming computation*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40754-3_9

Sharma, J. K. (2009). *Operations research: Theory and applications* (4th ed.). Macmillan.

Sokolinsky, L. B., & Sokolinskaya, I. M. (2023). Apex method: A new scalable iterative method for linear programming. *Mathematics, 11*(7), 1654. https://doi.org/10.3390/math11071654

Srinivasan, R. (2014). Introduction to linear programming. In *Strategic business decisions*. Springer, New Delhi. https://doi.org/10.1007/978-81-322-1901-9_2

Stroud, K. A. (2011). *Advanced engineering mathematics* (5th ed.). Palgrave Macmillan.

Vasant, P. M. (2003). Application of fuzzy linear programming in production planning. *Fuzzy Optimization and Decision Making, 2*(3), 229–241. https://doi.org/10.1023/A:1025094504415

Zionts, S. (1969). The criss-cross method for solving linear programming problems. *Management Science, 15*(7), 426–445. https://doi.org/10.1287/mnsc.15.7.426